DGK Ausschuss Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften

Reihe C | Dissertationen | Heft Nr. 970

Mirjana Voelsen

Combining Convolutions and Attention for Land Cover Classification of Satellite Image Time Series

München 2025

Verlag der Bayerischen Akademie der Wissenschaften, München

ISSN 0065-5325 ISBN 978 3 7696 5382 3

DGK Ausschuss Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften

Reihe C Dissertationen Heft Nr. 970

Combining Convolutions and Attention for Land Cover Classification of Satellite Image Time Series

Von der Fakultät für Bauingenieurwesen und Geodäsie
der Gottfried Wilhelm Leibniz Universität Hannover
zur Erlangung des akademischen Grades
Doktor-Ingenieurin (Dr.-Ing.)
genehmigte Dissertation

von

Mirjana Voelsen, M. Sc.

München 2025

Verlag der Bayerischen Akademie der Wissenschaften, München

ISSN 0065-5325 ISBN 978 3 7696 5382 3

Adresse des Ausschusses Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften:

Ausschuss Geodäsie (DGK) der Bayerischen Akademie der Wissenschaften
Alfons-Goppel-Straße 11 ● D – 80 539 München
Telefon +49 – 89 – 23 031 1113 ● Telefax +49 – 89 – 23 031 - 1283 / - 1100
e-mail post@dgk.badw.de ● http://www.dgk.badw.de

Prüfungskommission:
Vorsitzender: Prof. Dr.-Ing. habil. Christian Heipke
Referent: Apl. Prof. Dr. techn. Franz Rottensteiner
Korreferenten: Prof. Dr.-Ing. habil. Monika Sester
Prof. Dr..Ing. Raul Queiroz Feitosa

Tag der mündlichen Prüfung: 03.07.2025

:

Diese Dissertation ist auf dem Server des Ausschusses Geodäsie (DGK)
der Bayerischen Akademie der Wissenschaften, München unter <http://dgk.badw.de/>
sowie unter Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik
der Leibniz Universität Hannover (ISSN 0174-1454), Nr. 418,
<https://repo.uni-hannover.de/items/c5140660-7da9-4e17-a750-61c67e22a753>, Hannover 2025, elektronisch publiziert

© 2025 Ausschuss Geodäsie (DGK) der Bayerischen Akademie der Wissenschaften, München

Alle Rechte vorbehalten. Ohne Genehmigung der Herausgeber ist es auch nicht gestattet,
die Veröffentlichung oder Teile daraus zu vervielfältigen.

ISSN 0065-5325 ISBN 978 3 7696 5382 3

Abstract

With the availability of large amounts of satellite image time series (SITS), the Earth's surface can be monitored at large scale and with high temporal resolution so that the images can serve as a base for applications like urban planning or map updating. This work addresses the task of multi-temporal land cover (LC) classification using SITS data, specifically aiming to identify the physical material on the Earth's surface for every pixel in an image sequence. The unique combination of spectral, spatial, and temporal dimensions in SITS offers the advantage that information encoded in different dimensions can be combined to improve classification results. On the other hand, the data complexity can result in a high number of computations when it is processed jointly. A strategy to decrease the number of computations is the separate extraction of information from all individual dimensions. However, such a separation raises the question of what type of methods are suitable for which dimension, i.e how to extract information from the spectral, temporal or spatial dimensions for the case of SITS data, and how the extracted features should be fused afterwards. These questions have not yet been answered and analysed in detail for the task of multi-temporal LC classification with SITS. Another strategy to reduce the number of computations is based on the reduction of the number of input feature vectors to a Transformer model. To achieve this goal, several pixels in the input images are combined into one patch, which is then processed by the Transformer model. While this drastically reduces the number of computations in the subsequent layers, it also reduces the geometric resolution and consequently the model's ability to predict the exact position of class boundaries and fine structures when they cover an area which is represented only by a few patches.

In this thesis, a new method for multi-temporal LC classification is proposed, extending the Swin Transformer introduced by Liu et al. (2021) for pixel-wise classification to handle SITS as input and predict multi-temporal LC maps. To consider spatial and temporal dependencies from SITS data, a hybrid feature extraction module is introduced. This module processes the input data in two streams: spatial context is considered by convolutions, and temporal dependencies are considered by self-attention. This separation reduces the computational complexity and allows the usage of the optimal method in each dimension. While self-attention is expected to be well-suited to encode local and global temporal dependencies like seasonal effects, convolutions consider the local spatial neighbourhood, making them particularly effective for capturing spatial patterns. To further improve the performance of the classifier, a 3D patch generation module is introduced to mitigate the impact of patch generation. This module uses 3D convolutions to encode spatial-temporal dependencies at the original spatial resolution of the input images before any downsampling occurs.

The evaluation of the proposed method is based on two datasets, referred to as the Lower Saxony and Dynamic Earth datasets, for LC classification with SITS. The model's classification per-

formance varied across datasets, achieving mean F^1-scores between 73% and 79% on the Lower Saxony dataset and 44% on the Dynamic Earth dataset. While good performance was observed for most land cover classes, challenges arose for classes with fewer training samples or similar visual appearances. The experiments regarding the hybrid feature extraction module show that it achieves significantly better mean F^1-scores compared to a model that computes spatial and temporal features solely with self-attention. Specifically, the proposed model ($V\text{-}FE_{hyb}$) improved the mean F^1-score by 2% on the Lower Saxony test dataset and by 4.2% on the Dynamic Earth test dataset compared to the attention-based variant $V\text{-}FE_{att}$. Additionally, the new model slightly outperforms a model that jointly extracts spatial-temporal features using self-attention, while significantly reducing the computational complexity. The experiments regarding the 3D patch generation module show that this module leads to similar mean F^1-scores on one of the datasets and better mean F^1-scores on the second dataset in comparison to a model that uses the standard patch generation approach. In the end, the new method is compared to other baselines from literature. In this comparison, the proposed model performs better than models that are solely based on convolutions, which underlines that hybrid approaches are well-suited for the classification of SITS, while still performing similarly well to other hybrid approaches.

Keywords: Pixel-wise classification, remote sensing, Transformer models, satellite image time series

Kurzfassung

Durch die Verfügbarkeit großer Mengen an Zeitreihen von Satellitenbildern (SITS) kann die Erdoberfläche großflächig und mit hoher zeitlicher Auflösung beobachtet werden. Somit können SITS als Grundlage für Anwendungen wie Stadtplanung oder Aktualierungen von Datenbankbeständen genutzt werden. Diese Arbeit befasst sich mit der Aufgabe der multitemporalen Landbedeckungsklassifikation unter Nutzung von SITS als Eingangsdaten. Landbedeckung (LB) bezeichnet das physische Material auf der Erdoberfläche, das für jedes Pixel in den Eingangsbildern bestimmt werden soll, und dient als Grundlage für viele weiterführende Aufgaben. Die Kombination aus spektraler, temporaler und räumlichen Dimensionen in SITS bietet den Vorteil, dass die enthaltenen Informationen aus den verschiedenen Dimensionen kombiniert werden können um die Klassifikationsergebnisse zu verbessern. Andererseits kann die Komplexität dieser Daten zu einer hohen Anzahl von Berechnungen führen, wenn die Daten bzw. Dimensionen gemeinsam verarbeitet werden. Eine Strategie um die Anzahl der Berechnungen zu verringern ist eine separate Extraktion von Informationen aus den einzelnen Datendimensionen. Eine solche Trennung wirft jedoch die Frage auf, welche Methodik in welcher Dimension am besten geeignet ist, wobei im Falle von SITS die spektrale, zeitliche und räumliche Dimension auftreten, und wie die berechneten Merkmale anschließend fusioniert werden sollen. Diese Fragen sind für die Aufgabe der multitemporalen LB Klassifikation mit SITS noch nicht im Detail beantwortet und analysiert worden. Eine weitere Strategie zur Reduzierung der Anzahl an Berechnungen basiert auf der Reduzierung der Anzahl der Eingabevektoren für ein Transformer-Modell. Hierfür werden mehrere Pixel in den Eingabebildern zu einem Patch zusammengefasst und dem Transformer-Modell präsentiert. Durch diese Strategie wird die Anzahl der Berechnungen in den nachfolgenden Schichten drastisch reduziert, allerdings verringert sie auch die geometrische Auflösung und somit die Fähigkeit des Modells, die genaue Position von Klassengrenzen und feinen Strukturen korrekt zu klassifizieren, wenn sie ein Gebiet abdecken, das nur durch wenige dargestellt wird.

In dieser Arbeit wird eine neue Methode für die multitemporale LB-Klassifikation vorgestellt. Das neue Modell basiert auf dem Swin Transformer, der von (Liu et al., 2021) für pixelweise Klassifikation eingeführt wurde. Dieser wird erweitert um multitemporale Eingangsdaten, in diesem Fall SITS, verarbeiten zu können und multitemporale LB-Karten zu prädizieren. Um räumliche und zeitliche Abhängigkeiten der SITS zu berücksichtigen, wird ein hybrides Modul zur Merkmalsextraktion eingeführt. Dieses Modul unterteilt die Berechnung räumlicher und zeitlicher Abhängigkeiten aus den Eingabebildern: Der räumliche Kontext wird durch Faltungen (convolutions) extrahiert, und zeitliche Abhängigkeiten durch "Selbstaufmerksamkeit" (self-attention) berücksichtigt. Diese Trennung reduziert die Anzahl der notwendigen Berechnungen im Vergleich zu einem Modul in dem räumlich-zeitliche Abhängigkeiten gemeinsam berücksichtigt werden, und ermöglicht die Nutzung der jeweils optimalen Methode in der jeweiligen Dimension. Mithilfe

von Selbstaufmerksamkeit können sowohl lokale als auch globale Abhängigkeiten berücksichtigt werden, was für die Modellierung saisonaler Effekte besonders geeignet ist. Im Gegensatz dazu berücksichtigen Faltungen die lokale Nachbarschaft, was sich besonders eignet um räumlicher Abhängigkeiten der Rasterdaten einzubeziehen. Um die Klassifikationsgenauigkeit weiter zu verbessern, wird außerdem ein neues Modul für die Generierung der Patches zu Beginn der Prozesierung eingeführt, um die negativen Auswirkungen der Patch Erstellung abzuschwächen. Dieses Modul verwendet 3D-Faltungen (3D convolutions), um räumlich-zeitliche Abhängigkeiten in der originalen räumlichen Auflösung der Eingabebilder zu extrahieren, bevor die räumliche Auflösung reduziert wird.

Die Evaluation der neuen Methode basiert auf zwei Datensätzen, dem Niedersachsen- und dem Dynamic Earth Datensatz, für LB-Klassifikation mit SITS. Die erreichten Genauigkeiten des Modells variierte zwischen den Datensätzen und erreichte mittlere F^1-Werte zwischen 73% und 79% für den Niedersachsen-Datensatz und 44% für den Dynamic Earth Datensatz. Für die meisten LB Klassen wurden gute Klassifikationsergebnisse erzielt, Herausforderungen gab es jedoch bei Klassen mit weniger Trainingsbeispielen oder ähnlichem Aussehen im Vergleich zu anderen LB-Klassen. Die Experimente zur Analyse des hybriden Moduls für die Merkmalsberechnung zeigen, dass es im Vergleich zu einem Modell, das räumliche und zeitliche Merkmale ausschließlich durch Selbstaufmerksamkeit berücksichtigt, signifikant bessere mittlere F^1-Werte erreicht. Insbesondere verbesserte die neue Methode ($V\text{-}FE_{hyb}$) den mittleren F^1-Wert um 2% auf dem Testdatensatz für Niedersachsen und um 4.2% auf dem Testdatensatz für Dynamic Earth im Vergleich zu einer Modell-variante die nur auf Selbstaufmerksamkeit basiert ($V\text{-}FE_{att}$). Zusätzlich erreicht das neue Modell leicht bessere Klassifikationsergebnisse im Vergleich mit einem Modell, das räumlich-zeitliche Merkmale gemeinsam und basierend auf Selbstaufmerksamkeit extrahiert, während es die Berechnungskomplexität erheblich reduziert. Die Experimente bezüglich des neuen Moduls zur Patch-Generierung mithilfe von 3D-Faltungen zeigen, dass dieses Modul zu ähnlichen mittlere F^1-Werten bei einem der Datensätze und zu besseren mittleren F^1-Werten bei dem zweiten Datensatz führt, im Vergleich zu einem Modell, das die standardmäßige Strategie zur Patch-Generierung verwendet. Am Ende wird die neue Methode mit anderen Modellen aus der Literatur verglichen. In diesem Vergleich schneidet das neue Modell besser ab als Modelle, die ausschließlich auf Faltungen basieren, was verdeutlicht, dass hybride Ansätze für die Klassifikation von SITS gut geeignet sind. Im Vergleich zu anderen hybriden Ansätzen, die ebenfalls Faltungen und Selbstaufmerksamkeit nutzen, schneidet das neue Modell ähnlich gut ab.

Nomenclature

Abbreviations

BN, LN	Batch normalisation, Layer normalisation
CNN, FCN	Convolutional Neural Network, Fully Convolutional Neural Network
DL, DNN	Deep Learning, Deep Neural Network
GELU	Gaussian Error Linear Unit
GSD	Ground Sampling Distance
IoU	Intersection over Union
LC	Land cover
MLP	Multilayer perceptron
MSA, W-MSA	Multi-head self-attention, window-based multi-head self-attention
OA	Overall accuracy
PG	Patch generation
PPM	Pyramid Pooling Module
ReLU, lReLU	Rectified Linear Unit, leaky Rectified Linear Unit
RS	Remote sensing
SGD, ADAM	Stochastic gradient descent, adaptive momentum
SITS	Satellite image time series
TE	Temporal position encoding
TW	Temporal weighting module

Symbols

Image data

x, X, X_0	Pixel vector, image timeseries, mono-temporal image
y, Y, Y_0	Class label for a single pixel, multi-temporal label map, mono-temporal label map
$\hat{Y}$, $\hat{Y}$, $\hat{Y}_0$	Predicted class label, multi-temp. predicted label map, mono-temp. predicted label map
$\mathbf{a}$	Predicted probability vector
W_0, H_0	Height, width of input image in pixel
B, T	Number of spectral bands and timesteps of input image timeseries

h, w, b, t	Indices for height, width, bands and timesteps
n_c, c	Number of classes, iterator for classes
$\mathbb{C} = \{C^1, ..., C^{n_c}\}$	Set of classes
μ	Mean
σ	Standard deviation

Deep Learning

$\mathcal{C}$	Classification network
Θ, θ	Set of parameters of a classifier, single parameter in Θ
$\mathcal{T}$, $n_{\mathcal{T}}$	Labelled dataset, number of available inputs in the dataset
$\mathcal{L}$	Loss function
n_e, e	Total number of training epochs, iterator for epochs
η_e, η_f	Learning rate in epoch e, factor for learning rate
ω_c	Class weight for class c
n_{es}	Number of epochs for early stopping
n_{it}	Number of iterations in one epoch
mb, n_{mb}	Numer of images in one minibatch, number of samples (pixels) in one minibatch
$\mathcal{K}$, k	Kernel matrix and kernel size of a convolutional layer
str	Stride of a convolutional layer
h_p	Window size of a pooling layer
f	Activation function

Multitemporal SITS classification

A	Features in image-like structure
Z	Features in sequence-like structure
s, S	Index of stage, total number of stages
E_s, D_s	Encoder and decoder part of stage s
l_s, L_s	Index and total number of consecutively applied attention modules in stage s
n_p, N_s	Iterator and total number of spatial patches for one timestep in stage s
C_{in}	Number of input feature maps to Swin model
P	Patch size of Transformer models
M	Window size
h, n_{h_s}	Index and number of self-attention heads in stage s
Q, K, V	Query, key and value matrix for attention computation
W	Projection matrix

Contents

1 Introduction **1**
1.1 Motivation and goals 1
1.2 Contributions 5
1.3 Thesis outline 6

2 Basics **7**
2.1 Deep Neural Networks 7
2.1.1 Multilayer Perceptron 7
2.1.2 Activation functions 8
2.1.3 Classification layer 9
2.1.4 Supervised training of DNNs 9
2.1.4.1 Loss function for classification 10
2.1.4.2 Optimisation 10
2.1.4.3 Parameter initialisation 11
2.1.4.4 Normalisation layers 12
2.1.4.5 Stopping criterion: 13
2.1.4.6 Regularisation 13
2.2 DNNs for pixel-wise classification 14
2.2.1 Fully Convolutional Neural Networks 14
2.2.1.1 Convolutional layers 15
2.2.1.2 Downsampling layers 16
2.2.1.3 Upsampling layer 17
2.2.1.4 Skip connections 17
2.2.1.5 FCN architectures 18
2.2.2 Transformers for pixel-wise classification 20
2.2.2.1 Patch generation 20
2.2.2.2 Positional encoding 21
2.2.2.3 The attention mechanism of Transformer blocks 22
2.2.2.4 Transformer architectures for pixel-wise classification 23

3 Related Work **27**
3.1 Deep learning for pixel-wise classification 27
3.2 Deep Learning for timeseries classification 29
3.3 Classification of remote sensing data 32
3.3.1 Pixel-wise classification in remote sensing 32

3.3.2 Classification of satellite image time series . . . 34
3.3.3 Temporal and spatial position encoding . . . 37
3.3.4 Computational complexity . . . 39
3.4 Training data for remote sensing applications . . . 41
3.5 Discussion . . . 42
3.5.1 Comparison to the most similar works . . . 45

4 A new method for SITS classification . . . 49
4.1 Prerequisites . . . 49
4.1.1 Generation of image timeseries . . . 49
4.2 Model architecture . . . 50
4.2.1 Overview . . . 51
4.2.2 Encoder . . . 51
4.2.3 Decoder: Multi-temporal UPer-Net . . . 53
4.2.4 Spatio-temporal patch generation . . . 54
4.2.5 Temporal position encoding . . . 56
4.2.6 Spatio-temporal dependencies . . . 57
4.2.6.1 The hybrid feature extraction module . . . 57
4.2.6.2 Attention-based embedding of spatial and temporal dependencies . 61
4.2.6.3 Joint consideration of spatio-temporal dependencies . . . 62
4.2.7 Temporal weighting in skip connections . . . 64
4.3 Training . . . 65

5 Experimental Setup . . . 69
5.1 Datasets . . . 69
5.1.1 Lower Saxony & Kleve datasets . . . 69
5.1.2 Dynamic Earth dataset . . . 73
5.1.3 Discussion . . . 75
5.2 Objectives and structure of the experiments . . . 77
5.2.1 General training setup and parameter studies . . . 77
5.2.2 Experiment set 1: Proposed method for multi-temporal LC classification . . . 80
5.2.3 Experiment set 2: Evaluation of the hybrid feature extraction module . . . 81
5.2.4 Experiment set 3: Evaluation of the convolutional patch generation . . . 82
5.2.5 Experiment set 4: Evaluation of the temporal weighting module . . . 84
5.2.6 Experiment set 5: Comparison to other works . . . 85
5.3 Evaluation . . . 87

6 Results and Discussion . . . 89
6.1 Evaluation of the proposed method for multi-temporal LC classification (Exp. 1) . . . 89
6.1.1 Classification performance . . . 89
6.1.1.1 Results on the Lower Saxony dataset . . . 90
6.1.1.2 Results on the Kleve dataset . . . 92
6.1.1.3 Results on the Dynamic Earth dataset . . . 95
6.1.2 Analysis of predicted land cover changes . . . 97

6.1.2.1 Results on the Lower Saxony dataset . . . 97
6.1.2.2 Results on the Dynamic Earth dataset . . . 99
6.1.3 Discussion . . . 103
6.2 Evaluation of the hybrid feature extraction module (Exp. 2) . . . 105
6.2.1 Results on the Lower Saxony and Kleve datasets . . . 105
6.2.2 Results on the Dynamic Earth dataset . . . 109
6.2.3 Discussion . . . 110
6.3 Evaluation of the convolutional patch generation (Exp. 3) . . . 113
6.3.1 Results on the Lower Saxony and Kleve datasets . . . 113
6.3.2 Results on the Dynamic Earth dataset . . . 117
6.3.3 Discussion . . . 120
6.4 Evaluation of the temporal weighting module (Exp. 4) . . . 122
6.4.1 Results on the Lower Saxony and Kleve datasets . . . 123
6.4.2 Results on the Dynamic Earth dataset . . . 124
6.4.3 Discussion . . . 124
6.5 Comparison to other methods (Exp. 5) . . . 125
6.5.1 Results on the Lower Saxony and Kleve datasets . . . 125
6.5.2 Results on the Dynamic Earth dataset . . . 127
6.5.3 Discussion . . . 129

7 Conclusions and Outlook . . . 131
7.1 Conclusion . . . 131
7.2 Outlook . . . 134

Bibliography . . . 141

Curriculum Vitae . . . 149

1 Introduction

1.1 Motivation and goals

Pixel-wise classification is the task of assigning a class label to each pixel in an image based on a given class structure. For remote sensing (RS) applications, the images to be classified are typically acquired by cameras onboard aircrafts or satellites and cover large areas on the Earth's surface. Those images are georeferenced and commonly consist of several spectral bands to better distinguish different objects on the ground. For the last decade, the analysis of such image data has not only been fast on the methodological side, but also regarding the availability of even larger amounts of image data. Examples are satellite missions such as Sentinel-2 of the European Space Agency (ESA) and Landsat-8 of the National Aeronautics and Space Administration (NASA), which provide satellite imagery with regular temporal intervals covering almost the entire surface of the Earth. These image sequences are also called satellite image time series (SITS). SITS not only allow for analysing the current state of the Earth's surface, but also for monitoring its development over time. Applications include land cover and crop classification, change detection, deforestation mapping, map updating and the analysis of urban development. Figure 1.1 shows two exemplary applications for pixel-wise classification scenarios based on satellite imagery. The first example shows a land cover (LC) map for a Sentinel-2 image with 10 m ground sampling distance (GSD), while the second example shows a change map based on two satellite images from the Planet Labs satellites. In this thesis, SITS are applied to the task of LC classification for different timesteps. In LC classification, the class labels that are assigned to each image pixel correspond to the physical materials on the Earth's surface, e.g. *Water* or *Forest*.

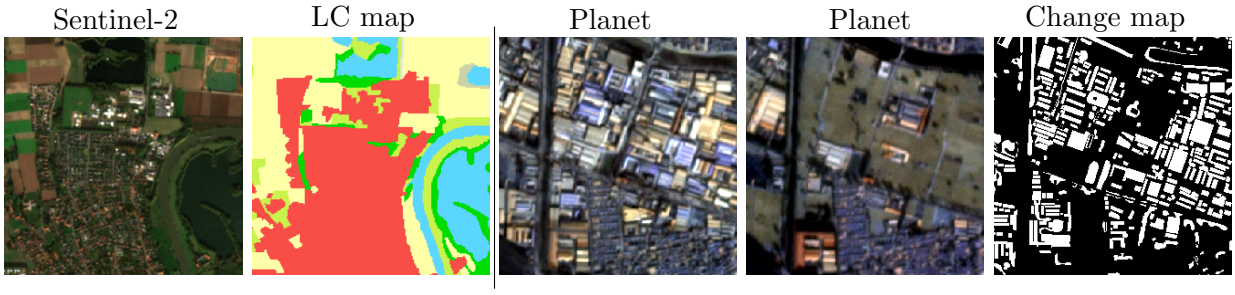

Figure 1.1: Examples for pixel-wise classification tasks based on satellite imagery. The left example shows a map for LC classification based on a Sentinel-2 image (blue: Water, red: Settlement, yellow: Agriculture, light green: Vegetation, dark green: Forest, brown: Barren land). The right example shows a change map (white areas indicate changes) based on the two shown Planet images.

Recent methods to predict pixel-wise class labels for image data are based on supervised deep learning (DL) models. In supervised DL, a classifier is trained based on training samples for which

the corresponding labels are known. In the case of pixel-wise image classification, the input to such a classifier is an image, and the output is a predicted label map of the same size. A training sample corresponds to one image pixel; during training, predictions for all pixels are used to compute the loss function and to update the weights of the classifier. First methods successfully applied to pixel-wise classification were Fully Convolutional Neural Networks (FCNs) (Long et al., 2015). They are based on convolutional layers which encode local spatial context from input images. These layers are used in combination with downsampling layers in the encoder and upsampling layers in the decoder. The main disadvantage of FCNs is the coarser spatial resolution in the deeper layers of a FCN that can result in difficulties to predict the correct position of class borders in the generated output.

More recent research for pixel-wise classification focuses on Transformer architectures that are based on self-attention and were initially introduced in the field of natural language processing NLP (Vaswani et al., 2017). The principle of self-attention allows to consider global and local dependencies based on all input feature vectors, independently of their order in the input sequence. After the success of these models in the field of NLP, they were adapted to image and pixel-wise classification. To obtain the input feature vectors needed for the self-attention layers, the grey values of the pixels from a patch of the input image are stacked and linearly projected to obtain a feature vector of a pre-defined dimension representing the patch (Dosovitskiy et al., 2021). Commonly, a hierarchical structure similar to the one in a FCN follows: First, self-attention and downsampling layers are combined in the encoder, whereas the subsequent decoder is based on convolutional and upsampling layers (Strudel et al., 2021; Liu et al., 2021).

Combining several pixels to a patch reduces the computational complexity in the self-attention layers and, therefore, also the training and inference time. However, using a large patch size for pixel-wise classification results in difficulties in classifying fine details or the correct position of class borders because larger regions in the input image are represented by one feature vector only (Strudel et al., 2021). For the pixel-wise classification of SITS, the impact of patch generation is even larger due to the coarser GSD of satellite imagery, e.g. 10 m for Sentinel-2, in comparison to aerial imagery. In this case, larger regions of the input image are represented by one input feature vector, e.g. 40×40 m when a typical patch size of 4×4 pixels is used for a Sentinel-2 image. On the other hand, a reduced patch size results in drastically longer training times and the need for more graphics processing unit (GPU) memory. The selection of a suitable patch size is, therefore, always a trade-off between computational costs and classification accuracy.

Several approaches try to mitigate the impact of the patch size, commonly by reducing the computational complexity within the self-attention layers. As a result, a smaller patch size can be used, which commonly results in an increase in the classification accuracy. One approach that is widely used to reduce the number of computations in a self-attention layer is the Swin Transformer (Liu et al., 2021), in which self-attention is only computed between patches within a defined window. This strategy is somewhat contrary to the general idea of computing global features directly within one self-attention operation, as, similar to convolutional layers, only a local neighbourhood is considered for calculating features. Liu et al. (2021) try to mitigate this effect by moving the windows in subsequent layers of the model to incorporate more global features

with increasing model depth. Other strategies compute global features by computing dependencies between different windows (Wang et al., 2021) or reduce the number of computations within the self-attention layer itself (Garnot and Landrieu, 2020; Zhang and Yan, 2023; Zaheer et al., 2020). However, the challenge that input patches are generated to reduce the total number of input feature vectors to be processed by a Transformer-based model still remains and leaves room for further research.

Many remote sensing approaches that were successfully introduced in the field of computer vision have been adapted to RS applications. From the methodological perspective, the classification of video sequences, e.g. for autonomous driving, is similar to the classification of SITS. Despite the methodological similarities to some computer vision applications, the special characteristics of SITS are the main challenge in creating a DL classifier for SITS. One characteristic is that the images of SITS are acquired at drastically longer temporal intervals and the usable image commonly have irregular temporal intervals due to cloud coverage. Another characteristic is the seasonally varying appearance for vegetation in combination with a more constant appearance of man-made objects. These properties indicate that self-attention is particularly suitable for the exploitation of temporal context and for applications such as LC or crop classification, because self-attention can consider global dependencies, i.e. from timesteps which are further apart, but also local ones, i.e. between neighbouring timesteps (Rußwurm and Körner, 2020; Voelsen et al., 2023; Zhang et al., 2023a). For instance, for the task of change detection, global temporal context might be very important to consider long-term temporal changes, while for the identification of different crop types, temporal dependencies between neighbouring timesteps are more important to model the seasonal changes due to growing and harvesting. In contrast, the regular grid and the local dependencies between pixels in the spatial dimensions of SITS suggest the use of convolutions for the extraction of features encoding local spatial dependencies in the context of pixel-wise LC classification. In literature, methods applied to pixel-wise classification of remote sensing images are based on convolutions (Yang et al., 2021; Pelletier et al., 2019) or on self-attention layers (Zhang et al., 2022b; Panboonyuen et al., 2021; Hanyu et al., 2024). However, models using self-attention to consider spatial context in feature extraction have to reduce computational complexity by generating patches at the beginning of the model and reducing the number of input patches for its self-attention layers, which both can negatively impact the models' performance. Therefore, it is still a research topic whether convolutional or self-attention layers are more suitable to consider spatial dependencies from SITS.

Besides the advantage that temporal information can be extracted from SITS, they also make it possible to generate multi-temporal output maps. For the application of LC, this means that not only the current or some representative state of LC can be predicted, but also the development of LC within the observed timespan. Existing approaches that classify SITS either focus on the generation of one output map only, e.g., for crop type classification (Garnot and Landrieu, 2021), or they are limited regarding the spatial dimensions of the input data, e.g. by only considering single pixels or a very small local neighbourhood (Rußwurm and Körner, 2020; Yuan et al., 2022; Zhao et al., 2023b). There are only a few exceptions of approaches that produce multi-temporal output maps. For instance, Stucker et al. (2023) adapt a U-Net by combining convolutional layers in the encoder and decoder stages with a self-attention module at the coarsest spatial resolution

and use SITS as input data. However, the task solved by the method is the removal of clouds, which is a regression instead of a classification task.

The approach presented in this thesis addresses the aforementioned research gaps for the application of LC classification with SITS. The first research question covered in this thesis asks which methodology is best suited to consider spatial context in SITS in order to exploit them in combination with temporal dependencies for LC classification. Another aspect related to this topic is the challenge of maintaining a reasonable computational complexity. For the purpose of answering the related research question, the application of convolutions is compared to the usage of self-attention to consider spatial context in the experiments conducted in this thesis.

The second research question addressed in this work is related to the patch generation process in the first layers of Transformer models that limits their classification performance, especially regarding object boundaries or input featuring fine structures. This challenge has an even higher relevance for the coarser GSD of satellite images when objects need to be classified that only cover a few patches or have an extent smaller than the patch size, e.g. streets. In this thesis, a new patch embedding module is introduced with the goal to minimise the effect of patch generation. Within the new module, 3D convolutions are used in combination with downsampling layers that reduce the spatial resolution in several steps. The reduction of the spatial dimension in several steps has the advantage that these features can also be integrated into the later layers in the decoder by using skip connections, which contrasts the standard patch embedding that essentially features a single downsampling step with a reduction factor of the patch size.

The methodology introduced in this thesis is evaluated for the application of multi-temporal LC classification. The information about LC serves as basis for topographic databases, which can be used for many follow-up tasks such as urban planning, navigation, or risk management. It is quintessential for these databases to be kept up-to-date, which is an expensive and time-consuming task that can be alleviated by the results of a classifier which predicts multi-temporal LC maps. The results of a classifier that predicts multi-temporal LC maps can help to keep such databases up-to-date, especially when freely available satellite images for multiple acquisition dates are available. On the other hand, the LC information already included, e.g. from the previous years, can be used to generate a large training dataset that is useful to train deep neural networks. Using the data already available naturally results in some incorrect class labels in areas where the LC has changed without these alterations having yet been integrated into the database. Wrong class labels are also called *label noise* in literature. Training with *label noise* is a larger research topic (Song et al., 2023) and not the focus of this thesis. However, as one of the used datasets is generated from a topographic database, this aspect impacts the training and evaluation processes. This dataset consists of Sentinel-2 SITS data, covering a timespan of four years with a GSD of 10 m that are combined with eight LC classes automatically generated from the ATKIS database (AdV, 2008). In addition, a second dataset, consisting of Planet images with 3 m GSD for 75 different areas spread around the whole Earth, is utilized. This dataset covers a time span of two years and labels for each month are generated in a manual labelling process (Toker et al., 2022).

1.2 Contributions

The goal of this thesis is to introduce a new method for multi-temporal LC classification that addresses the previously outlined challenges, specifically concerning the patch generation process and the integration of spatial and temporal context, while ensuring manageable computational complexity. These analyses are conducted to achieve a classification performance that makes it possible to use the generated LC maps for follow-up tasks, like the update of topographic maps.

SITS serve as input to the model, having four dimensions (time, spectral wavelength, height and width). The output of the model consists of one label map (referred to as *map* in the remainder of this thesis) for every image in the time series. The prediction of multiple LC maps over time makes it possible to predict different LC classes for the same area at different timesteps. The model itself is based on an encoder-decoder structure. In the encoder, first the new module for patch generation is used to extract features encoding spatio-temporal dependencies from the input data. This is followed by a combination of a new module to consider spatial and temporal dependencies separately by convolutions and self-attention, and downsampling layers to further reduce the spatial resolution. The decoder is based on UPer-Net (Xiao et al., 2018), consisting of convolutional and upsampling layers to obtain the original spatial resolution again. In addition, skip-connections are used to integrate features from the encoder into the decoder layers and, as a minor contribution, the temporal weighting module of Stucker et al. (2023) is modified and used to weight the spatial features by the temporal features at all model stages.

The scientific contribution of this thesis can be summarised as follows:

- A new model for pixel-wise classification of SITS is introduced. It takes a satellite image timeseries as input and predicts one class label for each pixel and timestep. The main contribution is a hybrid spatio-temporal feature extraction module that extracts features encoding spatial and temporal dependencies in separate streams. In the temporal stream, self-attention is used to consider local and global temporal context, while in the spatial stream, local spatial dependencies are considered by computing features using convolutional layers. The new model is evaluated in comparison to a model that jointly considers spatial-temporal context with self-attention and a model that encodes spatial and temporal dependencies separately by using self-attention in both streams.

- To further improve the classification performance, a new patch embedding module is introduced that is based on 3D convolutions to extract features encoding spatio-temporal dependencies at the original spatial resolution of the input images. It replaces the standard patch embedding module used in the Vision Transformer (ViT) (Dosovitskiy et al., 2021) and in the Swin Transformer (2021).

- As a minor contribution, the temporal weighting module used by Stucker et al. (2023) in the bottleneck layer of their model is extended to all spatial resolutions with the goal of capturing fine spatial details also within this weighting module.

- The proposed method is evaluated on two datasets for LC classification with SITS. Additionally, it is compared to other state-of-the-art methods, including a FCN and the *Utilise* model of Stucker et al. (2023).

In the context of this thesis the following research questions are formulated:

1. How does the proposed model, which incorporates temporal context, perform for the task of multi-temporal land cover classification? Is the model able to detect class changes over time, even if it is not explicitly trained for change detection?

2. Does the proposed hybrid feature extraction module, considering spatial context with convolutions and temporal context with self-attention, lead to better results than modules solely relying on self-attention for feature extraction?

3. Does the proposed patch embedding module, utilizing 3D convolutions to encode spatio-temporal dependencies, achieve better performance compared to standard patch embedding strategies such as the one in the Vision Transformer (Dosovitskiy et al., 2021) or when compared to using 2D convolutions to consider spatial context only?

4. Does an additional skip connection at a higher spatial resolution than the one of the feature maps after patch embedding lead to improved classification performance?

5. Does integrating the proposed temporal weighting module into the skip connections throughout all model stages enhance the classification performance compared to not using any temporal weighting?

6. Does a variant of the proposed methods outperform approaches from literature, specifically those using a multitemporal FCN (Voelsen et al., 2023), the Utilise model presented by Stucker et al. (2023) and a U-Net (Toker et al., 2022)?

With the goal to answer these questions, extensive experiments are conducted on two datasets for LC classification with SITS.

1.3 Thesis outline

The remainder of this thesis is structured as follows. Chapter 2 provides a brief introduction of the basics of the used Deep Learning concepts, the DL models that are used, and the training procedure. In chapter 3, an overview of the current state-of-the-art in pixel-wise and time series classification is presented, with a focus on approaches in the field of Remote Sensing. Chapter 4 introduces the new method, including a general model overview and a detailed description of the different proposed modules. Chapter 5 gives an overview about the datasets that are used for the experiments and introduces the different objectives of the experiments, corresponding to the stated research questions. This is followed by a presentation of the obtained experimental results in chapter 6. Finally, a summary of the results is given in chapter 7, followed by the conclusions with respect to the stated research questions and a discussion of open questions for future research.

2 Basics

In this chapter, the fundamental concepts that are relevant for this thesis are described. First, the basics about deep neural networks, pixel-wise classification and the training process are introduced in section 2.1. Afterwards, the basic components of DNN for pixel-wise classification are introduced in section 2.2.

2.1 Deep Neural Networks

Today, Deep Neural Networks DNNs represent the state-of-the-art for classification problems. In comparison to traditional machine learning methods, such as random forests, they do not require hand-designed features as input. Instead, the classifier learns which features are relevant to solve a specific task. In the end of a DNN, the extracted features are mapped to the output, e.g. different class scores in the case of classification problems (Goodfellow et al., 2016). In this thesis, DNNs are used for supervised classification tasks and therefore, only concepts relevant for these tasks are covered.

In this section, the main principles of DNNs, which are the basis for the architectures used for pixel-wise classification (cf. sections 2.2.1 and 2.2.2), are described. The section starts with the introduction of a multilayer perceptron (MLP) in section 2.1.1 and continues with the description of activation functions (cf. section 2.1.2) and the classification layer (cf. section 2.1.3). In the end, supervised training strategies for DNN are described in section 2.1.4.

2.1.1 Multilayer Perceptron

A MLP is a very simple feed-forward neural network. It defines a mapping $\mathbf{a} = g(\mathbf{u}, \Theta_{MLP})$ of an input $\mathbf{u}$ to an output $\mathbf{a}$ and learns the parameters Θ_{MLP}. A MLP consists of a number L of fully connected layers. Each layer $l \in \{1, ..., L\}$ consists of D^l nodes, with $n^l_{d^l}$ as the d^l-th node in layer l and $d^l \in \{1, ...D^l\}$. The input to layer l is the D^{l-1}-dimensional output vector of layer $l-1$, except for the first layer, for which the input is the D^0-dimensional input vector $\mathbf{u}$. The nodes form the fundamental element of an MLP and combine the outputs of all nodes from the previous layer $l-1$ to produce a new output. Each node $n^l_{d^l}$ is connected with all D^{l-1} nodes from the previous layer $l-1$. In each connection, the output from the node of the previous layer is multiplied by a weight $\theta^l_{d^l,k}$, with $k \in \{1, ..., D^{l-1}\}$ as the index of the node in layer $l-1$ and d^l as the index of the node in layer l. All weighted connections to node d^l are linearly combined and a bias $b^l_{d^l}$ is added, resulting

in the output o:

$$o_{d^l}^l = \sum_{k=1}^{D^{l-1}} \theta_{d^l,k}^l a_k^{l-1} + b_{d^l}^l \tag{2.1}$$

The outputs of all nodes in layer l are collected in a vector $\mathbf{o}^l = [o_1^l, ..., o_{D^l}^l]$ and then they are presented to an activation function f:

$$a_{d^l}^l = f(o_{d^l}^l) \tag{2.2}$$

The resultant activations are collected in a vector $\mathbf{a}^l = [a^l, ..., a_{D^l}^l]$ and serve as input to layer $l+1$. In case of the last layer L, the softmax function (cf. section 2.1.3) is used as activation function, and its output represents the model output $\mathbf{a} = \mathbf{a}^L$. Layers 1 to $L-1$ are called hidden layers, while layer L is the output layer. The weights $\theta_{d^l}^l$ and biases $b_{d^l}^l$ for all layers and all nodes from equation 2.1 form the set of parameters Θ_{MLP} of the complete MLP. A MLP is a simple DNN that can be used as a classifier $\mathcal{C}$. However, a classifier can also be composed of different layers types and the principles that are introduced in the following sections are not restricted to a specific layer type. Therefore, the trainable parameters of a classifier $\mathcal{C}$ are denoted with Θ in the following. The activation functions used in the hidden layers, also called non-linearities, are described in section 2.1.2, whereas the softmax function is presented in section 2.1.3.

2.1.2 Activation functions

Equation 2.1 determines in a linear combination of the input values. To model non-linear transformations of the input data, a non-linear activation function f is used to compute the final output of each layer (cf. equation 2.2). To decrease the computational complexity of DNNs, simple functions are the preferable choice. A challenge is the *vanishing gradient problem*, which means that gradients may become zero or near to zero for very small or very large input values, which gets problematic when multiple layers with gradients near zero follow each other.

The Rectified Lineas Unit (ReLU) was introduced by (Nair and Hinton, 2010) to counteract the *vanishing gradient problem* and is defined as:

$$f_{RL}(o) = \begin{cases} o, & \text{if } o > 0 \\ 0, & \text{otherwise.} \end{cases} \tag{2.3}$$

Using the ReLU activation function, the first derivative is 1 for values $o > 0$ and is zero for all values $o \leq 0$. This can lead to the effect that some nodes never contribute to the output computation because their output is always zero (*dead neurons*) and therefore, these weights are never updated during the training process. To counteract this problem a slightly different activation function called leaky Rectified Linear Unit (lReLU) was introduced by (Maas et al., 2013):

$$f_{lRL}(o) = \begin{cases} o, & \text{if } o > 0 \\ s \cdot o, & \text{otherwise.} \end{cases} \tag{2.4}$$

For values larger than zero ($o > 0$), the first derivative is 1 as for ReLU activations, but for values $o \leq 0$ the derivative is a scalar factor s that is a hyper-parameter. As a result, *dead neurons* are avoided.

Hendrycks and Gimpel (2016) introduce the Gaussian Error Linear Unit (GELU) activation function, integrating a more probabilistic view on the neuron's output. This is realised by multiplying the output o of a layer with the cumulative distribution function $\Phi(o)$ of the gaussian distribution, which results in:

$$f_{GL}(o) = o \cdot \Phi(o), \tag{2.5}$$

as the GELU activation function. Compared to ReLU, the GELU activation function is smoother, which Hendrycks and Gimpel (2016) found beneficial for the training process as it resulted in improved performance in their experiments.

2.1.3 Classification layer

In a multi-class classification problem, the initial input vector $\mathbf{u}$ is mapped to a class label y. The class label is based on the set of available classes $\mathbb{C} = \{C^1, ..., C^{n_c}\}$ with n_c as the total number of classes. To compute class scores for all n_c classes, the number of nodes D^L in the last layer n_L of a DNN is equal to the number of classes ($D^L = n_c$). The raw output $\mathbf{o}^L$ consists of unnormalised class-scores, which are commonly normalised by applying the softmax function as the activation function in the output layer (Bishop, 2006, p. 236):

$$a_c^L = \frac{exp(o_c^L)}{\sum_{i=1}^{n_c} exp(o_i^L)} \; \forall \; c \in \{1, ..., n_c\} \tag{2.6}$$

The values of $\mathbf{a} = \mathbf{a}^L = [a_1^L, ..., a_{n_c}^L]$ sum up to 1. An entry $a_c^L = P(y = C^c|\mathbf{u}, \Theta)$, with $c \in [1, ..., n_c]$, can be interpreted as the probability that the unknown class label y is the cth class label C^c, given the feature vector $\mathbf{u}$ and the network parameters Θ. Afterwards, $\mathbf{a}$ is used to compute the loss function during the training process (cf. section 2.1.4), while during inference the class with the highest posterior probability is chosen as the final class prediction $\hat{y} = argmax(\mathbf{a})$.

2.1.4 Supervised training of DNNs

In a supervised classification scenario, the parameters Θ of a classifier $\mathcal{C}$ are determined in the training process. The goal during training is to minimise the loss function $\mathcal{L}$, resulting in predictions that get closer to the training labels when the training process continues. The goal is that the classifier extracts meaningful features from the input $\mathbf{u}$ to generate the prediction $\mathbf{a} = \mathcal{C}(\Theta, \mathbf{u})$ which is then used to get the final class prediction $\hat{y}$ (cf. section 2.1.3). To determine the parameters Θ, a training dataset $\mathcal{T}_{tr} = \{\mathbf{u}_i, y_i\}_{i=1}^{n_\mathcal{T}}$ is given, in which for a total number $n_\mathcal{T}$ of inputs $\mathbf{u}_i$, the corresponding label y_i is given. During the training process, the parameters are updated in an iterative process by comparing the current class prediction of the model $\hat{y}$ with the given training labels y. in the following sections, the basic concepts about loss functions (section 2.1.4.1), optimisation strategies (section 2.1.4.2), parameter initialisation (section 2.1.4.3), normalisation (section 2.1.4.4) and regularisation (section 2.1.4.6) that are relevant in the context of this thesis are introduced.

2.1.4.1 Loss function for classification

The most frequently used loss function for multi-class classification is the cross-entropy loss:

$$\mathcal{L}_{CE}(\Theta) = -\frac{1}{n_s}\sum_{n=1}^{n_s}\sum_{c=1}^{n_c} O_n^c \cdot log(a_n^c), \tag{2.7}$$

where n_s is the total number of samples over which the loss is computed and $a_n^c = P(y_n = C^c|\mathbf{u}_n)$ is the predicted probability that the n-th sample belongs to class $y_n = C^c$, which is commonly computed using the softmax activation function (equation 2.6), and O_n^c is a binary indicator that equals 1 if sample $\mathbf{u}_n$ belongs to class C^c in the training dataset and equals 0 otherwise. Consequently, the cross-entropy tries to push the probability of the correct class to one.

2.1.4.2 Optimisation

In the context of DL, the model parameters Θ of a classifier $\mathcal{C}$ are determined by minimising a loss function $\mathcal{L}(\Theta, \mathcal{T})$ by using a given training dataset $\mathcal{T}$ (cf. section 2.1.4.1). As the parameter space is high-dimensional and the function g has a complex structure, it is not possible to simply compute the position of a minimum by setting the gradient $\nabla\mathcal{L}(\Theta, \mathcal{T})$ to zero. Instead, DNNs are trained based on gradient descent (GD). In GD, the parameters Θ of a classifier $\mathcal{C}$ are first randomly initialised, e.g. by drawing from a normal distribution. These initial parameters Θ^0 are iteratively updated according to the gradient vector by moving in the direction of the steepest descent $-\nabla\mathcal{L}(\Theta^i, \mathcal{T})$ in weight space in an iterative manner, with i as the index for the training iteration. The size of the step is controlled by the learning rate η, which is a defined hyper-parameter, resulting in the following update rule for the weights in iteration i:

$$\Theta^i = \Theta^{i-1} - \eta\nabla\mathcal{L}(\Theta^{i-1}, \mathcal{T}). \tag{2.8}$$

In DNNs the gradient vector $\nabla\mathcal{L}(\Theta^{i-1}, \mathcal{T})$ is computed by back propagation (Bishop, 2006, pp. 241 ff) and the loss function (cf. section 2.1.4.1) has to be differentiable with respect to each parameter in Θ.

Mini-batch stochastic gradient descent: In the standard GD, the loss is computed for the complete training dataset $\mathcal{T}$. As this is not feasible in practice, commonly mini-batch stochastic gradient descent (SGD) is used in the training process. In SGD, a small set of samples, referred to as a mini-batch, is used to compute the gradient $\nabla\mathcal{L}(\Theta, \mathcal{T}_{mb})$. The number of samples in one mini-batch n_{mb} is a hyper-parameter that has to be defined for the training process.

With SGD, one iteration i of the standard training process consists of several steps: First, a minibatch $\mathcal{T}_{mb}$ with n_{mb} samples is randomly chosen from the training dataset $\mathcal{T}$ and the output predictions $\mathbf{a}$ are computed for all samples in the minibatch. Afterwards, the loss is computed as the average loss over the n_{mb} samples (i.e. $n_s = n_{mb}$ in equation 2.7), and error backpropagation (Bishop, 2006, pp. 241 ff) is used to compute the gradient $\nabla\mathcal{L}(\Theta^i, \mathcal{T}^{mb})$, which is then used to update the weights for the next iteration.

The number of samples in a minibatch n_{mb} and the learning rate η are critical parameters for the training process. A small minibatch size results in fast computations but can also result in

non-representative samples in a specific minibatch, which might result in a gradient that does not point into the direction of the minimum for the whole dataset. For this reason, the minibatch size is commonly set as large as possible; it is usually limited by the memory available in the Graphics Processing Unit (GPU). The learning rate η also has a large impact on the training process. If the learning rate is too small, the model will converge slowly. On the other hand, a excessively large learning rate can result in a divergence or poor classification results during inference. Therefore, the optimal value for η is set during hyper-parameter tuning.

ADAM optimiser: Kingma and Ba (2015) introduced the **ada**ptive **m**omentum (ADAM) optimiser that extents SGD by computing adaptive learning rates for all model parameters from estimates of the first and second moments of the gradients. In each training iteration i, the moving averages of the gradients ($\dot{\mathbf{m}}^i$) and the squared gradients ($\ddot{\mathbf{m}}^i$) are updated. In the first iteration $i = 0$, the vectors of the first and second moments are initialised with zero, i.e. $\dot{\mathbf{m}}^0 = \vec{0}$ $\ddot{\mathbf{m}}^0 = \vec{0}$. Whereas $\dot{\mathbf{m}}^i$ is the estimate of the first moments (the mean) of the gradients, $\ddot{\mathbf{m}}^i$ is the estimate of the second moments of the gradient of $\mathcal{L}(\Theta^{i-1})$:

$$\dot{\mathbf{m}}^i = \beta_1 \cdot \dot{\mathbf{m}}^{i-1} + (1 - \beta_1)\nabla\mathcal{L}(\Theta^{i-1}) \tag{2.9}$$

$$\ddot{\mathbf{m}}^i = \beta_2 \cdot \ddot{\mathbf{m}}^{i-1} + (1 - \beta_2)\nabla\mathcal{L}(\Theta^{i-1})^2. \tag{2.10}$$

In 2.9 and 2.10, $()^2$ denotes the element-wise square. The hyper-parameters β_1 and β_2 control the exponential decay rates, typically set to $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The model parameters are updated based on the computed moments:

$$\Theta^i = \Theta^{i-1} - \eta^i \frac{\dot{\mathbf{m}}^i}{\sqrt{\ddot{\mathbf{m}}^i} + \epsilon}, \tag{2.11}$$

with

$$\eta^i = \eta \frac{\sqrt{1 - \beta_2^i}}{1 - \beta_1^i}. \tag{2.12}$$

The parameter ϵ is a very small constant used for numerical stability, and $\sqrt{\ddot{\mathbf{m}}^i}$ denotes the element-wise computation of the square root for all elements in $\ddot{\mathbf{m}}^i$. Compared to SGD, ADAM reduces problems with noisy gradients.

2.1.4.3 Parameter initialisation

The primary goal of parameter initialisation is to establish good starting values for the training process to achieve fast convergence. For a layer in a DNN, it is important to have different starting values for all nodes as they all shall extract different features from the input data. Commonly, a random initialisation is used because this is computationally cheap and it is very unlikely that the same initial values are assigned to two nodes in a layer (Goodfellow et al., 2016, pp.292 -295). A Gaussian distribution $\mathcal{N}$ with mean $\mu = 0$ and standard deviation σ is widely used for that purpose. Several strategies exist to select σ. He et al. (2015) introduce a strategy that works well with ReLU activations. The standard deviation σ^l in layer l is set to $\sigma^l = \sqrt{2/n^l}$ with n^l as the number of connections for one node in layer l to layer $l-1$ (for a MLP this is equal to the number of nodes in layer $l-1$). This strategy is used for the weights θ, while the biases b are all initialised by zero and is referred to as *variance scaling*.

2.1.4.4 Normalisation layers

Batch normalisation: During the training process, the model parameters Θ are updated using a small subset of the training data which is randomly chosen from the training dataset and called a minibatch. This results in variations of the input feature distributions in all layers of the model, leading to a slower training process because the parameters must adapt to the individual input distributions in each training iteration. As a consequence, more training steps with smaller learning rates are needed until the model converges (Ioffe and Szegedy, 2015). Ioffe and Szegedy (2015) call this phenomenon *internal covariate shift* and introduce a normalisation strategy called batch normalisation (BN).

In BN, the outputs of a layer are normalised to have zero mean and unit variance. The normalisation is applied to the raw outputs of a layer before the activation function is applied (2.1 for a MLP) (Ioffe and Szegedy, 2015). BN is applied per dimension $d^l \in \{1, ..., D^l\}$, i.e. for each output of a node o_{d^l} (for simplicity the layer index l is omitted here), the empirical mean μ_{d^l} and empirical variance $\sigma^2_{d^l}$ are computed over all samples in the current minibatch by:

$$\mu_{d^l} = \frac{1}{n_{mb}} \sum_{i=1}^{n_{mb}} o_{i,d^l}, \tag{2.13}$$

$$\sigma^2_{d^l} = \frac{1}{n_{mb}} \sum_{i=1}^{n_{mb}} (o_{i,d^l} - \mu_{d^l})^2 \tag{2.14}$$

n_{mb} denotes the number of samples in the minibatch. Using μ^l_d and $\sigma^2_{d^l}$, the normalised raw output o^{bn}_{i,d^l} is computed by:

$$o^{bn}_{i,d^l} = \gamma^{d^l} \cdot \frac{o_{i,d^l} - \mu_{d^l}}{\sqrt{\sigma^2_{d^l} + \epsilon}} + \beta^{d^l}. \tag{2.15}$$

The parameters γ^{d^l} and β^{d^l} are learnable parameters that are used to scale and shift the normalised activation, respectively, and ϵ denotes a small constant scalar for numerical stability. During training, the running averages of the means and variances are determined. These values are to be used to compute the normalised output values (eq. 2.15) during inference.

Layer normalisation: Similar to BN, Layer normalisation (LN) is used to normalise the outputs of a layer of a DNN to have zero mean and unit variance. The main difference is that LN is independent of the minibatch size, as the mean and variance are computed over the feature dimension instead of per dimension over the current minibatch. Ba et al. (2016) found that beneficial as LN can be applied to minibatch sizes of one, and the computations are the same during training and inference.

To apply LN to the raw outputs o_{d^l} (for simplicity the layer index l is omitted here), the empirical mean μ_{LN} and the empirical variance σ^2_{LN} are computed over all feature values $d^l \in \{1, ..., D^l\}$ of

$\mathbf{o} = [o_1, ..., o_{D^l}]$:

$$\mu_{LN} = \frac{1}{D^l} \sum_{d^l=1}^{D^l} o_{d^l}, \tag{2.16}$$

$$\sigma^2_{LN} = \frac{1}{D^l} \sum_{d^l=1}^{D^l} (o_{d^l} - \mu_{LN})^2. \tag{2.17}$$

Using μ_{LN} and σ^2_{LN}, the normalised raw output $o^{LN}_{d^l}$ is computed by:

$$o^{LN}_{d^l} = \gamma \cdot \frac{o_{d^l} - \mu_{LN}}{\sqrt{\sigma^2_{LN} + \epsilon}} + \beta. \tag{2.18}$$

ϵ denotes a small constant scalar for numerical stability, and the parameters γ and β are learnable parameters that are used to scale and shift the normalised features, respectively. $o^{LN}_{d^l}$ denotes the d^l-th normalised feature value which are combined to $\mathbf{o}^{LN} = [o^{LN}_1, ..., o^{LN}_{D^l}]$.

2.1.4.5 Stopping criterion:

Training is continued for a pre-defined number of iterations, where in each iteration a training minibatch is used to update the model's weights. In general, a model tends to overfit to the training data if the training continues for too long, but on the other hand, it might underfit if the training ends to early. To define the perfect time to end a training process is therefore a difficult task that depends on the given dataset and on the model that is used. A commonly used strategy to determine the end of the training phase is the usage of the validation dataset. After a defined number of iterations, which is referred to as one epoch, the model's performance on the validation dataset is computed. The performance is tracked as training continues and in the end, the weights from the epoch with the highest score on the validation dataset is chosen. The total number of epochs can be set to a defined value, normally based on experience. Another option is to keep track of the validation performance and stop the training process if the validation accuracy does not increase for a given number of epochs. This strategy is called *early stopping*.

2.1.4.6 Regularisation

The weights Θ of a classifier $\mathcal{C}$ are adjusted based on the training dataset $\mathcal{T}$ by minimising the loss function $\mathcal{L}$. Afterwards, the model is applied to new data that has not been observed in the training process. The overall goal of training is a good performance of the model on the new data, which are also referred to as test data. The ability of the model to perform well on the test data is called its generalisation ability. As the test data is not observed in training, the goal during training is to minimise the loss function based on the data from training. The loss value achieved in training is also referred to as the training error. After training, this value can also be computed for the test data, in which case it is referred to as the test error. A good generalisation performance is achieved if the gap between the training and test error is as small as possible (Goodfellow et al., 2016, pp.107-108). Furthermore, commonly, a third dataset, referred to as the validation dataset, is used in the training phase. The validation dataset is used to set hyper-parameters or can be used

to stop the training process, which is based on the validation error that is computed in the same way as the test error but on the validation dataset. To achieve good classification performance, it is crucial to find a balance between underfitting and overfitting. Underfitting occurs when a classifier is too simple to capture the underlying patterns of the data, which leads to poor performance on the training and test dataset. Overfitting occurs when the model is too complex and starts to capture noise and other detailed patterns that only occur in the training data. Therefore, overfitting results in a good performance on the training dataset, while the performance on the test dataset is poor. Techniques to improve the generalisation capability are referred to as regularisation techniques and include dropout, early stopping and data augmentation. These methods will be described in the following.

Dropout: Srivastava et al. (2014) proposed a strategy called dropout to prevent a model from overfitting. When dropout is used in a layer of a DNN, the activations of a node are set to zero with a given probability p, which is called dropout probability. In principle, dropout can be used in any layer of a DNN except of the output layer. The idea of dropout is to implicitly learn an ensemble of networks that are jointly used during inference. During inference, no dropout is used anymore and all nodes contribute to the output prediction. To get output values within the same value range as in training, the outputs of all nodes are multiplied by the dropout probability instead.

Data augmentation: Better generalisation performance can usually be achieved by using a larger dataset, but for most applications there is just a limited number of labelled training samples. In data augmentation, additional samples are generated synthetically by randomly transforming input samples in a reasonable way, which means that the samples still need to be representative of the given task. For the task of pixel-wise image classification, typical augmentation strategies include image rotations, scaling, brightness, geometric, radiometric or contrast adjustments.

2.2 DNNs for pixel-wise classification

While MLPs the input and output of each layer are one-dimensional vectors, the input and output of DNNs for pixel-wise classification are three-dimensional tensors. For the pixel-wise classification of images, the input is an image $X_0 \in \mathbb{R}^{B \times H_0 \times W_0}$ with B as the number of spectral bands and H_0, W_0 as the image height and width. The output of the model is a map $A \in \mathbb{R}^{n_c \times H_0 \times W_0}$, with n_c as the number of classes, that contains pixel-wise probability class scores $\mathbf{a}_{h,w}$ with $h \in \{1, ..., H_0\}$ and $w \in \{1, ..., W_0\}$ (cf. section 2.1.3). The final predicted label map $\hat{Y}_0 \in \mathbb{R}^{H_0 \times W_0}$ is derived by taking the index with the highest class score for each pixel $\hat{y}_{h,w} = argmax(\mathbf{a}_{h,w})$. Similar to the notation in section 2.1.3, the class label y is based on a set of predefined classes $\mathbb{C} = \{C^1, ..., C^{n_c}\}$. FCNs and Transformer architectures are two types of DNNs that are commonly used for pixel-wise classification and will be described in the following sections 2.2.1 and 2.2.2.

2.2.1 Fully Convolutional Neural Networks

FCNs were introduced by Long et al. (2015) for the task of pixel-wise image classification. Most FCNs follow an encoder-decoder structure. While in the encoder, features are extracted by convolu-

tional layers and the spatial resolution is reduced, in the decoder, convolutional layers are combined with upsampling operations. The main components of FCNs will be described in the sections 2.2.1.1 - 2.2.1.4 and two commonly used FCN architectures are described in section 2.2.1.5.

2.2.1.1 Convolutional layers

Convolutional layers are the main component of a FCN and are based on the mathematical concept of a convolution. The output activation map A^{out} is computed based on the input A^{in}, a set of C^{out} kernels $\{\mathcal{K}_1, ..., \mathcal{K}_{C^{out}}\}$ and bias parameters b_q for each kernel $\mathcal{K}_q$, with $\mathcal{K}_q$ and b_q being the parameters determined in training. A kernel $\mathcal{K}$ is a tensor with (D+1) dimensions that is shifted in D dimensions of the input image or feature map, computing one output value at each position. It is typical to use D-dimensional hyper cubes, e.g. 3×3 pixel for a 2D kernel, while the last dimension is always equal to the number of input features. This operation results in a D-dimensional output feature map when one kernel $\mathcal{K}$ is applied. Most common is the 2D convolution, obtained by shifting the kernel in the two spatial dimensions of the input feature maps. The 2D convolution is explained in the following, before several modifications of this basic variant are explained.

2D convolution: In a 2D convolution, the kernels are shifted in two dimensions of the input activation map $A^{in} \in \mathbb{R}$. This operation is often used for image data, shifting the kernels in the two spatial dimensions. For a 2D-convolution, the output activation value $a^{out}_{h,w,q}$ at position (h, w) for the q-th kernel, i.e. channel q in the output activation map A^{out}, is computed based on the following equation:

$$a^{out}_{h,w,q} = f\left(b_q + \sum_{q^{in}=0}^{C^{in}-1} \sum_{i_h=0}^{h_k-1} \sum_{i_w=0}^{w_k-1} k_{q,i_h,i_w,q^{in}} \cdot a^{in}_{(h+d_h-i_h),(w+d_w-i_w),q^{in}}\right) , \tag{2.19}$$

where $k_{q,i_h,i_w,q^{in}}$ refers to the weight in row i_h, coloumn i_w and channel q^{in} of the q-th kernel $\mathcal{K}_q$; h_k and w_k are the height and width of kernel $\mathcal{K}_q$, respectively, and C^{in} is the number of input feature maps of A^{in}. b_q is the bias and $a^{in}_{(h+d_h-i_h),(w+d_w-i_w),q^{in}}$ refers to the input value at position $(h + d_h - i_h, w + d_w - i_w)$ and channel q^{in} in the input map A^{in}. The parameters $d_h = h_k - 1$ and $d_w = w_k - 1$ take into account that the indices (i_h, i_w) of the kernel $\mathcal{K}_q$ do not refer to the spatial center of $\mathcal{K}_q$. The activation function is indicated by $f()$.

Padding: When a convolution is applied to the input activation map A^{in}, the size of the generated output map A^{out} is smaller than the input by $(H^{out} = H^{in} - h_k + 1, W^{out} = W^{in} - w_k + 1)$. The original size can be preserved by adding rows and columns at the borders of the input activation map accordingly. There are several types of padding that define how the new pixel values are generated. One common strategy is zero-padding, where the additional pixels are filled with zeros. Another common strategy is reflection-padding, where the values of the padding lines are filled by reflecting the input values at the original edges of the input feature map, i.e. a 1D array of $[2, 4, 6, 7]$ would result in $[4, 2, 4, 6, 7, 6]$ if reflection-padding with one line on both sides is applied.

3D convolution: The concept of moving a kernel in the convolutional operation can be extended to three dimensions, which is then referred to as a 3D convolution. This can be used, for instance,

to shift the kernel in the spatial and temporal dimension of an input timeseries. Similar to the computation of 2D convolutions, it is assumed that the input activation map A^{in} has one dimension more than the number of dimensions in which the kernel is shifted. The computation of the output activation value $a^{out}_{h,w,d,q}$ at position (h, w, d) of the q-th channel in the output activation map A^{out} is then based on the following computation:

$$a^{out}_{h,w,d,q} = f\left(b_q + \sum_{q^{in}=0}^{C^{in}-1} \sum_{i_h=0}^{h_k-1} \sum_{i_w=0}^{w_k-1} \sum_{i_d=0}^{d_k-1} k_{q,i_h,i_w,i_d,q^{in}} \cdot a^{in}_{(h+d_h-i_h),(w+d_w-i_w),(d+d_d-i_d)q^{in}}\right) , \quad (2.20)$$

where $k_{q,i_h,i_w,i_d,q^{in}}$ refers to the weight in row i_h, coloumn i_w, depth i_d and channel q^{in} of the q-th kernel $\mathcal{K}_q$; h_k, w_k and d_k are the height, width and depth of kernel $\mathcal{K}_q$, respectively, and C_{in} is the number of input feature maps of A^{in}. b_q is the bias and $a^{in}_{(h+d_h-i_h),(w+d_w-i_w),(d+d_d-i_d),q^{in}}$ refers to the activation value at position $(h + d_h - i_h, w + d_w - i_w, d + d_d - i_d)$ and feature map q^{in} in the input activation map A^{in}. Again, the offsets $d_h = h_k - 1, d_w = w_k - 1$ and $d_d = d_k - 1$ take into account that the indices (i_h, i_w, i_d) of the kernel do not refer to the spatial centre of $\mathcal{K}_q$. The size of the kernel matrices can be different in height, width, and, in the case of 3D convolutions, depth. However, in this thesis, only kernels with the same size in all dimensions are used, and therefore the parameter k is referred to as the kernel size in all dimensions, i.e. $h_k = w_k = d_k = k$.

Batch normalisation for convolutional layers: In BN the raw outputs of a layer are normalised to have zero mean and unit variance as introduced in section 2.1.4.4. When BN is applied to convolutional layers, the normalisation is applied per feature map, i.e. the empirical mean and variance are computed based on one channel of the output for all input images in the current minibatch in case of 2D convolutions.

2.2.1.2 Downsampling layers

FCN architectures are commonly based on a hierarchical encoder-decoder structure, in which the spatial resolution is reduced in the encoder and enlarged again in the decoder. This has the advantage that the number of pixels is reduced in the deeper layers of the model and more features can be computed with equal computational costs, and the receptive field gets larger. Commonly applied strategies to reduce the spatial resolution are strided convolutions and pooling layers, which are both described in the following.

Strided convolution: In the standard convolution, the kernel $\mathcal{K}$ is shifted pixel by pixel over the input feature map and at each position, one output value is computed. The step size between subsequent computations of output values is referred to as the stride str of the convolution, i.e. $str = 1$ pixels for the standard case. In the end, for a patch of $str \times str$ pixels, one output value is computed. For stride values larger than one, the resolution is reduced approximately by the factor of str. In general, the stride can also be different for height and width. However, in this thesis, the stride will always be identical for both dimensions, and therefore the parameter str refers to the stride of a convolution in both dimensions.

Pooling layers: In a pooling layer, a window with a defined size $h_p \times h_p$ is shifted in the two spatial dimensions of the input activation map in the same way a convolutional kernel is shifted

using a stride of $str = h_p$. In this way, pooling is applied to non-overlapping image windows. In general, the window size can be different in height and width, but, in the context of this thesis, only square windows are used. For each region, a single output value is computed based on a pooling function by using all pixels inside the current window. Common pooling functions are maximum-pooling, i.e. the output value is determined as the maximum value of all pixels in the window, or average pooling, which means that the output is defined to be the mean value of all input values. With this operation, the spatial dimensions are reduced approximately by a factor of h_p.

2.2.1.3 Upsampling layer

The decoder of a FCN consists of convolutional layer combined with upsampling layers to obtain the original spatial resolution again. Common strategies to upsample include different interpolation strategies and transposed convolutions, which are both described in the following.

Interpolation: Interpolation is a straightforward and easy strategy to increase the spatial resolution of a feature map and is, e.g., used in (Ronneberger et al., 2015). Interpolation strategies include linear, bilinear or nearest-neighbour interpolation. The only needed parameter is the factor by which the spatial resolution shall be increased; a commonly used value is a factor of two. Interpolation layers do not have any trainable parameters, but their usage may lead to imprecise object boundaries in the output predictions.

Transposed convolutions: In a transposed convolutional layer, the upsampling process is learned (Long et al., 2015). The process can be interpreted as follows: First, a number of rows n_r after each row in the input feature map, and a number of columns n_c after each column is inserted in the input feature map A^{in}. The new rows and columns are filled with zeros. This operation results in an upsampled version A'^{in} with n_r times more rows and n_c times more columns than the original input feature maps A^{in}. Thus, the numbers n_r and n_c control how much the resolution is increased, i.e. $n_r = n_c = 1$ results in an upsampling factor of approximately two. After A'^{in} is created, it serves as input to a standard convolutional layer with a defined kernel size k. The weights in the kernel are learned, resulting in learning how to combine the input features and fill the gaps for the larger output feature map. The kernel size k also impacts the size of the output feature maps and to compensate for this, padding is commonly applied to obtain integer upsampling factors. In practice, the transposed convolution is optimised through direct kernel application on the input feature map, imitating the insertion of zeros to minimise the number of necessary computations.

2.2.1.4 Skip connections

One challenge of a FCN are correct class predictions for pixels near object or class boundaries, caused by the loss of information about their exact position during the downsampling process. A strategy to improve the classification for border pixels is the usage of skip connections, which are used in most FCN architectures. Skip connections connect corresponding layers of the encoder and decoder that have the same spatial resolution. In this way, information from earlier layers in the encoder with higher spatial resolutions serves as additional input to the corresponding decoder layers. There are several strategies to combine the features of the encoder with those of the

decoder. For instance, in the U-Net architecture (Ronneberger et al., 2015) the features from the encoder and decoder are concatenated, resulting in a larger number C_{skip} of feature maps with $C_{skip} = C_{enc} + C_{dec}$, where C_{enc} and C_{dec} represent the numbers of feature maps from the encoder and decoder, respectively. In other architectures, e.g. UPer-Net (Xiao et al., 2018), the encoder and decoder features are combined by element-wise addition, with the limitation that the number of feature maps from the corresponding encoder and decoder layers has to be identical.

2.2.1.5 FCN architectures

In this section, two commonly used FCN architectures are introduced. The first one is the U-Net architecture, introduced by (Ronneberger et al., 2015). Afterwards, UPer-Net, an adaptation of U-Net introduced by Xiao et al. (2018), is described.

U-Net: U-Net, introduced by Ronneberger et al. (2015) for biomedical image segmentation, is a FCN architecture based on the encoder-decoder concept. It consists of a number S of stages, with one stage referring to all layers that share the same spatial resolution. To distinguish the encoder and decoder parts of the model, E_s refers to stage s in the encoder and D_s refers to the corresponding stage in the decoder of the model, with $s \in \{1, ..., S\}$. Note that the decoder stages are counted backwards to have the same index s as the encoder stage with the same spatial resolution. In an encoder stage E_s, two convolutional layers with kernel size $k = 3$ and zero padding are applied, each followed by ReLU activations. Afterwards, except for the last stage E_S, maximum-pooling layers with a window size and stride of $h_p = 2$ are applied to reduce the spatial dimensions by a factor of two. Each time the spatial resolution is reduced, the number of feature maps is doubled, resulting in $C_s = C_{in} \cdot 2^{s-1}$ feature maps in stage s with C_{in} denoting the number of feature maps in the first encoder stage. A decoder stage D_s consists of three steps: First, the spatial resolution is upsampled by a factor of two using bilinear interpolation. Afterwards, the feature maps of the decoder are concatenated with those from the corresponding encoder stage E_s. Two convolutional layers with $k = 3$ and zero passing are applied, each followed by BN and ReLU activation. In the first of these convolutions, the number of feature maps is reduced by a factor of two. As the decoder stages start with the upsampling operation, the first decoder stage is D_{S-1}. After the decoder stage D_1, a convolutional layer with $k = 1$ converts the feature maps to raw class scores, which are then normalised by the softmax function. Table 2.1 gives an overview of the layers that are applied in the different stages. It has to be mentioned that this setup is not the original U-Net proposed in (Ronneberger et al., 2015) because several small adaptations are included, e.g. the usage of BN or padding.

UPer-Net: UPer-Net, introduced by Xiao et al. (2018), is an encoder-decoder architecture designed for multi-task learning. In the context of this thesis, only the task of pixel-wise classification is considered, and the other output heads are neglected. UPer-Net is proposed as a decoder architecture that can be combined with any hierarchical encoder. Xiao et al. (2018) use the ResNet architecture (He et al., 2016) as the encoder. An example for a decoder is given in figure 2.1.

The input feature maps to UPer-Net are the feature maps A_1 - A_S generated by the corresponding encoder layers E_1 - E_S. The input feature map A_s has a shape of $C_s \times H_s \times W_s$, with C_s indicating

Stage	Layer type	Depth
$E_1 - E_S$	2×(conv(3), BN, ReLU)	$C_s = 2 \cdot C_{s-1}$
	maxpool(2)	C_s
$D_{S-1} - D_1$	Up(2)	C_{s+1}
	concat(E_s, D_s)	$2 \cdot C_{s+1}$
	2×(conv(3), BN, ReLU)	$C_s = C_{s+1}/2$
Classification	conv(1), softmax	n_c

Table 2.1: Overview of the different stages of the U-Net architecture. conv(k): 2D convolution with kernel size k, maxpool(h_p): maximum pooling with window size and stride h_p, concat(): concatenation of feature maps from corresponding encoder and decoder stages (skip connections), Up(f): up-sampling by factor f using bi-linear interpolation, $C_s = C_{in} \cdot 2^{s-1}$ denotes the number of feature maps in stage s and C_{in} the number of input feature maps.

the number of feature maps in encoder stage E_s, and (H_s, W_s) the height and width of the feature map, respectively. UPer-Net consists of stages D_S - D_1, counted from S to 1, to have the same stage index s in encoder and decoder stages with the same spatial resolutions. In D_S, first a Pyramid Pooling Module (PPM) (Zhao et al., 2017b) is applied to compute hierarchical global representations of the input image. Inside the PPM module, four different average pooling layers, i.e. pyramid scales, are applied separately to the input feature maps. In the first pyramid layer, global average pooling is applied, i.e. the pooling window size is set to $h_p = H_S$ and one output value is computed for each input feature map. In the other three average pooling layers, windows of size $h_p = H_S/2$, $h_p = H_S/3$ and $h_p = H_S/6$ are used. Afterwards, a convolutional layer with $k = 1$ and C_{dec} output feature maps, followed by BN and ReLU, is applied to each of the four pyramid levels separately. Afterwards, upsampling layers based on bilinear interpolation are applied to increase the spatial resolution again to $H_S \times W_S$ for all pyramid levels. In the end, the outputs of all four pyramid levels and the input features A_S are concatenated, which results in a feature dimension of $C_S + 4 \cdot C_{dec}$. After the PPM module, a convolutional layer with $k = 3$ and C_{dec} output dimensions is applied, followed by BN and ReLU activation. The stages D_{S-1} - D_1 only consist of a convolutional layer with $k = 3$, followed by BN and ReLU activation.

Between the different stages, skip connections are used. For the skip connection before stage D_{S-1}, the feature maps of encoder stage E_{S-1} are combined with the upsampled features from decoder stage D_S. For all following skip connections, the features of the corresponding encoder stage E_s are combined with the output of the previous skip connection, cf. figure 2.1. The features are combined by element-wise addition and, to obtain the same feature dimension, the features coming from the encoder stages are transformed to C_{dec} feature maps by a convolutional layer with $k = 1$, followed by BN and ReLU, before using them in the skip connection. The features from the previous skip connection are upsampled by a factor of two with bilinear interpolation to match the spatial resolution of the features of stage E_s. After the feature maps are combined by element-wise addition, they serve as the input to the next stage D_s as well as to the following skip connection.

To generate the final output of the model (Classification block in figure 2.1), the feature maps of all stages are combined. To be able to combine them, the feature maps from all decoder stages

are first upsampled by bilinear interpolation by a factor 2^{s-1}. Afterwards, the feature maps from all stages are concatenated along the feature dimension, resulting in $S \cdot C_{dec}$ feature maps, which are then processed by a convolutional layer with $k = 3$, BN and ReLU activation, to obtain an output dimension of C_{dec}. Finally, another convolutional layer with $k = 1$, followed by BN and ReLU activation, maps the C_{dec} feature maps to n_c raw class score maps. In the model proposed in (Xiao et al., 2018), the spatial dimension in stage D_1 is already downsampled by a factor of 4, i.e. $H_1, W_1 = (H_0/4, W_0/4)$. Therefore, an upsampling layer based on bilinear interpolation with a factor of 4 is applied before the raw class scores are normalised by a softmax layer.

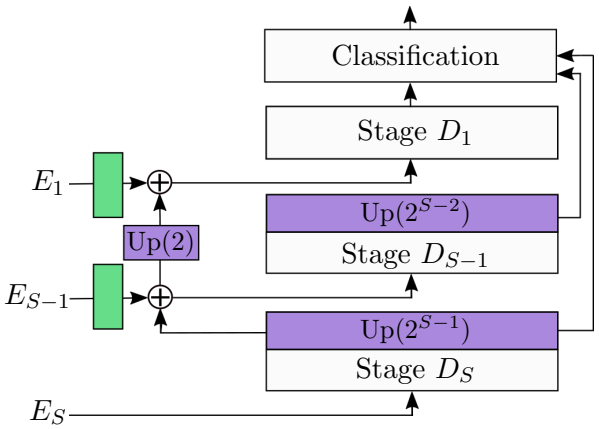

Figure 2.1: Overview of UPer-Net with three stages ($S = 3$). Up(f): Upsampling by a factor f with bilinear interpolation, $\oplus$: element-wise addition of the input feature maps. Green boxes indicate a convolution with $k = 1$, followed by BN an ReLU activation.

2.2.2 Transformers for pixel-wise classification

In this section, the concepts of models based on attention for pixel-wise classification are introduced. First, the patch generation process is described in section 2.2.2.1, followed by the introduction of the positional encoding in section 2.2.2.2. Afterwards, the main concept of attention, as used in the Transformer model (Vaswani et al., 2017), is described in section 2.2.2.3. In the end, two Transformer-based methods for pixel-wise classification are described in section 2.2.2.4.

2.2.2.1 Patch generation

To be able to apply self-attention as described in section 2.2.2.3 to an input image $X_0 \in \mathbb{R}^{B \times H_0 \times W_0}$, with B, H_0 and W_0 indicating the number of spectral bands, height and width of the input image, respectively, the image needs to be transformed into a sequence $Z_{emb} = \{\mathbf{z}^1, ..., \mathbf{z}^{n_{seq}}\}$, with C_{in} as the feature dimension of $\mathbf{z}^{i_{seq}}$.

For pixel-wise classification, a straightforward way to create this sequence is to take every single pixel as one element of dimension B and linearly project it to C_{in} dimensions. The number of pixels is referred to as $N_0 = H_0 \cdot W_0$. In practice, the computational costs would be comparably high for a self-attention layer with complexity $\mathcal{O}(n_{seq}^2)$, i.e., $\mathcal{O}((H_0 \cdot W_0)^2)$ if all pixels are used as an input feature vector. As a consequence, patch generation (PG) is applied before the first self-attention layer to reduce the sequence length. Note that PG is also referred to as patch embedding

in literature, but for a clear notation in comparison to positional encodings (cf. section 2.2.2.2), PG is used in the context of this thesis. In PG, multiple input pixels are combined and linearly projected to one vector of C_{in} elements. For this purpose, the image is split in the spatial dimensions into non-overlapping patches of size $P \times P$, with P as the patch size, and these patches are used to generate 1D vectors of dimension $P^2 \cdot B$ by stacking the vectors containing the spectral band values for every pixel in the patch on top of each other. Each of these 1D vectors is processed by a fully connected layer to linearly project it to C_{in} dimensions. This results in the sequence Z_{emb}, consisting of $N_1 = \frac{H_0}{P} \cdot \frac{W_0}{P}$ patches that serve as the input to the first self-attention layer.

In the remainder of this thesis, a sequence-like structure of features is indicated by the notation $Z \in \mathbb{R}^{N \times C}$, while $A \in \mathbb{R}^{H \times W \times C}$ denotes an image-like structure of the corresponding data with $N = W \cdot H$ and H, W indicating the image height and width. The principle is visualised in figure 2.2. Note that indices will be added to Z and A to indicate a specific layer or stage of a model. Z and A can be rearranged at any time, which is necessary as some operations work in the spatial dimension (e.g. convolutions) while others need the sequence structure (e.g. self-attention).

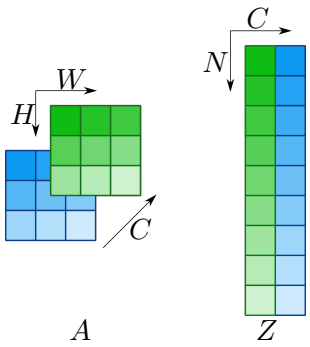

Figure 2.2: Visualization of the two different structures to handle the features in Transformer and convolutional models. A indicates the image-like structure, having the two spatial dimensions H and W, while Z indicates the sequence-like structure for which the spatial dimensions are combined to one dimension of size $N = H \cdot W$. Both data structures can be rearranged to the other format at any time in the model.

2.2.2.2 Positional encoding

In a self-attention layer, no information about the order of the individual elements in the input sequence Z_{emb} is available. By adding a positional encoding to the input embeddings, information about the relative or absolute positions of the feature vectors in the input sequence is provided to the model. The positional encoding can be used for inputs of different spatial, temporal or any other sequence-like structure. There are several ways to encode the positions of the input sequence. In general, the encodings can be categorised into learned and fixed encodings. While learned encodings include trainable parameters that are adjusted in the training process, fixed encodings are computed based on the positions of the elements in the input sequence and do not change in the training and inference processes. In this work, only the fixed positional encoding of Vaswani et al. (2017) is used. It is based on sine and cosine functions of different frequencies. This positional encoding has the same dimension C_{in} as the input embedding to be able to add it to the output of the patch embedding process. The i_c-th feature, with $i_c \in \{1, ..., C_{in}\}$ as the index for

the feature dimension, of the position encoding for the feature vector at position i_{seq} in the input sequence is computed by equation 2.21:

$$PE_{i_{seq},i_c} = sin\left(\frac{i_{seq}}{\tau^{\frac{2c}{C_{in}}}} + \frac{\pi}{2} mod(i_c, 2)\right), \tag{2.21}$$

with τ as a hyperparameter, which is typically set to $\tau = 10000$.

2.2.2.3 The attention mechanism of Transformer blocks

The attention mechanism processes input sequences of n_{seq} feature vectors $\mathbf{z}_i$ with $i \in \{1, ..., n_{seq}\}$, each having a dimension of C_{in}. The individual input vectors of an input sequence are combined into one matrix $Z \in \mathbb{R}^{n_{seq} \times C_{in}}$, in which each row represents one input vector $\mathbf{z}_i$. The main component of the attention mechanism, as introduced by Vaswani et al. (2017), is the scaled dot-product attention. It is based on three of such matrices, the query matrix Q, key matrix K and the value matrix V, all having n_{seq} rows. The value matrix is associated with the input features, resulting in C_{in} columns for V. The number of columns C_Q of Q and K can be different to C_{in}, but has to be the same for Q and K. The scaled dot product attention is derived by computing a matrix multiplication between Q and K and dividing the result by $\sqrt{C_Q}$. Afterwards, the softmax function (equation 2.6) is applied to normalise the output of each row and use it to weight V:

$$Att(Q, K, V) = SoftMax(QK^T/\sqrt{C_Q}) \cdot V = A^T \cdot V. \tag{2.22}$$

The derived features can be interpreted as a weighted mean of the input features in the value matrix. The attention matrix A represents the used weights, which are the inner product of the rows of the key (K) and query (Q) matrices and represent the similarity between these vectors. The output $Att(Q, K, V)$ has the same dimensions as the value matrix V.

Multi-head attention: Attention according to equation 2.22 can be computed using a single attention function. Another strategy is multi-head attention (MHA), which Vaswani et al. (2017) found beneficial compared to the single-head approach because it allows the model to extract information from different representations of the input data. In MHA, self-attention is computed in a number n_h of parallel heads. For this purpose, Q, K and V are linearly projected n_h times to compute self-attention for different sub-representations of the input data in parallel. Afterwards, the outputs of all heads are concatenated and linearly projected again to compute the final output:

$$\begin{aligned} MHA(Q, K, V) = MultiHead(Q, K, V) &= Concat(head_1, \ldots, head_{n_h}) \cdot W^O \\ head_h &= Att(Q \cdot W_h^Q, K \cdot W_h^K, V \cdot W_h^V), \end{aligned} \tag{2.23}$$

with $W_h^Q, W_h^K \in \mathbb{R}^{C_Q \times c_k}$, $W_h^V \in \mathbb{R}^{C_{in} \times c_v}$ and $W^0 \in \mathbb{R}^{n_h \cdot c_v \times C_{in}}$ denoting parameter matrices for the linear projections, h representing the index of head h, c_k the feature dimensions of Q_h and K_h and c_v corresponding to the feature dimension of V_h. Usually, the number n_h of heads and the dimension C_{in} are set as hyperparameters, and c_v, c_k are set to $c_v = c_k = C_{in}/n_h$. By doing so, the total computational costs are similar to those of single-head attention with $c_k = c_v = C_{in}$, but the model can extract features from different sub-representations of the data (Vaswani et al., 2017).

Self-attention: Depending on the computation of Q, K and V, there are two main variants of attention to be differentiated. In *self-attention*, Q, K and V are all computed from the same input Z, using three fully-connected layers (cf. section 2.1.1), which are represented by the parameter matrices $W^Q, W^K, \in \mathbb{R}^{C_{in} \times C_Q}$ and $W^V \in \mathbb{R}^{C_{in} \times C_V}$ that contain the learnable parameters. Multi-head self-attention (MSA) is then computed based on equation 2.23 by:

$$Z_{MSA} = MSA(Z) = MHA(Z \cdot W^Q, Z \cdot W^K, Z \cdot W^V). \tag{2.24}$$

In this way, all output feature vectors are dependent on all the feature vectors of the input, and therefore, global context is considered.

Transformer block: Commonly, self-attention is integrated in transformer blocks, which is shown in figure 2.3. Such a block consists of two components. The first one is the MSA layer and the second one is a MLP with two layers that have output dimensions of C_{MLP} and C_{in}, respectively, and having GELU as non-linearity. Layer Normalisation (LN) is applied before each component. Additionally, residual connections combine the inputs of the components with their outputs, realised by an element-wise matrix addition. Commonly, a number L of attention blocks are applied consecutively in a Transformer encoder, resulting in the following computation for block $l \in [1, ..., L]$:

$$\begin{aligned} \hat{Z}^l &= MSA(LN(Z^{l-1})) + Z^{l-1} \\ Z^l &= MLP(LN(\hat{Z}^l)) + \hat{Z}^l, \end{aligned} \tag{2.25}$$

with $\hat{Z}^l \in \mathbb{R}^{C_{in} \times n_{seq}}$ denoting the output of the MSA layer, $Z^l \in \mathbb{R}^{C_{in} \times n_{seq}}$ representing the output of the transformer block and Z^{l-1} denoting the output of the previous layer $l-1$. For block $l = 1$ the input sequence serves as input, i.e. $Z^{l-1} = Z_{emb}$ if patch generation is used.

Cross-attention: In *cross-attention*, K and V are computed similarly to self-attention. The query matrix Q, however, is computed based on another source S, i.e. $Q = S \cdot W^Q$. The source S can represent another input sequence, e.g. from another stage in the overall model. As a consequence, Multi-head cross-attention (MCA) is then computed by:

$$Z_{MCA} = MCA(Z, S) = MHA(S \cdot W^Q, Z \cdot W^K, Z \cdot W^V). \tag{2.26}$$

Commonly, cross-attention is integrated into transformer blocks too, adapted to handle both inputs Z and S, e.g. by using parallel linear layers and layer normalisation.

2.2.2.4 Transformer architectures for pixel-wise classification

Several methods adapt the Transformer model of Vaswani et al. (2017) for pixel-wise classification. The Swin Transformer (Liu et al., 2021) is one of the commonly used architectures and serves as the basis for the proposed method of this thesis (cf. section 4). Furthermore, the Utilise model of Stucker et al. (2023) is used as a baseline model for comparison. Therefore, both architectures are described in the following.

Swin Transformer: The Swin Transformer model introduced by Liu et al. (2021) is based on a hierarchical encoder-decoder structure. Those layers that extract features from the same spatial

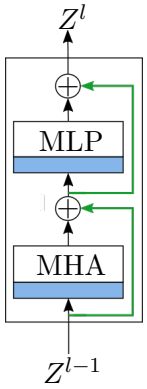

Figure 2.3: Structure of a self-attention block according to Vaswani et al. (2017). Z^{l-1}: input sequence to module l, Z^l: output of module l. MHA: Multi-head attention, MLP: Multilayer Perceptron, blue rectangle: layer normalisation, $\bigoplus$: element-wise addition, green arrows: residual connections.

resolution are referred to as a stage. In each stage, modified self-attention blocks, i.e. W-MSA instead of MSA, are applied to extract spatial context. Between subsequently applied encoder stages, patch merging layers are used to decrease the spatial resolution. Both layer types are explained in the following, before the Swin Tranformer architecture is explained in detail.

Shifted-window based self-attention: To be able to use a comparably small patch size in a Transformer model for pixel-wise classification, Liu et al. (2021) introduce the **s**hifted **win**dow (swin) approach, which is applied to reduce the computational complexity of self-attention.

For the W-MSA, window partitioning is applied before self-attention is computed for all windows in parallel in the way described in section 2.2.2.3. In window partitioning, the input $A \in \mathbb{R}^{H \times W \times C}$ with (H, W) as the current height and width of the feature maps and C as the feature dimension of the layer, is arranged into non-overlapping windows, each containing $M \times M$ patches and starting from the top-left image patch. Afterwards, multi-head self-attention is computed for all windows separately in self-attention blocks as shown in figure 2.3. Again, L self-attention blocks are applied consecutively, resulting in the following computations for block l:

$$\begin{aligned} \hat{Z}^l &= W\text{-}MSA(LN(Z^{l-1})) + Z^{l-1} \\ Z^l &= MLP(LN(\hat{Z}^l)) + \hat{Z}^l, \end{aligned} \tag{2.27}$$

with Z^{l-1} as the output of the previous W-MSA block, $\hat{Z}^l \in \mathbb{R}^{C_{in} \times N}$ as the output after the W-MSA layer and $Z \in \mathbb{R}^{C_{in} \times N}$ as the output of the self-attention block. For the first block $l = 1$, Z_{emb} serves as input. $W\text{-}MSA$ indicates that windows are created before MSA is computed in each window separately, and directly after the MSA, the output features for all patches are structured as Z^{l-1} again. All other computations are jointly applied to all patches in Z^{l-1}.

As there are no connections between the individual windows, dependencies between patches in different windows are not computed. To overcome this limitation, the windows are shifted by $\frac{M}{2}$ patches in height and width for the next self-attention module, which results in a second window configuration that introduces connections between patches that were separated into different windows before. This process results in some windows that are smaller than the window size $M \times M$.

For these cases (Liu et al., 2021) introduce a cycling shift approach in which smaller windows from the right or top margin of the image are combined with those from the left/lower margin. A typical window size is $M = 7$, resulting in $M^2 = 49$ patches being considered in one MSA computation.

Patch merging: To produce a hierarchical representation, the spatial resolution needs to be reduced. To be able to do this, the feature maps $A \in \mathbb{R}^{H \times W \times C}$, with (H, W) as the height and width of the feature maps and C as the feature dimension, are split into non-overlapping regions of 2×2 patches each. Afterwards, the C features from all four patches in one region are concatenated to a vector with dimension $4C$. Finally, a fully connected layer is applied to these vectors to transform them to the output dimension of $2C$. This reduces the number of patches by a factor of four, which is equivalent to reducing both spatial dimensions H and W by a factor of two.

Model: Let $X_0 \in \mathbb{R}^{B \times H_0 \times W_0}$ be an input image with B, H_0 and W_0 indicating the number of spectral bands, height and width of the input image, respectively. First, patch embedding, as described in section 2.2.2.1, is applied using a patch size P, resulting in $Z_{emb} = \{\mathbf{z}^1, ..., \mathbf{z}^{N_1}\}$ with $N_1 = H_0/P \cdot W_0/P$ and C_{in} as the feature dimension. Z_{emb} serves as input to the first stage of the Swin encoder. The encoder stages are denoted by E_s, with $s \in [1, ..., S]$ being the index of a stage and S representing the total number of stages. In each stage, a certain number L_s of W-MSA blocks are applied consecutively, with $l_s \in [1, ..., L_s]$ denoting the lth block in stage E_s. Between subsequent stages, patch merging is applied, which reduces the number of patches by a factor of four. The number of feature maps in the first stage is C_{in}, which is doubled whenever patch merging is applied, i.e. in stage E_s the number of feature maps is $C_s = C_{in} \cdot 2^{(s-1)}$. The input and output of each stage are $Z_{s-1}^{PM} \in \mathbb{R}^{C_s \times N_s}$ and $Z_s \in \mathbb{R}^{C_s \times N_s}$, respectively, with PM denoting the output after patch merging, C_s denoting the feature dimension, $N_s = H_s \cdot W_s$ representing the number of patches and H_s and W_s corresponding to the image height and width in stage E_s, respectively. The Swin Transfomer can be combined with any hierarchical decoder architecture. Liu et al. (2021) combine it with the UPer-Net, a convolutional decoder (Xiao et al., 2018); cf. section 2.2.1.5.

Utilise: Utilise is a sequence-to-sequence model that Stucker et al. (2023) introduced for the application of cloud removal from SITS. The model follows an encoder-decoder structure and is based on convolutional layers. Additionally, a so-called Lightweight Temporal Attention Encoder (L-TAE) module is used in the deepest layer of the model to compute temporal features with self-attention. In the following, the three components (encoder, decoder, L-TAE) of the Utilise model are described.

Encoder: An input timeseries X of size $T \times B \times H_0 \times W_0$ serves as input to the encoder, which consists of several stages. In each stage, a convolutional block (CB) is applied separately to all timesteps. In the Utilise model, a CB consists of a convolutional layer with $k = 3$ and C_s as the feature dimension in stage $s \in \{1, ..., S\}$, with S as the total number of stages, followed by ReLU activation and a residual convolution with $k = 3$ and C_s' as the feature dimension. Between the CBs, a convolutional layer with $str = 2$, followed by ReLU activation, decreases the spatial resolution by a factor of two.

Lightweight Temporal Attention Encoder (L-TAE): The output $A_S \in \mathbb{R}^{T \times C_S' \times H_S \times W_S}$ of the last encoder stage E_S serves as input to the L-TAE module. In this module, temporal dependencies are

considered based on an adaptation of MSA with the goal of reducing computational complexity. Therefore, A_S is separated into $N_S = H_S \times W_S$ sequences, with $Z^{n_p} \in \mathbb{R}^{T \times C'_S}$ denoting the n_p^{th} sequence. For each sequence Z^{n_p}, a modified MSA block is applied. A key modification in the L-TAE is the computation of the input matrices used to map the features to the inputs Q_h, K_h and V_h for head h. Instead of performing linear projections to query, key, and value matrices, the input sequence Z^{n_p} is partitioned into n_h groups, each having a feature dimension of $c_k = C'_S/n_h$ (cf. equation 2.33). Consequently, each head receives an input $Z_h \in \mathbb{R}^{T \times c_k}$. Z_h is then used to compute Q_h, K_h by a linear transformation, while the input is directly used as V_h:

$$MSA(Z^{n_p}) = MHA(Q_h, K_h, V_h) = Concat(head_1, \ldots, head_{n_h}) \cdot W^O \tag{2.28}$$

$$head_h = Att(Q_h, K_h, V_h) \tag{2.29}$$

$$Q_h = (Z_h + TE_h) \cdot W_h^Q \tag{2.30}$$

$$K_h = (Z_h + TE_h) \cdot W_h^K \tag{2.31}$$

$$V_h = Z_h + TE_h \tag{2.32}$$

$$Z_h = Z^{n_p}[1:T, (h-1)c_k + 1 : hc_k]. \tag{2.33}$$

with $W_h^K, \in \mathbb{R}^{c_k \times c_k}$ and $W^O \in \mathbb{R}^{n_h \cdot c_k \times C_{in}}$ as parameter matrices for the linear projections and h as the index for the individual heads. TE is a position encoding (cf. equation 2.21) applied to the temporal dimension of Z^{n_p}. The L-TAE block is integrated into the self-attention block as depicted in figure 2.3, replacing the MSA block.

The outputs of the temporal encoder are used to weight the outputs of all stages of the spatial encoder. To be able to do this, the outputs of the temporal encoder are upsampled by bilinear interpolation to the required spatial resolution of the different stages. Afterwards, the temporal features are multiplied with the outputs of the spatial encoder by element-wise multiplication. This is followed by a convolutional layer with $k = 1$ and ReLU activation. The outputs of these temporal weightings in the skip connections serve as additional input to the decoder stages.

Decoder: The spatial decoder consists of several stages D_s with the same spatial resolutions as those of the corresponding encoder stage E_s. Note that the decoder stages are counted backwards to have the same index $s \in \{S, ..., 1\}$ as the encoder stage with the same spatial resolution. For stages $\{S-1, ..., 1\}$, each stage consists of a CB with the same structure as in the encoder. Between the stages, an upsampling layer with a transposed convolution is applied to increase the spatial resolution by a factor of two for the following stage. For stage S, i.e. the stage directly after the L-TAE block, no CB is applied, and the output of the L-TAE block directly serves as input to the upsampling layer. After each upsampling layer, the features from the decoder are concatenated with the features from the skip connections. After the last stage of the model, the last CB is followed by a final CB that maps the features to the spectral space again. This final layer uses a sigmoid activation to regress reflectances in the range from 0 to 1.

3 Related Work

This chapter gives an overview of the current state-of-the-art in the fields related to this thesis. It starts with a general overview of methods used for pixel-wise classification in section 3.1 and time series classification in section 3.2, before an overview of methods related to these topics in the field of Remote Sensing is given in section 3.3. Section 3.3 additionally includes a discussion about training data for RS applications, the temporal and spatial position encoding, and computational complexity of Transformer models. In the last section of this review, the most important aspects are summarised, and open research questions related to this thesis are discussed, thus identifying the research gap to be addressed in this thesis.

3.1 Deep learning for pixel-wise classification

In pixel-wise classification, a class label is assigned to each pixel in the input image, based on a pre-defined class structure, as explained in detail in section 2.1.4. In DL, the classifier takes the image directly as input and predicts class probabilities that describe the semantic class of the individual pixels. This is possible because DL models learn to extract meaningful features from the input images that are relevant to distinguish the different classes. This is a contrast to traditional machine learning approaches, e.g. Random Forests (Breiman, 2001) or Support Vector Machines (Cortes and Vapnik, 1995), which are based on hand-crafted features defined by an expert, which are extracted from the images before they serve as input to a classifier.

The first DL models using image data were Convolutional Neural Networks (CNNs) for image classification, which take an image as input and predict one class label for it. They were first introduced by LeCun et al. (1989) for the task of hand-written zip-code recognition. With the advances in hardware in combination with the availability of larger training datasets it became possible to train model s with much more trainable parameters. Thus, CNNs were revived by Krizhevsky et al. (2012). The first models for pixel-wise classification were Fully Convolutional Neural Networks (FCNs) introduced by Long et al. (2015), who adapt CNNs to the task of pixel-wise classification. Long et al. (2015) apply convolutions to larger images and replace the last fully connected layers of a CNN for image classification by an upsampling layer to create an output of the same spatial size as the input image. Noh et al. (2015) introduce the encoder-decoder structure for pixel-wise classification. While in the encoder the spatial resolution is successively downsampled by pooling operations between the convolutional layers, the original spatial resolution is reconstructed by applying upsampling operations in the decoder. With the goal of not losing the spatial information about class boundaries, Noh et al. (2015) save the maximum-pooling indices of the used values and use them again in the upsampling layers to reconstruct the exact placement

of the original values. Whereas this is a simple and lightweight way to restore fine-grained spatial information, it can also prevent the model to learn more complex spatial relations, e.g. when context information is important for the application. Ronneberger et al. (2015) introduce the U-Net architecture, originally for the application of biomedical image segmentation. As the reconstruction of the correct positions of object borders is still a challenge, Ronneberger et al. (2015) introduce skip connections to counteract this problem. Skip connections (c.f. section 2.2.1.4) fuse feature maps from the encoder layers to corresponding layers in the decoder to include information about the exact position of boundaries from the earlier layers to the decoder. Since the success of the U-Net architecture, it has been used as the basis for many FCN variants, e.g. UPer-Net (Xiao et al., 2018). However, convolutions only capture local spatial context in the earlier layer of a CNN. With deeper layers, more global context is integrated, caused by the downsampling layers, but the spatial resolution is coarser at this point. Therefore, the accurate prediction of object borders remains a challenge for hierarchical approaches, which is only partially overcome by using skip-connections.

With the success of Transformers, introduced by Vaswani et al. (2017) for tasks in the field of NLP, they have been adapted to various tasks in Computer Vision, including image classification (Dosovitskiy et al., 2021), object detection (Li et al., 2022b) and pixel-wise classification (Strudel et al., 2021). Whereas CNNs are based on convolutions that capture local context, e.g. in the spatial neighbourhood of a pixel, Transformers are based on the principle of self-attention (c.f. section 2.2.2.3). Self-attention considers global and local context directly based on all input feature vectors, considering features independently from their order in the input sequence. In the context of this thesis, the term *self-attention* always refers to the type of self-attention as used in the Transformer model (Vaswani et al., 2017). In the field of NLP, the input and the output commonly consist of sequences of words, and the input feature vectors are obtained by transforming the words into feature vectors of a defined dimension. Dosovitskiy et al. (2021) introduce the Vision Transformer (ViT) for image classification. They modify the input layer of the original Transformer by stacking the grey values of image patches (e.g. 16×16 pixels), which is followed by a linear projection to obtain feature vectors with defined dimensionality again. Afterwards, these patches, i.e. feature vectors which represent a certain area of the input image, are used in the standard Transformer model and self-attention layers are used to compute new features that encode dependencies between them. Strudel et al. (2021) further adapt the ViT for semantic segmentation by using a decoder based on convolution and upsampling layers to determine pixel-wise class labels.

Similar to the downsampling process in CNNs, the classification of fine details and object borders suffers from the patch generation process in the first layer of a Transformer model. Patch generation is necessary because the length of the input sequence and, therefore, the computational costs drastically increase for image data: If all pixels of an image were used as input features, self-attention would be computed between all combinations of the $H \cdot W$ pixels. The number of inputs is reduced to $\frac{H}{P} \cdot \frac{W}{P}$ if patches are generated. The selection of the patch size P is therefore always a trade-off between computational costs and classification accuracy. To be able to use a smaller patch size, Liu et al. (2021) introduce another strategy to decrease the computational costs for pixel-wise classification with Transformer models. In their Swin Transformer, self-attention is computed only in local windows consisting of a fixed number of patches (cf. section 2.2.2.4). The assumption behind this is that neighbouring patches include the most important feature information that is needed to

classify an individual patch. To allow the model to include global information, these windows are shifted between subsequent network layers to allow an information flow between them. Additionally, Liu et al. (2021) use a hierarchical representation, gradually merging neighbouring patches in deeper layers. As a result, the Swin Transformer can be used as an encoder in combination with any hierarchical convolutional decoder that upsamples the feature maps to the original resolution again. Liu et al. (2021) combine their Swin Transformer with the UPer-Net decoder. Similar to CNNs, the classification of fine objects and class boundaries remains a challenge, even though a smaller patch size (a standard value is a patch size of $P = 4$) can be used in the Swin Transformer architecture. Furthermore, it is highly dependent on the application and object size whether the spatial context contained in the used windows is appropriate to model the dependencies that are most relevant to obtain good classification performance. This aspect is analysed with resepct to Remote Sensing data in section 3.3.1 and, as there are several other strategies to decrease the computational complexity in Transformer models, this aspect will be discussed in detail in section 3.3.4.

3.2 Deep Learning for timeseries classification

Timeseries are encountered in many real-world applications, e.g. human activity recognition (Mutegeki and Han, 2020), video scene classification (Arnab et al., 2021) or health analysis (Song et al., 2018), resulting in a large research field. Commonly applied methods are based on DL and include MLPs, CNNs, Recurrent Neural Networks (RNNs), and Transformer models. Mohammadi Foumani et al. (2024) define a time series as an ordered set of T pairs of observations and timesteps. The observations for one timestep, however, can consist of many data types, from scalar values like temperatures to feature vectors or feature tensors of higher dimension. Most approaches in the field of timeseries classification (TSC) work with feature vectors as input data, which is, therefore, the first focus of this section. Afterwards, approaches working with image time series, e.g. video sequences, are discussed as this data type is closely related to the data used in this thesis.

Deep Learning for TSC: Fawaz et al. (2019) and Mohammadi Foumani et al. (2024) both discuss the different types of DNN for TSC. Both agree that the commonly used models for TSC developed from MLPs to CNNs and RNNs and, in more recent developments, to Transformer models. Methods are commonly tested on a variety of datasets, e.g. the UCR time series classification archive, a large publicly available repository for TSC datasets established by the University of California, Riverside (UCR), and can be ranked based on accuracy metrics achieved over all the available datasets (Dau et al., 2018). While the use of MLPs is easy to set up and train, the input data is processed in a fixed and determined order, which means that special temporal relationships, e.g. irregular temporal intervals, are not considered. Furthermore, in most MLPs, no hierarchical or multi-scale features are computed - a drawback, as time series often include long and short-term trends (Mohammadi Foumani et al., 2024). Due to these characteristics, MLPs perform worse than other DL-based methods for time series classification (Fawaz et al., 2019; Wang et al., 2017; Zheng et al., 2014). The second model type that can be used for TSC are RNNs, which are a specific type of DNN built to work with sequential data by integrating an internal memory about the previous points in the sequence, first introduced by Rumelhart et al. (1985). One of their main advantages

is their ability to handle variable input and output lengths. Several adaptations of the original RNN exist, e.g. the Long short-term memory (LSTM) introduced by Hochreiter and Schmidhuber (1997) to counteract problems such as vanishing or exploding gradients that occur due to the back-propagation process over long sequences. In the field of TSC, Hüsken and Stagge (2003) introduce sequence-to-sequence RNNs for the task of trajectory classification. In their experiments, one challenge was the long training time, which they could improve by giving an additional prediction task to the model. Several approaches that followed use a hybrid CNN-RNN structure to combine the advantages of both methods (Karim et al., 2019; Mutegeki and Han, 2020). Mohammadi Foumani et al. (2024) conclude that several approaches use RNNs for TSC, but it is not the commonly used approach anymore, mainly due to a more difficult training procedure compared to other methods and challenges such as the vanishing gradient problem.

Since their success in image- and pixel-wise classification, approaches based on convolutions are also applied in other fields, including TSC. A special case of TSC is the usage of images as observations, however, this aspect is discussed later in this section. When feature vectors for each timesteps serve as input to the model, the common strategy is the usage of 1D convolutions that shift over the timeseries to extract features from neighbouring timesteps. Zheng et al. (2014) were the first to use a CNN for TSC by applying such 1D-convolutions to each input channel in parallel and concatenating the output features that are then processed by a fully connected layer. Since then, there have been many adaptations. For instance, Zhao et al. (2017a) use a CNN based on 2D-convolutions shifting the kernels in the temporal and the feature dimensions, to explot temporal dependencies within each timeseries in combination with dependencies between the features. Afterwards, a MLP is used to map the computed features to the class scores. Several more approaches apply 1D-convolutions to timeseries of feature vectors (Wang et al., 2017; Geng and Luo, 2019; Ismail Fawaz et al., 2020; Fawaz et al., 2019). In summary, convolutional approaches outperform MLP and RNN models in most cases (Mohammadi Foumani et al., 2024). Nevertheless, they also have some limitations. For example, only local features, i.e. dependencies between neighbouring timesteps, are captured using the convolutional operation, and temporal dependencies over a larger number of timesteps are not considered because most models do not downsample the data in the temporal dimension. Furthermore, irregular temporal intervals cannot be taken into account as the same filter matrix is used for all timesteps in a sliding window approach of a convolutional layer. This can potentially result in difficulties to encode temporal dependencies if the temporal intervals differ too much within one timeseries, or if the time periods differ between training and test samples. Furthermore, it is not clear whether these conclusions can be transferred to image timeseries.

Following the success of Transformer models in the field of NLP (Vaswani et al., 2017), the Transformer principle was also adopted for TSC, motivated by the intrinsic similarity to NLP due to the sequential order of both data types. Self-attention solves some of the challenges that convolutional or recurrent approaches have to face. The main advantages are that global and local temporal context can be captured within one operation, and that the temporal distance between two timesteps has no impact on the extracted features. Song et al. (2018) were the first to adopt the Transformer model from (Vaswani et al., 2017) to clinical time series classification. They used the Transformer encoder from (Vaswani et al., 2017), followed by dense interpolation to fuse the

features from all timesteps. In their experiments, including different clinical time series datasets such as mortality and length of stay, the new method outperformed the LSTM baseline, especially if the different classification tasks were combined into a multi-task learning framework. Zhao et al. (2023a) introduce several adaptations of the Transformer model tailored to TSC. The authors use deformable convolutions to extract features that serve as input to the subsequent self-attention modules. The idea is to use the convolutions to consider local temporal context, and self-attention to capture global temporal context. Additionally, they use a random masking mechanism to speed up training and reduce noise. They achieve good results compared to several baseline approaches while requiring a smaller number of parameters.

In summary, there is much research in the field of DL for TSC for many different applications. Before the success of self-attention, approaches based on convolutions achieved the best performance, but, they come with some limitations, e.g. only local temporal context can be considered in one convolutional layer. This limitation could be overcome by using Transformer models, which has the advantage that temporal dependencies can be incooperated based on all input timesteps within one operation and independent of the temporal distance of two timesteps.

Deep Learning for TSC with image data: In such approaches, images as observations for each timestep. Besides SITS, that are one type of temporal image sequences in RS, video scenes are a widely used type of TSC with image data. Videos are typically captured by cameras from a fixed or moving viewpoint that record the dynamic environment. In comparison to SITS, which usually cover the same area on the ground for multiple timesteps, the scene in a video changes from frame to frame as, typically, objects in the scene move or the camera itself is moving, e.g. when it is mounted on a vehicle. Therefore, the applications for video scenes differ from the applications for SITS and include object detection and tracking, e.g. for video surveillance (Gao et al., 2023; Du et al., 2024) and scene understanding, e.g. in the context of autonomous driving (Muhammad et al., 2022). The variety of applications results in a large amount of research in the field of video scene classification, but it also comes with some specific challenges, e.g. occlusions, distinguishing similar objects, or changing properties when objects move. Muhammad et al. (2022) give an overview of recent methods applied to scene understanding and multi-object tracking, while Du et al. (2024) survey the recent approaches for multi-object tracking. On the methodological side, several methods used in pixel-wise classification or TSC without images as observations are also used for different classification task with video scenes. Common methods to classify objects and to track them over time are based on DNN and extract spatial features based on encoder-decoder architectures, similar to methods for pixel-wise classification as discussed in section 3.1. For instance, in video object segmentation, the segmentation process can be conducted frame by frame based on FCNs (Gao et al., 2023). Recent methods are also based on self-attention, Du et al. (2024) summarise several self-attention-based methods for object detection that consider temporal and spatial interactions, for instance, between multiple potential objects.

To summarise, the basic methods used in video classification are the same as those used in pixel-wise classification or other TSC tasks. However, the applications are different as usually objects are tracked over time, resulting in different needs and challenges of approaches to solve these tasks.

3.3 Classification of remote sensing data

In the field of Earth Observation, optical RS sensors capture reflected energy from the Earth's surface by different sensors that work from a distance, e.g. operated from a satellite or an aircraft. The acquired data are used for further analysis, e.g. to assist in a map updating or monitoring tasks regarding Earth resources (Lillesand et al., 2015). A large number of sensors and platforms exist. For instance, over 1200 satellites are currently orbiting the Earth with the purpose of remote sensing[1]. This overview focuses on methods related to RS images acquired from satellites or aircrafts, which are commonly based on DL, frequently adapting approaches from the Computer Vision domain (cf. sections 3.1 and 3.2) to the specific characteristics of RS imagery. One property of RS images which are commonly acquired with near nadir viewing direction, i.e. the camera is facing vertically downward from the platform, is an almost constant spatial resolution on the ground, expressed by the Ground Sampling Distance (GSD), in the whole image or even a larger image block, as the platform usually flies at a constant height above the ground. The information about the GSD can be taken into account when choosing the receptive field of the model. Depending on the application, spatial context in the direct neighbourhood of a pixel or from pixels that are more distant can be important to achieve a good classification performance. While satellite images frequently have a medium spatial resolution, i.e in the range of meters to decimeters, they often come with a high temporal resolution when the satellite constellations cover the same area on the ground in regular intervals, i.e. 5 days in the case of Sentinel-2. Therefore, in addition to spatial context, temporal dependencies can be considered when using satellite images for tasks such as pixel-wise classification, which can improve the classification performance. In this section, first, an overview of current research for pixel-wise classification of RS imagery is given in section 3.3.1 before this topic is extended to multi-temporal satellite images in section 3.3.2.

3.3.1 Pixel-wise classification in remote sensing

Methods for pixel-wise classification of RS imagery are mainly based on convolutions and self-attention. The applications involve many areas, e.g. road extraction (Zhang et al., 2018; Ayala et al., 2021), building segmentation (Kaiser et al., 2017), change detection (Caye Daudt et al., 2019), tree species classification (Lobo Torres et al., 2020), verification of topographic databses (Yang et al., 2021) or land cover classification (Yang et al., 2021; Pelletier et al., 2019; Voelsen et al., 2022). Convolutional approaches are mainly based on FCNs such as U-Net (Ronneberger et al., 2015), SegNet (Badrinarayanan et al., 2017), or DeepLab architectures (Chen et al., 2018), which have been adapted to the remote sensing domain. In most cases, the models are only slightly changed, e.g., by including residual connections to the standard U-net (Zhang et al., 2018). Lobo Torres et al. (2020) compare several FCN architectures for tree species classification with drone images. In their study, less complex models achieve the best results, which the authors explain by the limited availability of training samples for this application. In some scenarios, the specific characteristics of the data can be included into the model to further improve the performance. In this regard, Yang et al. (2021) introduce a hierarchical model that categorises land use on different semantic levels, e.g., a pixel is assigned to the coarse class *Forest* and to the type of forest at the finer level,

[1] https://www.ucsusa.org/resources/satellite-database, accessed on 10-12-2024

which improves the results for pixel-wise land use classification and can be used for the verification of topographic databases afterwards. Maggiori et al. (2017) extend a FCN with a module that learns how to combine features from different spatial resolutions that are included in their training dataset. This module shows improvements in the performance compared to other approaches for pixel-wise classification and is one way to integrate local and global context into the model.

Recent research focuses on attention-based and hybrid approaches combining self-attention and convolutional modules to determine class labels at pixel level. Aleissaee et al. (2023) give an overview of the latest research in the field of RS. There are only a few approaches that are purely based on self-attention. For instance, Zhang et al. (2022b) use a Transformer network for change detection by replacing all convolutional layers in a Siamese encoder-decoder network by Swin Transformer layers. In their experiments involving four change detection datasets, their model outperforms the baseline methods in most but not all cases. Xu et al. (2021) use an encoder based on the Swin Transformer in combination with a MLP decoder for edge detection in aerial images. Using the MLP instead of a decoder based on UPer-Net slightly decreases the quality of the results but is computationally much less complex. The most frequently used approaches use a combination of self-attention and convolution. A common strategy to do so is to combine a Transformer encoder with a decoder similar to the one used in U-Net (Zhang et al., 2022a; Wang et al., 2022b; Panboonyuen et al., 2021; Hanyu et al., 2024). In this regard, Panboonyuen et al. (2021) compare different decoder designs that are based on convolutions in combination with a Swin Transformer encoder for LC and crop classification. In their experiments with satellite and aerial imagery, the model that uses a pyramid scene parsing module in the decoder outperforms the other variants, which highlights the importance of global spatial features for their application.

One challenge when applying Transformer models to RS imagery is the patch generation process. For GSDs of several meters, which is the current standard for freely available satellite data, this results in patches that can cover several objects on the ground, which results in difficulties in classifying pixels merged into one patch but belonging to different classes correctly. Hanyu et al. (2024) counteract this problem by integrating an additional skip connection at the original spatial resolution of the input image into their model which is based on self-attention in the encoder and convolutions in the decoder. Hanyu et al. (2024) apply the model to several datasets with aerial imagery for the application of land use and land cover classification. With the proposed adaptation, their model achieves better classification results for tiny objects, e.g. vehicles, in comparison to other baseline models. Several approaches use convolutions in parallel with attention in the encoder part of the network (Gao et al., 2021; He et al., 2022; Wang et al., 2022a; Xiao et al., 2022; Yao et al., 2024). The idea of these approaches is to extract local spatial context by convolutions, while global spatial context is exploited by the attention modules. However, most approaches use the Swin Transformer to decrease the computational complexity, resulting in self-attention layers that compute spatial dependencies in local windows, which restricts the receptive field of the model. Global features can only be computed in deeper layers with reduced spatial resolution, similar to encoder-decoder architectures that use convolutions.

To summarise, the methods applied to remote sensing images are based on convolutions and self-attention or hybrid variants to combine the advantages of both methods. The specific characteristics

or applications, such as the larger GSD, lead to modifications of the models with the goal to further improve the classification performance, but, the requirements differ between applications and also depending on the input data, i.e. the used GSD or available spectral bands. This makes it difficult to transfer results achieved for one application to another one; e.g. results based on aerial imagery with 10 cm GSD to satellite images with 10 m GSD. A further challenge when applying DL methods to RS imagery is the availablitly of large training datasets. This aspect is discussed in section 3.4.

3.3.2 Classification of satellite image time series

Most satellites for earth observation, including the Sentinel or Landsat missions, provide images of the same areas with constant revisiting times, producing SITS. The temporal information can help to model dynamic processes like seasonal variations in the vegetation or rapid changes like natural disasters (Miller et al., 2024). DL models can extract meaningful features from such complex data under the condition that enough training data are available to learn these characteristics. Methodologically, SITS are one example for TSC with image data (section 3.2). While earlier works focus on RNNs, FCNs and combinations thereof, most current approaches use a Transformer-based architecture, commonly combined with convolutional parts, e.g., in the decoder. In this section, the most important works related to the topics of this thesis are introduced, structured into paragraphs discussing convolutional methods, self-attention based methods and hybrid models that combine convolutions and self-attention. Afterwards, several approaches that predict maps for several timesteps are discussed. For a broader review, we refer the reader to Miller et al. (2024). Further challenges to use SITS in DL are available training datasets (section 3.4), a temporal position encoding as the order of the sequence is not taken into account in self-attention (section 3.3.3) and the computational complexity (section 3.3.4), which are discussed separately.

Convolutional methods for SITS classification: While convolution-based models are mainly used to exploit spatial context, there are several adaptations for SITS classification, including methods based on 1D-, 2D- and 3D convolutions. In SITS classification, 1D convolutions are commonly used to encode temporal dependencies. Di Mauro et al. (2017) were among the first authors to apply 1D convolutions to the temporal dimension and won a challenge in which Landsat-8 timeseries are used for LC classification. Pelletier et al. (2019) propose a temporal CNN to classify crop types from SITS. They compare 1D convolutions applied in the spectral or temporal dimensions and 2D convolutions, which shift the kernel in the temporal and spectral dimensions to other baselines, including a RNN. Their model, using 2D convolutions, outperforms the other approaches. One drawback of the methods mentioned so far is that they do not consider any spatial context. 2D convolutions are widely used to capture spatial context (cf. section 3.3.1). When applied to SITS data, the question arises, how to compute and combine spatial, temporal and spectral dependencies. One strategy is the usage of 3D convolutions. Instead of sliding the kernels in two dimensions (2D convolutions), a 3D kernel slides in three dimensions of the input image, e.g. the spatial and temporal or spatial and spectral dimensions. The main drawback of using 3D convolutions is the additional computational complexity as the number of computations increases by a factor equal to the size of the third dimension. Several approaches use 3D convolutions for spatio-temporal feature extraction. For example, Ji et al. (2018) use a 3D-CNN for crop classification using SITS,

applying 3D convolutions in the spatial and temporal dimensions of an input region of 8×8 pixels. The generated output is a feature vector of class scores for the central pixel of the input region. The authors conclude that the 3D-CNN is especially suitable for modelling the dynamics of crop growth. Similarly, Teimouri et al. (2022) use 3D convolutions to extract spatial-temporal features from optical and radar SITS for crop classification on pixel-level. In contrast to Ji et al. (2018), who combine all spectral bands when applying the 3D convolution, Teimouri et al. (2022) separate the spectral bands for the 3D convolutional layer and combine them afterwards. In this way, their model performs better than other 3D-CNNs they compare their method to. Voelsen et al. (2022) compare a 3D U-Net for LC classification based on SITS with a 2D U-Net and a variant that uses 3D convolutions only in the first layers of the model. In the 2D U-Net, the spectral and temporal dimensions are stacked before the input is processed by the model, resulting in $T \times B$ input features, with T as the number of timesteps and B as the number of spectral bands. In their experiments, the usage of 3D convolutions only slightly improves the model's performance, while the usage of multi-temporal input data significantly improves the results in comparison to a mono-temporal variant. Fernandez-Beltran et al. (2021) use a 3D-CNN for a regression task. In order to estimate rice yield, the authors use 3D convolutions to extract spatial-temporal features from large-scale SITS and additional soil and climate data and achieve better results than other traditional and DL approaches.

While using 3D convolutions results in improved classification performance in comparison to models based on 2D convolutions, they also come with some limitations. The increase in computational complexity results in longer training and inference processes, and a balance needs to be found between model performance and computational time. Another drawback of applying convolutions to the temporal dimension is that changing time intervals can be difficult to handle, because information about changes in the temporal distances between subsequent image acquisition times is not taken into account and the same weights are used for the whole timeseries (Rußwurm et al., 2020). Furthermore, the extraction of global temporal context is not included in the discussed approaches. A larger context region could be integrated by larger convolutional kernels in the temporal dimension, for example, by applying dilated convolutions, however, irregular temporal distances remain a problem.

Self-attention for SITS classification: Most recent approaches for SITS classification focus on Transformer models. The characteristics of SITS, e.g. irregular temporal intervals, seasonally varying appearance of vegetation, and almost constant appearance of man-made objects, indicate that self-attention might be particularly suitable for modelling temporal dependencies. This overview focuses on the most recent research based on the self-attention mechanism used in the Transformer model of Vaswani et al. (2017). SITS have one dimension more than images and, therefore, computational complexity is a challenge which is even more problematic for this type of input data. Approaches applying Transformers to SITS follow different strategies to deal with this challenge, which is analysed in this section.

Transformer models consider dependencies between all input feature vectors simultaneously, which means that global and local dependencies can be exploited within one computation. To integrate information about the order of the input sequence into the model, a positional encoding

is commonly added, e.g. in the beginning of the network (cf. section 3.3.3). Such an encoding can be extended to the spatial or temporal dimensions when SITS are used. A temporal encoding has the advantage that information about irregular time periods can be integrated into the model. Rußwurm and Körner (2020) were among the first authors to adapt the encoder of the Transformer model for the pixel-wise classification of SITS. Their Transformer architecture outperforms architectures such as a 1D temporal CNN and a LSTM on unprocessed satellite data, i.e. data without atmospheric corrections, while these methods perform equally well on preprocessed data. However, Rußwurm and Körner (2020) only use pixel-wise timeseries, and therefore, no spatial context is used. Another purely attention-based approach is introduced by Tarasiou et al. (2023), who adapt the ViT (Dosovitskiy et al., 2021) for crop classification with SITS. First, patches are generated similar to the way it is done in the Swin Transformer model. To reduce the computational complexity, self-attention is first computed between all timesteps of the same patch, and afterwards, in the subsequent layer, attention is computed between all patches of the same timestep, arguing that this order of feature extraction is more suitable for their application. However, it remains unclear if this assumption is transferable to other applications. Yan et al. (2022) classify LC from SITS using an architecture that computes dependencies only for timesteps determined to be important by using modified self-attention layers. This is a way to reduce computational costs for long input sequences. While this model outperforms other methods, e.g. a LSTM network, the authors do not compare it to the standard Transformer approach, and no spatial context is considered in the model.

Hybrid approaches for SITS classification: The majority of approaches combine self-attention layers to extract temporal dependencies with convolutional layers to extract spatial context (Garnot and Landrieu, 2021; Zhang et al., 2023a; Li et al., 2022a; Voelsen et al., 2024). In this regard, Garnot and Landrieu (2021) use a multi-temporal adaptation of U-Net with a Lightweight Temporal Attention Encoder (L-TAE) module (cf. section 2.2.2.4 in the bottleneck layer for the segmentation of crop parcels. In the encoder, convolutional layers are applied separately to all timesteps. The L-TAE module computes temporal attention at pixel-level of the coarsest resolution; afterwards, these feature maps are upsampled to all spatial resolutions and used in the skip connections to weight the different timesteps, which is realised by an element-wise matrix multiplication. Furthermore, Garnot and Landrieu (2021) aggregate the temporal dimension as only one output timestep is needed. This is done by a sum over the results of the matrix multiplication across the temporal dimension. Afterwards, a convolutional decoder is used to predict a crop map at the original spatial resolution. Zhang et al. (2023a) use the L-TAE module to encode temporal dependencies between all temporal input features of the input time series. Separately, convolutions are used to capture temporal dependencies from neighbouring timesteps. Finally the features are fused using a MLP to predict crop types and LC. This method performs better than others against which it is compared, including a LSTM and temporal CNN, but as only time series of single pixels are processed by the self-attention module, no spatial context is considered. Li et al. (2022a) propose a hybrid model for crop classification using optical and radar satellite imagery. They first use depthwise separable convolutions to exploit spatial context. These features are then fed into a ViT to consider temporal dependencies. This model outperforms other baseline models

for spatial-temporal crop classification with SITS. However, only a small spatial neighbourhood is used.

Multi-temporal outputs with SITS: The methods for SITS classification mentioned so far only predict a single label map from the time series. This is well suited for applications like crop monitoring, but other applications such as the monitoring of environmental changes rely on spatio-temporal classifications. One example is (Otto et al., 2024), where different spatio-temporal data sources, including land use information, are used to model pollution from fine particulate matter over a time span of six years. In the field of SITS classification there are only a few methods that generate multi-temporal output label maps. Yuan et al. (2022) propose the *SITS-former*, a model for Sentinel-2 time series classification. The *SITS-former* applies 3D convolutions to extract spatio-spectral features for each timestep, which serve as input to a Transformer encoder. The input regions have a size of 5×5 pixels, which reduces the spatial context to a small local neighbourhood. Yuan et al. (2022) pre-train the model in a self-supervised way by masking out the input values from randomly selected timesteps that shall be predicted during the pre-training phase. Whereas the output of the pre-trained model is multi-temporal, the authors combine them to one output map for the application of crop classification. Zhao et al. (2023b) introduce a purely attention-based model for active fire detection by combining a Transformer encoder with a MLP head. The input consists of a time series of pixels which are classified as *fire* or *non-fire* for each timestep. The results of their experiments show that the temporal information is more important than the spatial one, which is expected for such an application. Stucker et al. (2023) adapt the U-TAE from Garnot and Landrieu (2021) for sequence-to-sequence cloud removal based on SITS, which is a regression task. In contrast to Garnot and Landrieu (2021), who aggregate the temporal dimension to one output map, the number of timesteps remains the same in (Stucker et al., 2023). The features extracted by the L-TAE module are used as weights, which is again realised by an element-wise matrix multiplication, that guide the model to determine the timesteps that contain the most important information. While spatial context is considered at all spatial resolutions, the temporal attention is only computed in the coarsest resolution, which can prevent the resultant features from representing fine details.

To sum up, SITS are a complex data source, combining spectral, temporal, and spatial information. Choosing suitable methods to extract features from the different dimensions can significantly improve the performance of classification tasks using SITS. One challenge is the increase in computational complexity caused by the higher dimensionality of SITS. Most approaches solve this by separating the computations in different dimensions, but, it remains open which methods are most suitable for which dimension and if joint feature extraction can significantly improve the results. Many approaches utilise limited spatial context, i.e. only single pixels or a few neighbouring pixels, but, for most applications related to SITS, considering spatial context in larger regions is important to achieve good classification performance.

3.3.3 Temporal and spatial position encoding

In Transformer models, self-attention is used to compute similarity scores between the input feature vectors. This computation takes place between all possible pairs of input vectors. This has the

advantage that the model is not biased towards neighbouring observations. On the other hand, the model does not have any information about the ordering in the input sequence. A commonly applied strategy to counteract this drawback is the usage of a positional encoding that is added to the features either in or before the self-attention layers. Such a positional encoding can be constant or include learnable parameters. Vaswani et al. (2017) use a fixed positional encoding with sine and cosine functions of different frequencies to encode the positions of the feature vectors in the input sequence (cf. section 2.2.2.2). It is added element-wise to the input embeddings by simply computing the sum. This type of encoding performs similarly well to a learned encoding in the experiments in (Vaswani et al., 2017). Many approaches working with image or SITS data adopt the positional encodings from the NLP field to the spatial and temporal dimensions. While in all cases the usage of some type of spatial or temporal position encoding improves the classification performance, the type of encoding is less critical, because most approaches seem to perform similarly well (Dosovitskiy et al., 2021; Liu et al., 2021; Voelsen et al., 2024). The most relevant approaches applied to SITS classification are discussed in the following.

Most approaches for image or pixel-wise classification adopt the position encodings from the field of NLP. Dosovitskiy et al. (2021) compare a learnable 1D spatial position encoding with a learnable 2D spatial position encoding for the task of image classification. While the 1D encoding is based on the patch order in the input image, meaning that after the last patch in the first row the first patch in the second row follows, the 2D embedding learns two encodings for the two spatial dimensions of the input image. In their experiments, the 1D spatial encoding performs best, but only marginally better than both other variants, while there is an overall improvement of 2.5% compared to the variant using no spatial position encoding at all. Besides the actual strategy to compute the encoding, another crucial consideration is the integration of the encoding into the Transformer model. Dosovitskiy et al. (2021) compare the common strategy to add the position encoding to the features of the patch embedding with the strategy to add the encodings at the beginning of each self-attention layer. In their experiments, the common strategy works best, but only by a small margin. Liu et al. (2021) use a relative position encoding that is added to the attention matrix (QK^T) in each self-attention layer. This spatial position encoding is learned and based on the relative 2D position, realised by a matrix with similar size as QK^T, of the patches in each window. In their experiments, this type of encoding outperforms model variants with no or an absolute spatial position encoding. Several other works simply adopt positional encodings but do not focus on analysing several variants (Strudel et al., 2021; Zhang et al., 2023a). However, approaches that compare several variants of positional encodings agree that the general usage of an encoding is of higher importance than the type that is used.

For SITS classification not only the spatial order of image patches can contain important information, but also the temporal one. Additionally, temporal sequences often have irregular temporal intervals that vary in between different input timeseries. Therefore, temporal position encodings (TEs) commonly include some information about the acquisition date, encoded in different ways. Yuan et al. (2022), Zhang et al. (2023a) and Garnot et al. (2020) adopt the position encoding from Vaswani et al. (2017) to a TE. Yuan et al. (2022) use the doy of year (DOY) as the input to the (co)sine function; this type of TE improves the performance of their model compared to a learned TE. Zhang et al. (2023a) encode the timestep in the input timeseries with the (co)sine

function but do not further analyse the impact of the encoding. Garnot et al. (2020) adapt the position encoding from Vaswani et al. (2017) to temporal positions based on the number of days since the first acquisition date that is available in the time series and integrate this encoding in their temporal auto-encoder module to classify crop types based on SITS. Tarasiou et al. (2023) use a learned TE that is added to the patch embedding. They create a lookup-table in which the learned positional encoding for each acquisition date in the training dataset is saved. One drawback of this encoding is that only those dates that are available in the training data are encoded, whereas others need to be inter- or extrapolated. Nevertheless, this type of TE significantly improves the models performance compared to the same model without a TE, but no other encodings are tested in their experiments. Voelsen et al. (2024) compare a TE based on the DOY with an encoding based on the DOY and year of acquisition. In their experiments, the encoding that additionally uses the acquisition year slightly outperforms the other variant and a model without any positional encoding. However, similar to the approach of Tarasiou et al. (2023), the usage of the acquisition year limits the model to learn the temporal characteristics of the years for which training data are available.

To conclude, approaches in the field of SITS classification agree that using a TE significantly improves the models performance. The type of the encoding seems to be less critical, as there is no tendency towards one encoding type. A fixed encoding can be useful to incoperate the same temporal information for repeating temporal intervals, e.g. for the same day in different years. On the other hand, a learnable encoding can adapt to the specific data characteristics, but, it is dependent on the training data, i.e. temporal sequences not inlcuded in the training data might be more difficult to classify. Therefore, the selection of a specific TE is always closely related to the application and the data. For instance, it might be more important for a task such as crop classification to include information about the DOY, encoding the season, while for an application like deforestation mapping, the days since a specific starting date can be more relevant.

3.3.4 Computational complexity

Self-attention, as introduced by Vaswani et al. (2017), is computed between all combinations of input feature vectors. Therefore, the computational complexity of the self-attention layer directly depends on the length of the input sequence. This results in limitations regarding the model or input size when the available GPU resources cannot handle the data anymore. For the application of pixel-wise classification, the patch size, i.e. number of pixels that are combined in the patch generation process, is one important parameter that can be changed to adjust the number of input vectors to the self-attention layers, but, if more pixels are combined into one patch, i.e. if the number of input features to the self-attention layers is reduced, the classification performance commonly will decrease. To counteract this limitation, several approaches to decrease the computational complexity have been introduced, which can be roughly categorised into two groups. Approaches belonging to the first group decrease the length of the input sequence by selecting only a defined number of input features vectors. Approaches in the second group use the complete input sequence, but reduce the computational complexity in the self-attention layer itself. The most relevant approaches related to this thesis are discussed in the following.

Several commonly used architectures apply strategies to reduce the length of the input sequence and, therefore, belong to the first category. The Swin Transformer is one of the most widely used Transformer models for pixel-wise classification (Liu et al., 2021). The main contribution of the Swin Transformer is the sliding window approach, computing spatial attention only in local windows of a defined number of patches. These two strategies significantly reduce the number of input patches to the self-attention layer, though, at the cost of losing some global context, which can only be considered in the deeper layers of the model having a coarse spatial resolution. Wang et al. (2021) introduce a Pyramid Vision Transformer for pixel-wise classification. Their main architecture is similar to the Swin Transformer, including patch merging layers to obtain a hierarchical feature representation and self-attention layers to capture spatial dependencies at different spatial resolutions. To reduce the computational complexity within the self-attention layers, the key and value features of r patches are merged and projected to a lower dimension. In this way, the computational complexity is reduced by a factor of r^2 compared to the normal multi-head self-attention block. Compared to the Swin Transformer, in which self-attention is computed in local windows, the strategy from Wang et al. (2021) computes global dependencies between all merged patches in the image, but, at a coarser spatial resolution due to the merging process. Both approaches achieve better performance than other baselines in their experiments, but it may depend on the application whether local or global context is more important. Regarding the classification of image time series, the separate consideration of spatial and temporal dependencies is a common strategy. For instance, Arnab et al. (2021) use a ViT for video scene classification and compare different variants for capturing spatial and temporal context. The best performing variant is the one in which self-attention is computed between all (spatial and temporal) input feature vectors, but it also requires the highest computational costs. When the computation is separated (first spatial, then temporal), the quality of the results decreases only slightly, but the computational effort is reduced to 60%. These results might be transferable to applications in the RS domain, but, to the best of our knowledge, no existing approach working with SITS compares the performance of a model which separates spatial and temporal computations to a model that jointly captures spatio-temporal context.

Approaches belonging to the second group try to reduce the computational complexity within the self-attention layer. Garnot and Landrieu (2020) introduce the L-TAE module that combines several strategies to reduce the computational complexity (cf. section 2.2.2.3). Their main contributions are the direct usage of the input features as value matrix, the definition of the query matrix as a model parameter, and splitting the input features directly into different groups that serve as input to the individual heads instead of learning a linear mapping from the input. Zhang and Yan (2023) introduce a router mechanism that first gathers information from all input dimensions by using a query with highly reduced dimension. Afterwards, the aggregated features transformed to feature with higher dimensionality again by using a key and a value matrix with reduced dimensions. Zaheer et al. (2020) reduce the number of computed dependencies in the self-attention matrix $A = QK^T$ by only computing selected values from A instead of computing the complete matrix A. This is done by combining random attention, window attention, and global attention. In random attention, only a defined portion of randomly selected elements of A are computed, while in window attention the values in A related to neighbouring feature vectors are computed and in global attention, the

values in A that correspond to features within the same row or column are computed. With this approach, Zaheer et al. (2020) can process eight times more input feature vectors compared to the standard approach while still achieving state-of-the-art performance.

One approach that is related to both categories is the architecture proposed by Ding et al. (2024) for real-time semantic segmentation of remote sensing images, e.g. for the usage on drones. They focus on reducing the computational complexity by two contributions: First, they introduce a selective scanning approach that enlarges the receptive field of the model without increasing the computational complexity. This is done by scanning the image in the row and column of the current image patch and only computing spatial context between these patches. In the architecture of Ding et al. (2024) these features are extracted by convolutional layers, however, the selective scanning approach could easily be transferred to self-attention layers. Second, a Transformer branch is only used in the training process by using loss functions that are based on the features from the convolutional and self-attention layers. The goal is that the Tranformer branch supports the main architecture to find suitable weights by focussing on the most relevant features. During inference, the learned weights from the encoder and decoder can be used without the Transformer branch, which significantly reduces the inference time. With these adaptations, Ding et al. (2024) achieve state-of-the-art performance on two of the three used datasets, while processing the images in real-time with a lightweight architecture.

To summarise, there are various approaches for reducing the computational complexity without compromising performance, i.e. by decreasing the number of input vectors to a self-attention layer or by modifying the self-attention layer itself in a way that the number of computations is reduced. Several methods are applied to SITS data, but, to the best of our knowledge, no direct comparison of the strategies to reduce the computational complexity has been conducted so far. Another interesting aspect that is only covered by Arnab et al. (2021) for video scene classification is a comparison to models in which the computational complexity is not reduced, e.g. by separating the temporal and spatial dimensions. Such an analysis with SITS could evaluate to what degree the mentioned modifications have an impact on the performance for SITS classification tasks.

3.4 Training data for remote sensing applications

One challenge in the field of remote sensing is the available training data. While satellite images are available in large amounts, the generation of accurate training labels is time-consuming and expensive, and the variety of applications and sensors results in different requirements for the annotations in terms of expert knowledge (An et al., 2024). This challenge results in many task-specific remote sensing datasets that are commonly limited in size, and several strategies try to compensate for the lack of training data for DL models.

One strategy that is often followed in RS is the usage of existing datasets, e.g. from online maps (Kaiser et al., 2017) or topographic databases (Maas et al., 2018; Yang et al., 2021; Voelsen et al., 2024). The training labels are generated by rasterising the available information from the database and combining them with corresponding aerial or satellite imagery. This process leads to large

training datasets; however, commonly, they include some wrong assignments in the labels, which is referred to as *label noise* in literature. If a dataset includes *label noise*, this can have an impact on the training process and, consequently, on the classification performance. If *label noise* is known to be included in a dataset, its impact can be reduced by several strategies; see Song et al. (2023) for a detailed overview of this research field. Topographic databases are used as training labels in several works with the goal to use the classifier later to update the existing outdated databases. Maas et al. (2018) integrate a noise model directly in a random forest classifier to cope with the *label noise*. Other approaches use datasets including *label noise* without strategies to mitigate its impact (Yang et al., 2021; Voelsen et al., 2024). Kaiser et al. (2017) use a FCN for the semantic segmentation of roads and buildings and generate a large training dataset based on free available online maps that helps to improve the classification accuracy when used for pre-training. In their experiments, they analyse the impact of a large training dataset with *label noise* and come to the conclusion that a large dataset, even if it includes some errors, nevertheless helps to improve the classification accuracy, compared to models trained with just a small dataset without any errors. When DNNs are used as classifiers, the advantage is that commonly included strategies for regularisation, such as data augmentation, dropout or batch normalisation (cf. section 2.1.4.6), also help to mitigate the effect of *label noise* because they help to reduce the model's capacity to learn the structure of the noise.

To summarise, the challenge of acquiring sufficient amounts of training data for RS applications remains due to the varying requirements of different applications and used sensors. While strategies such as using automatically generated datasets can mitigate the effect of limited training data, this approach typically leads to *label noise* that can impact the model's performance. Other strategies to compensate for limited training labels include pre-training on different larger training datasets or training based on self-supervision. Due to the diversity of remote sensing datasets and applications, no solution fits all needs. In the context of this thesis, a dataset based on training labels derived from an existing topographic database is used. However, the impact of *label noise* is not the focus of this thesis but can be an aspect to be analysed in future research.

3.5 Discussion

SITS are a complex data source, combining two spatial, one spectral, and one temporal dimension. Methods that are applied to this type of data need to extract meaningful features from these dimensions individually but also should capture inter-dependencies between them (Miller et al., 2024). In itself, this is a challenging task, but several further aspects need to be considered. One aspect is the computational complexity that can drastically increase when using four-dimensional SITS data. This is closely related to the generation of patches that are commonly used to reduce the computational complexity in the beginning of the model but comes with the cost of loosing some finer spatial structures. Another aspect is a suitable receptive field that is appropriate for the pixel size in object space (GSD). For SITS, the GSD is commonly coarser compared to aerial imagery that is used in many approaches. Finally, most approaches focus on the generation of one output map, e.g. for the task of crop classification, and only some works also generate multi-temporal output maps. In the following, these aspects are discussed in detail, thus defining the research gap

to be tackled in this thesis. The chapter concludes with a discussion of the works that are most closely related to the presented one, focusing on the differences to them.

Capturing spatial and temporal context: Recent methods for SITS classification focus on Transformer models. Especially for the extraction of temporal dependencies, current approaches concur that self-attention outperforms other methods such as CNNs or RNNs (Garnot and Landrieu, 2021; Rußwurm and Körner, 2020; Li et al., 2022a; Voelsen et al., 2024). The main advantage of self-attention is that global and local context can be considered within one self-attention layer, while the temporal encoding helps to include information about the temporal ordering and irregular temporal intervals. Regarding spatial computations, convolutions are successfully used in many works (Ji et al., 2018; Stucker et al., 2023; Garnot and Landrieu, 2021; Zhang et al., 2023a; Yuan et al., 2022), but recent approaches also use self-attention in these dimensions and achieve good performance (Tarasiou et al., 2023; Zhang et al., 2022b; Xu et al., 2021). While convolutions extract local context in the spatial neighbourhood of a pixel, self-attention can potentially also consider global context. In practice, the global view is commonly restricted, e.g. to a window (Liu et al., 2021), due to the computational complexity. Some works try to integrate at least some global context, e.g. by random attention (Zaheer et al., 2020) or by shifting the generated windows (Liu et al., 2021). However, in most models, the receptive field enlarges only with deeper hierarchy levels that have a coarser spatial resolution. This might be one reason that the majority of approaches for pixel-wise classification of SITS uses convolutions in some part of the model, at the latest in the decoder part to obtain feature maps with the original spatial resolution. Therefore, it is still an open research question whether self-attention or convolutional layers are better suited to incoorperate spatial context, and it might depend on the application which method performs better on SITS.

Most approaches for SITS classification have in common that they consider spatial and temporal context separately, e.g. after each other (Tarasiou et al., 2023; Garnot and Landrieu, 2021) or in parallel (Voelsen et al., 2024), mainly due to the increase in computational complexity when all feature vectors serve as inputs to a self-attention layer. The separation has the advantage that different methods can be used in different dimensions that are well suited for the individual characteristics of the data. This might be the main reason why hybrid approaches commonly result in improved performance compared to approaches relying on a single method for applications related to SITS classification. However, existing approaches employing hybrid strategies are limited with respect to the spatial receptive field (Yuan et al., 2022; Zhao et al., 2023b; Li et al., 2022a; Zhang et al., 2023a) or by exploiting temporal dependencies only at a very coarse spatial resolution (Stucker et al., 2023; Garnot and Landrieu, 2021). Especially for tasks like LC classification both, spatial and temporal context are of equal importance, and a model that exploits both types of context by using suitable methods for each dimension is expected to achieve a better performance compared to models with restricted receptive field or solely relying on a specific method. Therefore, the investigation of hybrid models for SITS classification, separating the extraction of spatial and temporal context and using well suited methods for the different characteristics of the data, across varying spatial resolutions, represents a research gap.

To close this research gap, this thesis proposes a hybrid module in which spatial and temporal dependencies are modelled in separated streams, followed by a fusion layer. For encoding temporal

dependencies, self-attention layers are used, as recent research agrees that this method outperforms other methods in the temporal dimension. Spatial context is extracted by convolutions in the new model. Convolutions capture context from the local spatial neighbourhood, which is expected to be well suited for the coarser spatial resolution (3 and 10 m) of the satellite datasets that are used. However, the receptive field enlarges with deeper model layers because a hierarchical encoder-decoder architecture is used. To the best of our knowledge, none of the existing methods adequately addresses this task, which combines larger spatial and temporal receptive fields while the computational complexity is still manageable due to the separate computations in the spatial and temporal dimensions.

Another aspect in SITS classification is the receptive field of a model, which is, among others, defined by the depth of the model, i.e. the number of downsampling layers, in relation to the GSD of the dataset. A receptive field of 10×10 pixels would cover an area of 10 m×10 m in object space for a GSD of 1 m, but, for another sensor with a GSD of 10 m, the same model would cover 100 m×100 m in object space. This example shows that the use of a different dataset in the same model can lead to an input area of completely different size, impacting the model's results. The receptive field in object space should, therefore, always be selected based on the application and the dataset, e.g., by selecting it in relation to the expected size of the objects to be classified. However, none of the approaches discussed in this chapter is analysing a suitable receptive field of the model in relation to the used datasets. In this thesis, this research gap is addressed by performing a larger number of ablation studies to find the appropriate receptive field for the respective dataset.

Generation of multi-temporal output maps: Most approaches that are discussed in section 3.3.2 deal with tasks for which the generation of a mono-temporal output map is sufficient, e.g. for the classification of crop types. For other applications, including monitoring of processes of LC, it is important to predict output maps for several timesteps, making it possible to analyse the temporal change of LC. Only a few approaches follow this strategy in the field of SITS classification. Yuan et al. (2022) introduce the SITS-Former that is pre-trained in an unsupervised way, whereas Stucker et al. (2023) combine a U-Net with temporal self-attention for cloud removal from SITS. However, none of them deals with multi-temporal LC classification. The method proposed in this thesis fills this research gap by introducing a new architecture for SITS classification that predicts one output map for each input timestep. Spatial and temporal dependencies are extracted in all model stages, helping the model to learn dependencies in these dimensions. To the best of our knowledge, this is the first time this is done for LC classification with SITS.

A further aspect related to the generation of multi-temporal output maps is the question of how the temporal dimension is kept and used throughout all model layers. Stucker et al. (2023) apply convolutions separately in all encoder and decoder layers, and a connection between the different timesteps is integrated by the L-TAE module, which is used in the coarsest spatial resolution of the model and upsampled to use the extracted temporal dependencies as weights in the skip connections. However, none of the existing approaches related to SITS classification investigates how temporal features, extracted at all hierarchy levels of an encoder-decoder architecture, can be integrated into the later decoder layers. Therefore, in this thesis, the approach to use temporal features to weight spatial features introduced by Garnot and Landrieu (2020) is extended to all

spatial resolutions instead of only using it in the bottleneck layer. In this way, it is possible to consider temporal dependencies in different spatial resolutions, which is important because in deeper layers the extracted spatial features summarise the characteristics of a larger receptive field, while in the first layers, more simple features are computed based on a relatively small receptive field.

Patch generation: One of the challenges in pixel-wise classification is the large number of pixels in comparison to other applications, e.g. to temporal sequences. In the Transformer models that have been adapted for image classification, a patch embedding module is used at the beginning to combine several pixels into one feature vector (Dosovitskiy et al., 2021). This idea is also used in most models applied for pixel-wise classification (Strudel et al., 2021; Liu et al., 2021). In the Swin Transformer, the sliding window approach makes it possible to reduce the number of pixels that are combined in one patch, but still commonly 16 ($4 \cdot 4$) or 4 ($2 \cdot 2$) pixels are combined in this step. With the coarser GSD of satellite imagery this results in already large areas that are covered by only one patch, e.g. 40×40 m for Sentinel-2 images with 10 m GSD. The whole process of patch generation, therefore, results in difficulties in predicting the correct position of class boundaries or, for the application of LC classification, in correctly classifying fine structures such as streets (Strudel et al., 2021). Some strategies have been introduced to counteract this difficulty, e.g. by using skip connections to include information about the exact position from the early layers into the later layers of the model, but, they can only partly mitigate this problem. Several approaches which are not based on Transformers only apply convolutional layers at the original spatial resolution of the input images in SITS classification (Stucker et al., 2023; Yuan et al., 2022). However, to the best of our knowledge, no existing approach in the field of SITS classification directly replaces the patch embedding module, as used in (Dosovitskiy et al., 2021) or (Liu et al., 2021), by a convolutional embedding. In this work, this research gap is addressed by introducing a convolutional patch embedding module that replaces the standard one. This module combines 3D convolutional layers with max-pooling layers to downsample the spatial resolution. Therefore, spatial and temporal dependencies are already extracted at a higher spatial resolution compared to models using the standard patch embedding module. Furthermore, the extracted features can be used in additional skip connections, serving as additional features to the last layers of the decoder.

3.5.1 Comparison to the most similar works

To highlight the contribution of this thesis, the most similar works from literature are discussed in detail with a focus on highlighting the similarities as well as the main differences between them and the proposed method. Most relevant are the works of Stucker et al. (2023) and Yuan et al. (2022) because, similarly to the method proposed in this thesis, these approaches are based on SITS and follow different strategies to consider spatial and temporal dependencies. Furthermore, both approaches predict multi-temporal output maps, which is rarely the case in approaches related to SITS classification.

The Utilise model, introduced by Stucker et al. (2023) and explained in detail in section 2.2.2.4, is one of the approaches most similar to the method introduced in this thesis. One significant

difference is the application for which the model is developed, because Utilise solves the task of cloud removal from SITS and therefore the generated output maps consist of pixel intensity values, which corresponds to a regression task. Nevertheless, the model is easily adjustable to classification tasks by modifying the last layer of the model. Utilise uses convolutional layers separately for all input timesteps in a hierarchical encoder with four stages to extract features encoding spatial context. In the bottleneck layer, a lightweight temporal-attention encoder (L-TAE, cf. section 2.2.2.3) is used to capture temporal context. The main reason to do so only at the coarsest spatial resolution level is a reduction of the computational complexity due to the lower number of patches at that level. This is the first main difference to the model proposed in this thesis, which performs computations in the spatial and temporal dimensions at all hierarchy levels in the model. Stucker et al. (2023) argue that it is sufficient to consider temporal dependencies at the coarsest spatial resoltion for the task of cloud removal, but it is unclear if this assumption is true and holds for different applications including land cover classification, as Stucker et al. (2023) do not compare their strategy to a baseline model that extracts temporal context at other spatial resolutions. Similar to the encoder, convolutional layers are used separately for all timesteps in the decoder part of the model, and they are combined with upsampling layers to obtain the original spatial resolution at the last stage of the model. This results in an output map for each input timestep and is similar to the model proposed in this thesis. The Utilise model is relatively lightweight, i.e. the setting used by Stucker et al. (2023) results in only one million trainable parameters, and the combination of convolution and self-attention results in good performance. The main disadvantage is the late computation of temporal context at the coarsest spatial resolution, which raises the question if that is sufficient, especially for the task of LC classification. In the experiments conducted in this thesis, the Utilise model is used for comparison purposes to investigate the model's performance as well as the computational complexity in comparison to the method proposed in this thesis.

Yuan et al. (2022) propose the SITS-Former, a Transformer model for SITS classification. They pre-train it in an unsupervised manner on a large Sentinel-2 dataset by predicting missing patches in the timeseries. Similarly to the method proposed in this thesis, the authors use a temporal encoding and multi-head self-attention layers as in the original Transformer model (Vaswani et al., 2017). The input to the SITS-Former is a temporal sequence of patches with a spatial extent of 5×5 pixels, which drastically limits the spatial context that can be considered. In the patch embedding of the SITS-former, 3D convolutions are used to extract spatial-spectral features from all timesteps in parallel, and feature vectors of a defined dimension are obtained, which serve as inputs to the Transformer encoder. Therefore, during patch embedding, only spatial and spectral context is encoded, while the encoder solely focuses on extracting temporal dependencies. The generated output is multi-temporal, but for crop classification, which is the goal of Yuan et al. (2022), the output is fused to one output map again. A large difference compared to the method proposed in this thesis is the usage of a large unlabelled SITS dataset for unsupervised pre-training. This is a promising direction for future research when only a small labelled dataset is available. However, in this thesis, a comparably large SITS dataset is available, and self-supervised learning is not the focus of this thesis. Another difference to the method proposed in this thesis is that spatial and temporal dependencies are considered one after the other, but in combination with a very small spatial receptive field as no spatial context beyond the borders of the 5×5 pixels input

patches is considered. However, Yuan et al. (2022) do not investigate if the limited spatial context has a negative impact on the classification performance. In contrast to the SITS-Former, the model introduced in this thesis considers spatial and temporal context in all stages of the model and the spatial receptive field is not restricted to a small window. Additionally, the 3D patch embedding captures spatial and temporal context already at the original spatial resolution, which has the advantage that the extracted features can be integrated into the decoder layers to increase the performance for object boundaries or the classification of fine details.

To summarise, in this thesis, a new model for multi-temporal SITS classification is introduced. It is based on a hybrid module that considers spatial context by using convolutions and temporal context using self-attention. On the one hand, this separation of feature computation results in a less complex structure and, on the other hand, it is expected that convolutions are better suited to capture spatial context. For the consideration of temporal context, self-attention is used. Furthermore, the model generates multi-temporal output maps, making it possible to predict different classes for the same pixel over time, which makes it possible to use the predictions for monitoring land cover processes. Furthermore, the patch embedding from the original Swin model (Liu et al., 2021) is replaced by a patch embedding module based on 3D convolutions. In this way it is possible to extract spatio-temporal features at the original resolution of the input images before the spatial resolution is reduced. An additional contribution is the temporal weighting, i.e. multiplication of the features from the temporal and spatial streams in the skip connections of all model stages. In this way, features encoding spatial and temporal context at all hierarchy levels are integrated into the decoder layers of the model which shall help to predict more accurate class boundaries.

4 A new method for SITS classification

In this chapter, the new method for multi-temporal pixel-wise classification of SITS is introduced. The prerequisites for the proposed method are presented in section 4.1. The general overview of the model is given in section 4.2.1 followed by a detailed description of the encoder (section 4.2.2) and decoder architectures (section 4.2.3). Afterwards, the individual modules used in the new architecture are described in sections 4.2.4 - 4.2.7, including the spatio-temporal patch generation module, the temporal position encoding, the hybrid feature extraction module, which encodes the spatio-temporal dependencies, and the temporal weighting module. Section 4.2.6 additionally includes two modules for encoding spatial and temporal features that are both solely based on self-attention layers and serve as modules for comparison in the experiments. The training procedure is described in section 4.3.

4.1 Prerequisites

The proposed method requires multi-temporal input images $X \in \mathbb{R}^{T \times B \times H_0 \times W_0}$, where T represents the number of timesteps, B the number of spectral bands, and H_0 and W_0 the height and width of the input images, respectively. The model predicts multi-temporal maps $\hat{Y} \in \mathbb{R}^{T \times H_0 \times W_0}$. In training they are compared to multi-temporal labels maps $Y \in \mathbb{R}^{T \times H_0 \times W_0}$ for computing the loss. In general, the number of used timesteps, spectral bands and the input height and width can be selected freely, but have to be identical in the training and inference processes. It is further assumed that the GSD stays constant during the training and inferencing process. In the context of this thesis, the used time series always covers a time period of one calendar year (January to December). This has the advantage that a whole vegetation cycle is covered, which has been found to be beneficial for the classification performance, especially for vegetation or crop classes when SITS are used (Ji et al., 2018; Rußwurm and Körner, 2020). The number of timesteps T is defined as a hyper-parameter that stays constant during training and inference. The process to generate the image timeseries X during training and inference is described in detail in section 4.1.1 In the context of this thesis, the proposed method is applied and evaluated for the task of LC classification. In general, the model can also be used for different applications in the field of remote sensing, e.g. crop mapping or deforestation detection, under the condition that multi-temporal images and label maps are available.

4.1.1 Generation of image timeseries

The number of timesteps T used during training and inference is set as a constant model parameter. As discussed before, one timeseries used as input always covers one calendar year (January to De-

cember). To create the input X, the calendar year is split into T time intervals with approximately the same number of days each. Afterwards, for each time interval, one representative satellite image is chosen based on a strategy that differs between training and evaluation. An overview of the strategy to select images that are combined to one image timeseries X is shown in figure 4.1.

During training, the representative image for each time interval is chosen randomly from all available images in that interval. In this way, the training dataset is even larger compared to a strategy using the same acquisition date every time. For this purpose, in the training process, first, an area is selected randomly from the available training data. Afterwards, one image is selected randomly for each temporal interval from all acquisition dates available for the selected area. This is visualised by the blue rectangles in figure 4.1. As a last step, random data augmentation, including rotations by 90°, 180°, 270° and horizontal and vertical flipping is applied on the complete image timeseries, which means that the same augmentations are applied to all images in the same timeseries. In the end, the selected images showing the same area but for different timesteps within the temporal intervals are combined to form the input X. In this way, even if the same area is chosen multiple times for training, the used images can still be different, which increases the variability of the whole training dataset. In previous experiments, this strategy performed better than always using the same image for each temporal interval, probably due to the larger variability of the appearance of objects.

The validation and test dataset are used to monitor the performance of the classifier during the training process or to compare different model variants after the training process finished. Therefore, validation and testing have to be based on the same image timeseries every time the validation or test accuracies are computed. To generate the test dataset $\mathcal{T}_{te} = \{X_j, Y_j\}_{j=1}^{n_{\mathcal{T}_{te}}}$, a number of $n_{\mathcal{T}_{te}}$ image patches are selected from the available dataset $\mathcal{T}$. For each temporal interval $t \in \{1, ..., T\}$, the image that is acquired most closely in time to the middle of the interval is chosen, visualised by the orange rectangles in figure 4.1. In this way, the intervals between the acquisition times of the images in one input SITS are as similar to each other as possible, and the evaluation is based on the same images for all experiments. This procedure is also applied for the validation selection. It is worth noting that care is taken for the training, validation and test datasets not to overlap.

4.2 Model architecture

This thesis presents a new model for spatio-temporal SITS classification at pixel level. The proposed model integrates a new patch generation module at the model's input stage, followed by hybrid feature extraction modules that simultaneously capture spatial and temporal dependencies. The patch generation module employs 3D convolutions that extract spatio-temporal features by shifting a 3D kernel matrix in the spatial and temporal dimensions of the input X. The hybrid feature extraction module uses self-attention to consider temporal context and separate 2D convolutions to capture spatial dependencies. Both modules are integrated into a multi-temporal encoder-decoder framework. By combining the strength of self-attention, convolution and feature extraction already at the original spatial resolution, this model is expected to improve the classification performance

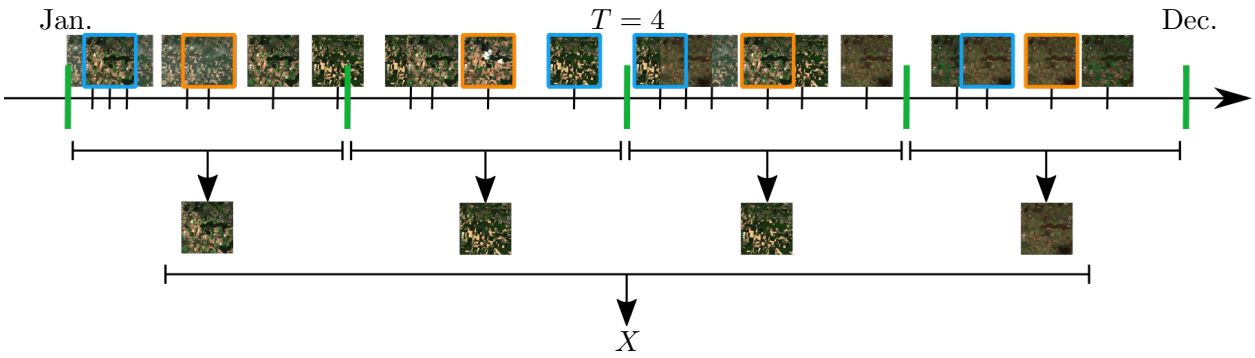

Figure 4.1: Visualisation of the strategy to generate the input timeseries X for a number of timesteps $T = 4$, i.e. each time interval would cover approximately three months. Blue rectangles visualise the random selection strategy used during the training process. The orange rectangles visualise the selection of the images that are acquired most closely in time to the middle of the corresponding time interval, which is the strategy used in the inference process.

compared to approaches using the standard patch generation or solely relying on spatial or temporal context.

4.2.1 Overview

The new model for SITS classification is based on an encoder-decoder architecture and is shown in figure 4.2. In the encoder, the input X is processed by the new 3D patch generation (3D-PG) module that reduces the spatial dimension to $(H_0/P \times W_0/P)$, before temporal position encoding (TE) is added. Afterwards, a number S of stages follows. In the individual stages $s \in \{1, ..., S\}$, the new hybrid feature extraction module FE_{hyb} is used L_s times to extract feature encoding spatial and temporal context in separate streams. The 3D-PG and the FE_{hyb} modules are the main contributions of this thesis. Between subsequently applied stages, the spatial resolution is reduced by a factor of two using a patch merging layer (cf. section 2.2.2.4).

The hierarchical encoder can be combined with any hierarchical decoder under the condition that the decoder is able to handle multi-temporal images too and predict maps for every timestep. The decoder used in this thesis is based on UPer-Net (Xiao et al., 2018); it is extended to handle the additional temporal dimension and is used to generate the pixel-wise predictions maps $\hat{Y}$. Between the encoder and decoder stages of the same spatial resolution, skip connections are used. In these skip connections, the new temporal weighting module (TW) is applied, which uses the features extracted by the temporal and spatial streams of the last FE_{hyb} module of the corresponding encoder stage. One important property of the proposed method is that the number of timesteps T is maintained throughout all encoder and decoder stages. In the following, the general structure of the encoder and decoder is introduced in sections 4.2.2 and 4.2.3, respectively.

4.2.2 Encoder

The encoder follows a hierarchical structure that processes the data at different spatial resolutions and is used, e.g. in the U-Net (Ronneberger et al., 2015) and the Swin Transformer (Liu et al.,

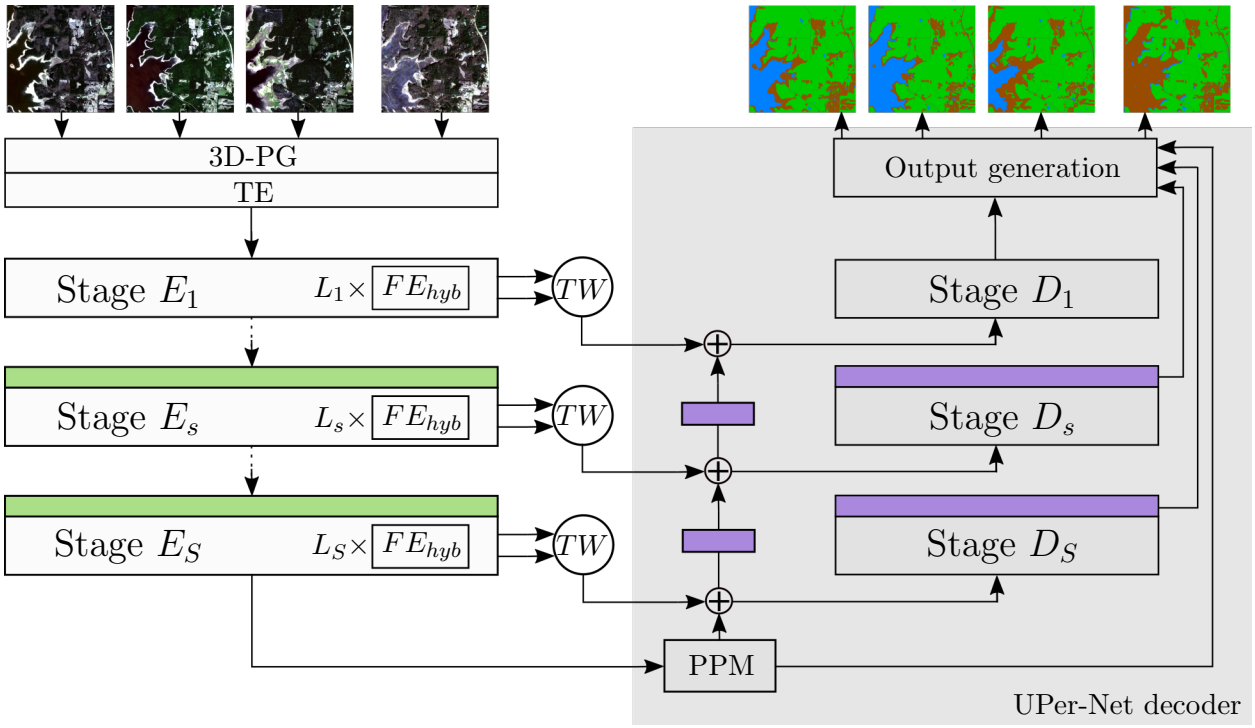

Figure 4.2: General model overview. 3D-PG: patch generation based on 3D convolutions (section 4.2.4). TE: Temporal position encoding. FE_{hyb}: Hybrid feature extraction module (section 4.2.6.1), E_s, D_s: Encoder and decoder stage s with the same spatial dimension, respectively. L_s: Number of sequentially used FE_{hyb} modules for stage E_s. TW: Skip connection with temporal weighting (section 4.2.7), PPM: Pyramid pooling module from UPer-Net. Green layers: Patch merging, violet layers: Upsampling by bilinear interpolation $\oplus$: Element-wise addition of input features.

2021). In this thesis, one hierarchy level is defined as the group of all layers that share the same spatial resolution and is referred to as one stage. The main difference to the mentioned models from literature is the number of timesteps T of the input $X \in \mathbb{R}^{T\times B\times H_0\times W_0}$ that is maintained throughout the whole model. To be able to consider temporal and spatial dependencies in the encoder stages, a new module FE_{hyb} is introduced. In this module spatial dependencies are encoded by convolutions and temporal dependencies are considered by self-attention layers. This is done in separate streams, which significantly reduces the computational complexity. An overview of the encoder is shown on the left side of figure 4.2.

The input X is first processed by the 3D patch generation (3D-PG) module, which is one of the main contributions of this thesis. In this module, spatio-temporal dependencies are encoded by applying 3D convolutions, shifting the kernels in the spatial and temporal dimensions of X at the original spatial resolution. Afterwards, the spatial dimension is reduced to obtain the required image patches that are needed for the following layers. In the 3D-PG module, the number of input bands is gradually increased from B bands of the input X to C_{in}, a feature dimension that is defined as a hyper-parameter. In the context of this thesis, a patch is represented by a feature vector of a certain dimension (C_{in} after applying the 3D-PG module) that contains features from a defined region of the input image. That is, after patch generation, information from $P \times P$ pixels is represented by one feature vector. The region for which a feature vector is representative gets larger in the deeper model stages, i.e. whenever patch merging is applied. The temporal dimension

remains unchanged, resulting in T patches that describe the same region of the input images for the different timesteps. A detailed description of the 3D-PG module is given in section 4.2.4.

Afterwards, a temporal position encoding (TE) is added to the features of all timesteps to include information of the acquisition dates that are used within the timeseries. The temporal position encoding is based on the position encoding from (Vaswani et al., 2017) and adapted to take the acquisition dates of the input images into account. It is described in section 4.2.5. After the patch generation and temporal encoding are applied, the sequence $Z_{emb} \in \mathbb{R}^{T \times C_{in} \times N_1}$ is generated. The number of input patches to the first stage of the model is $N_1 = H_0/P \cdot W_0/P = H_1 \cdot W_1$ for each timestep, defined by the height H_0 and width W_0 of the input images and patch size P. Z_{emb} serves as input to the first stage E_1 of the encoder. Note that, similar to the mono-temporal Swin Transformer, the sequence $Z_{emb} \in \mathbb{R}^{T \times C_{in} \times N_1}$ can be rearranged to a tensor $A_{emb} \in \mathbb{R}^{T \times C_{in} \times H_1 \times W_1}$, which is necessary e.g. for the convolutional layers in the FE_{hyb} module.

The main component of the encoder are the stages E_1 - E_S, with S representing the total number of stages, adapted from mono-temporal encoder-decoder architectures, e.g. the Swin Transformer described in section 2.2.2.4. The output of a stage s is defined by $Z_s \in \mathbb{R}^{T \times C_s \times N_s}$, where C_s denotes the feature dimension, $N_s = H_s \cdot W_s$ the number of patches and (H_s, W_s) denote the height and width of the feature maps in stage s, respectively. The output Z_s serves as input to the patch merging layer that reduces the spatial dimension by a factor of two ($H_{s+1} = H_s/2, W_{s+1} = W_s/2$) before the next stage is applied. During patch merging, a linear layer transforms the features vectors of those patches that are merged into one representative feature vector. In this layer the number of feature maps is doubled compared to the number of features in the previous stage, resulting in a feature dimension of $C_s = C_{in} \cdot 2^{s-1}$ in stage s. In each stage, a certain number L_s of the new hybrid spatio-temporal feature extraction modules FE_{hyb} are applied consecutively, with $l_s \in [1, ..., L_s]$ being the index of the modules in stage s. L_s is set as a hyper-parameter for each stage. In the FE_{hyb} module, spatial context is encoded by convolutional layers, and temporal context is considered based on self-attention. This is done in two separate streams, the outputs of which are combined to form the final output $Z_{s,l}$ of module l in stage s. The details about the FE_{hyb} module are described in section 4.2.6.1. The total number of stages S is a hyperparameter that can be adjusted to change the receptive field and number of trainable parameters of the model. The output of the last encoder stage Z_S serves as input to the deepest layer of the decoder, which is described in section 4.2.3.

4.2.3 Decoder: Multi-temporal UPer-Net

The multi-temporal decoder is an extension of the UPer-Net decoder, introduced by Xiao et al. (2018) for mono-temporal images and explained in section 2.2.1.5. The goal of using a multi-temporal decoder is to maintain the number of timesteps T and obtain the original spatial resolution in the last layer of the model, resulting in the predictions $\hat{Y} \in \mathbb{R}^{T \times H_0 \times W_0}$. Similar to the mono-temporal variant, the multi-temporal UPer-Net consists of several stages that are denoted by D_s. As shown in figure 4.2, the decoder stages are counted backwards starting from S at the deepest stage, to 1 as the last stage (before output generation) of the model. This is done to have the same spatial resolution H_s, W_s in corresponding encoder and decoder stages E_s and D_s. The output of a decoder

stage is denoted by $A_{D_s} \in \mathbb{R}^{T \times C_{dec} \times H_s \times W_s}$, where C_{dec} denotes the number of feature maps that stays constant in all decoder stages. The main difference to the mono-temporal UPer-Net (section 2.2.1.5) is the additional temporal dimension of the feature maps A_{D_s}. Whenever computations, e.g. convolutions or upsampling layers, are applied to A_{D_s}, this is done separately for all timesteps T. Weights are shared across time to not drastically increase the number of trainable parameters and to keep the number of parameters independent from the number of used timesteps. Another adaptation is the separation of the PPM module from the deepest decoder stage D_S. This is done to be able to integrate the information from the skip connections from encoder stage E_S into the decoder, which was not the case in the mono-temporal UPer-Net. Skip connections connect the encoder stages with the features from the decoder. Another novelty is the usage of the temporal weighting module TW in these skip connections. The extracted features from the temporal and spatial streams from the last FE_{hyb} module of the corresponding encoder stage serve as input to this module. In the TW module, the output of the temporal stream are used as weights for the features extracted by the spatial stream, similarly to the strategy applied in the L-TAE introduced by Garnot and Landrieu (2020). Details about the temporal weighting module are presented in section 4.2.7.

The remaining layers of UPer-Net are structured similarly to those in the mono-temporal version described in section 2.2.1.5. In the skip connection $skip_{D_s}$, the feature maps coming from the TW module of the corresponding encoder stage E_s are combined with the feature maps $A_{skip(s+1)}$ from the previous skip connection or, in case of the first stage, from the output A_{PPM} of the PPM module. If the features from the previous skip connection are used, the spatial resolution is upsampled by a factor of two using bilinear interpolation. The features are again combined by element-wise addition, and the obtained output $A_{skip(s)}$ serves as input to decoder stage D_s as well as to the skip connection of stage $s-1$. A stage D_s consists of a convolutional layer with kernel size $k = 3$, BN and an activation function. The activation function is defined in the hyper-parameters studies (cf. section 5.2.1) in which ReLU, lReLU and GELU are tested. For the output generation, the feature maps $A_{D(s)} \forall s \in [1, ..., S]$ are upsampled by a factor 2^{s-1} and the output feature maps A_{PPM} from the PPM are upsampled by 2^S. After these upsampling layers all feature maps have a spatial resolution of $(H_0/P, W_0/P)$. These feature maps are concatenated and transformed to C_{dec} dimensions again by using a convolutional layer with $k = 3$, BN, and an activation function. Afterwards, a last convolutional layer with $k = 1$, BN and activation function maps the features to n_c raw class scores, which are upsampled by the factor of P corresponding to the patch size that is used in the patch generation module, which are normalised by a softmax layer. This results in the final output of the model $A_{out} \in \mathbb{R}^{T \times n_c \times H_0 \times W_0}$. An overview of the decoder with all important parameters is given in table 4.1.

4.2.4 Spatio-temporal patch generation

One contribution of this thesis is the introduction of a new spatio-temporal patch generation module It combines 3D convolutional layers in the two spatial and the temporal dimensions with maximum-pooling layers to reduce the spatial resolution afterwards. In this way, spatio-temporal features are extracted at the original spatial resolution before this dimension is reduced to obtain $A_{emb} \in$

Stage	Operation(s)	Output	Dimension
Bottleneck	PPM(A_S)	A_{PPM}	C_{dec}
$Skip_{D_S}$	$TW(A_S) + A_{PPM}$	$A_{skip(s)}$	C_{dec}
$Skip_{D_{S-1}-D_1}$	$TW(A_s)$+Up($A_{skip(s+1),2}$)	$A_{skip(s)}$	C_{dec}
D_s	CB($A_{skip(s)}$,3)	$A_{D(s)}$	C_{dec}
Output generation	Concat($\forall s$: Up($A_{D(s)}$,2^{s-1}),Up(A_{PPM},2^{S-1}))	A_{CC}	$S \cdot C_{dec} + C_{dec}$
	CB(A_{CC}, 3)	A_{C1}	C_{dec}
	CB(A_{C1}, 1)	A_{C2}	n_c
	Up(A_{C2},P)	A'	n_c
	Softmax(A')	A_{out}	n_c

Table 4.1: Multi-temporal UPer-Net that is used as decoder in the multi-temporal model for SITS classification. PPM: Pyramid pooling module, CB(input,k): convolutional layer with kernel size k, followed by BN and activation function. TW: Temporal weighting module, +: Element-wise addition, Concat: concatenation of listed input feature maps along the feature dimension. Up(input,f): Upsampling of the input by factor f with bilinear interpolation. S: total number of stages.

$\mathbb{R}^{T \times C_{in} \times H_0/P \times W_0/P}$. The new module is referred to as the 3D-PG module and an overview of the module is shown in figure 4.3.

The input to the 3D-PG module is the input image time-series $X \in \mathbb{R}^{T \times B \times H_0 \times W_0}$ and the output, after adding the temporal position encoding, is $A_{emb} \in \mathbb{R}^{T \times C_{in} \times H_0/P \times W_0/P}$. The patch size P serves as the downsampling factor by which the spatial resolution is reduced at the end of the 3D-PG module. Commonly used values for the patch size P in the Swin Transformer are $P = 4$ or $P = 2$, but larger values can also be used. To avoid a drastic reduction in one downsampling layer, the number of downsampling layers depends on the used patch size P and is defined as $P/2$. This means that in every downsampling layer the spatial resolution is reduced by a factor of two, and at each resolution, spatio-temporal context is considered. Due to the downsampling factor of two used in the maximum-pooling layers, the patch size P is restricted to values that can be expressed by $P = 2^i$, where i is a non-negative integer. The feature dimension is increased over the subsequently applied PG blocks. In block $j \in [1, ..., P/2]$ the feature dimension is set to $C_j = j \cdot C_{in}/(P/2)$. A sequence of layers that extract features from feature maps with the same spatial resolution in combination with the following downsampling layer is referred to as one patch-embedding block (indicated by the red rectangle in figure 4.3). In each patch-embedding block a number n_{emb} of 3D convolutions, each followed by BN and an activation function, is applied (blue rectangles in figure 4.3), before a downsampling layer reduces the spatial dimension by a factor of two with maximum-pooling. After $P/2$ patch-embedding blocks have been applied, the temporal position encoding is added and final output $A_{emb} \in \mathbb{R}^{T \times C_{in} \times H_0/P \times W_0/P}$ is obtained. Note that A_{emb} and $Z_{emb} \in \mathbb{R}^{T \times C_{in} \times N_1}$ with $N_1 = H_1 \cdot W_1 = H_0/P \cdot W_0/P$ contain the same feature values, but in different structures (image-like (A_{emb}) vs. sequence-like (Z_{emb})) and can be converted into each other at any time in the processing chain. Both structures are needed in different parts of the modules that are introduced in the following. In order to use a homogeneous notation, the image-like notation will be used in the remainder of this chapter, except for specific equations that need the sequence-like structure as the input.

The patchification is one of the limiting factors of using Transformer models for image classification. Spatio-temporal features are extracted at the original image resolution by using the 3D-PG module, which can potentially help to improve the classifier's performance at object borders or for fine structures. To be able to use these features with a high spatial resolution in the later layers of the model, the generated feature maps before the downsampling layer can serve as inputs to additional skip connections that connect the patch generation module with the last layers of the decoder. These connections are not used in the proposed model, but in another variant to analyse the model's performance with and without this connection (cf. section 5.2.4). The maximum patch size used in this thesis is $P = 4$, which means that two patch generation blocks are used consecutively.

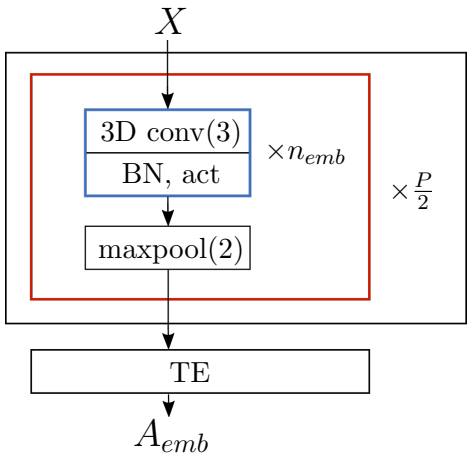

Figure 4.3: General structure of the 3D patch generation module (black rectangle). The red rectangle indicates one patch-embedding block. In each patch-embedding block, a number n_{emb} of 3D convolutions (3D conv(k)) with kernel size k, each followed by BN and activation function, are applied (blue rectangle). maxpool(f): Maximum pooling by factor f, P: Patch size, TE: Temporal position encoding module. $X \in \mathbb{R}^{T \times B \times H_0 \times W_0}$: Input timeseries, $A_{emb} \in \mathbb{R}^{T \times C_{in} \times H_0/P \times W_0/P}$: Output feature maps.

4.2.5 Temporal position encoding

Due to cloud cover, data gaps, or other circumstances, the satellite images that can be used for classification can have irregular temporal intervals. Therefore, the information about acquisition times and temporal differences in the input timeseries can provide additional important information to the classifier. In the temporal position encoding module, which is referred to as TE, the information about all used acquisition times in the used timeseries is provided to the model.

The temporal position encoding that is employed in this thesis is based on the temporal encoding from (Garnot et al., 2020). It employs the acquisition date within a year of the used satellite images:

$$te_{t,i_c} = sin\left(\frac{DOY(t)}{\tau^{\frac{2i_c}{C_{in}}}} + \frac{\pi}{2} mod(i_c, 2)\right), \tag{4.1}$$

with $DOY \in [1, ..., 365]$ being the day of the year the satellite image at timestep t was acquired, $t \in [1, ..., T]$ the current timestep, and $i_c \in [1, ..., C_{in}]$ representing the feature index. The parameter

τ is set to $\tau = 10000$ in all experiments in this thesis. This results in a feature vector $\mathbf{te}_t = [te_{t,1}, ..., te_{t,C_{in}}]$ for every timestep t, which encodes the DOY in a unique way. This also means that for the same day in different years, the temporal encoding is identical. This can help the model to capture seasonal or other temporal effects in a similar way for images acquired at the same time in different years, under the assumption that the seasonal changes are similar in different years. The temporal position encoding is computed for each input timestep and results in $T_{te} = [\mathbf{te}_1, ..., \mathbf{te}_T] \in \mathbb{R}^{T \times C_{in}}$. T_{te} is repeated N_1 times to be able to add the temporal information to every patch, and the result is added to the output of the patch generation module by element-wise addition to integrate the temporal information to every patch. The output after the temporal position encoding is $A_{emb} \in \mathbb{R}^{C_{in} \times T \times H_1 \times W_1}$ with $N_1 = H_1 \cdot W_1$.

4.2.6 Spatio-temporal dependencies

Spatial and temporal dependencies are both crucial for pixel-wise classification of SITS. Therefore, effective methods for the embedding of context in these dimensions is essential to achieve a good classification performance. This thesis introduces a new hybrid spatio-temporal feature extraction module, FE_{hyb}, which is the core component of the proposed model for SITS classification. The new module combines the encoding of spatial dependencies by convolutions with the consideration of temporal dependencies by self-attention. The FE_{hyb} module is described in detail in section 4.2.6.1.

To evaluate the performance of the proposed FE_{hyb} module, two additional feature extraction modules are introduced. Both of these modules are purely based on self-attention. The FE_{att} module independently considers spatial and temporal context in two separate streams, which is similar to the FE_{hyb} module, but the spatial stream is based on self-attention. The third feature extraction module, FE_{full}, applies joint spatio-temporal attention across all three dimensions concurrently by considering patches from all timesteps and all spatial positions simultaneously within one self-attention block. Both of the introduced modules can replace the FE_{hyb} module, which is done in separate model variants, enabling a comparative analysis of the proposed model's performance relative to these alternative approaches. The FE_{att} and the FE_{full} modules are described in section 4.2.6.2 and 4.2.6.3, respectively.

4.2.6.1 The hybrid feature extraction module

The new hybrid module FE_{hyb} that is used to consider spatial and temporal context is introduced with the goal of combining convolutions and self-attention layers to be able to exploit the advantages of both methods. While convolutions are successfully used for many applications in image and pixel-wise classification, self-attention was reported to outperform other methods for sequential data (cf. section 3.3.1). As SITS combine both aspects, it is expected that a hybrid approach is well suited for the application of SITS classification while reducing the number of computations and therefore training time. The FE_{hyb} module is realised by separating the computation of features encoding spatial and temporal context in two streams, which are followed by a fusion module to merge the features of both streams. The principle is illustrated in figure 4.4. In the spatial

stream, convolutions are used to compute features encoding spatial context separately for each of the timesteps. This is indicated by the yellow rectangle in figure 4.4 for one of the timesteps. In the temporal stream, temporal dependencies are considered by self-attention layers, applied separately tor all patches covering the same spatial area. The patches considered simultaneously within one self-attention layer are indicated in red in figure 4.4.

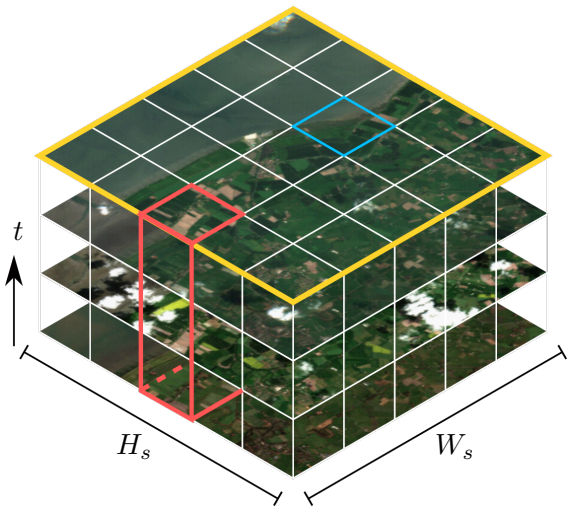

Figure 4.4: Visualization of the principle of feature extraction in the FE_{hyb} module. The blue rectangle indicates one patch, i.e. a certain area of the input image represented by a feature vector of dimension C_s. The area of the input image is represented by the yellow rectangle including the patches of stage s that are all considered in the spatial stream, while the red cuboid indicates the patches considered in one self-attention block in the temporal stream. (H_s, W_s): Height and width of the feature map in stage s, t: Index for timesteps. Note that the satellite images are only used for visualisation purposes, because the input to a FE_{hyb} module commonly consists of the features from the previous FE_{hyb} module or the patch generation module and not of the original spectral input bands.

In stage s, the input to the FE_{hyb} module ($l_s = 1$) is the output $A_{s,l-1}$ of the previous module or, in the case of the first module in stage s, the output $A_{s-1,L_{s-1}}$ of the patch merging module applied at the end of stage $s-1$. For the first stage ($s = 1$), the input to the FE_{hyb} module is the output A_{emb} after the patch generation and temporal encoding modules. For simplicity, only the notation $A_{s,l-1}$ is used in this chapter to denote the input the FE_{hyb} module. The input $A_{s,l-1}$ of the FE_{hyb} module serves as input to both streams of this module. In the spatial stream, the feature map $A^{sp}_{s,l}$ is computed based on convolutional layers. In the temporal stream, the features $A^{te}_{s,l}$ are computed using self-attention. Afterwards, these features are combined to a feature map $A_{s,l}$ in a fusion module, which serves as input to the next module $l+1$, or, in case the last module in the stage is applied ($l_s = L_s$), to the next patch merging module. An overview of the general structure of an encoder stage E_s, consisting of L_s consecutively applied FE_{hyb} modules, is shown in figure 4.5. The output of the spatial and temporal streams of the last module L_s in stage s are also used in the temporal weighting module in the skip connections (section 4.2.7). In the following, the spatial and temporal streams as well as the feature fusion are explained in detail.

Temporal stream: In the temporal stream of the FE_{hyb} module, multi-head self-attention (MSA) blocks, described in section 2.2.2.3, are used to compute features encoding temporal depen-

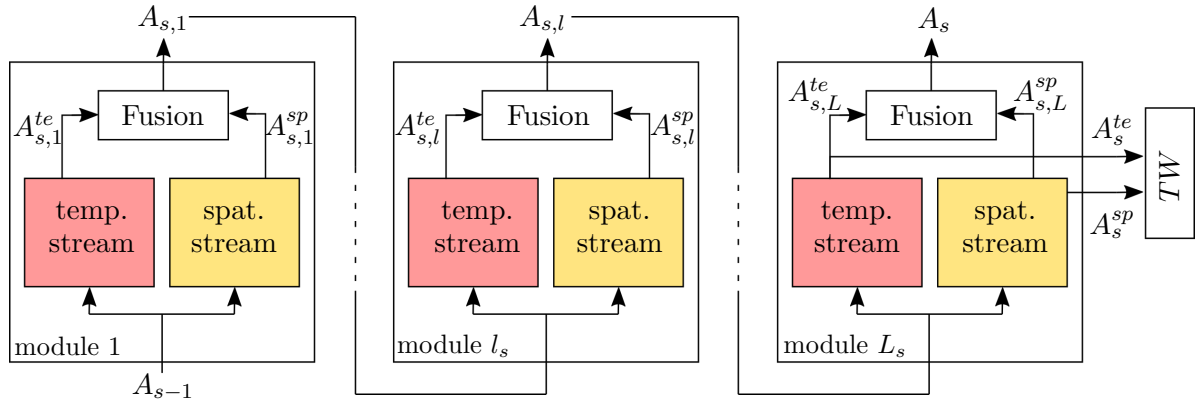

Figure 4.5: Overview of an encoder stage E_s considering L_S consecutively applied FE_{hyb} modules. A detailed overview of the spatial and temporal streams is given in figure 4.6.

dencies in one temporal sequence of patches representing the same spatial area on the ground. The structure of the temporal stream is depicted on the left side in figure 4.6. Throughout the temporal stream the sequence-like structure $Z_{s,l-1}$ is used because self-attention is applied independently for all timeseries of features corresponding to the same spatial region. The total number of spatially separated patches in stage s is denoted by N_s, resulting in N_s MSA blocks that are applied separately. In each block, one sequence of T patches that represent the same spatial area on the ground (red cuboid in figure 4.4) is used to compute new features that encode temporal dependencies. The index $n_p \in [1, ..., N_s]$ is used to indicate the n_p-th sequence of feature vectors obtained from $Z_{s,l-1}$, i.e patches, qhich is denoted by $Z^{n_p}_{s,l-1} \in \mathbb{R}^{T \times C_s}$. This results in the following computation in the n_p-th MSA block:

$$\begin{aligned} \hat{Z}^{n_p}_{s,l} &= MSA(LN(Z^{n_p}_{s,l-1})) + Z^{n_p}_{s,l-1} \\ Z^{n_p}_{s,l} &= MLP(LN(\hat{Z}^{n_p}_{s,l})) + \hat{Z}^{n_p}_{s,l}, \end{aligned} \tag{4.2}$$

by using:

$$\begin{aligned} MSA(LN(Z^{n_p}_{s,l-1})) &= Concat(head_1, \ldots, head_{n_h}) \cdot W^O \\ head_h &= Att(Q_h, K_h, V_h) \\ Att(Q_h, K_h, V_h) &= Softmax(Q_h K_h^T / \sqrt{c_k}) \cdot V_h \end{aligned} \tag{4.3}$$

with

$$\begin{aligned} Q_h &= LN(Z^{n_p}_{s,l-1}) \cdot W^Q_h, \\ K_h &= LN(Z^{n_p}_{s,l-1}) \cdot W^K_h, \\ V_h &= LN(Z^{n_p}_{s,l-1}) \cdot W^V_h. \end{aligned} \tag{4.4}$$

$\hat{Z}^{n_p}_{s,l} \in \mathbb{R}^{T \times C_s}$ represents the output of the MSA-layer, $Z^{n_p}_{s,l} \in \mathbb{R}^{T \times C_s}$ denotes the output after applying the MLP and the residual connection and LN indicates layer normalization $W^Q_h, W^K_h \in \mathbb{R}^{C_s \times c_k}$, $W^V_h \in \mathbb{R}^{C_s \times c_v}$ and $W^0 \in \mathbb{R}^{n_h \cdot c_v \times C_s}$ denote parameter matrices for the linear projections and $head_h$ represents the hth head of a total number of n_h heads. The number of heads n_h is set

as a hyper-parameter and the feature dimensions c_k of Q_h and K_h and c_v of V_h are defined as $c_k = c_v = C_s/n_h$.

As the last step in the temporal stream of the FE_{hyb} module, the resulting features of all N_s MSA blocks are stacked, meaning that the N_s outputs of size $T \times C_s$ of the MSA blocks are stacked to a tensor of size $T \times C_s \times N_s$ (indicated by st in figure 4.6). This results in the final output of the temporal stream $Z^{te}_{s,l} \in \mathbb{R}^{T \times C_s \times N_s}$ which is rearranged to $A^{te}_{s,l} \in \mathbb{R}^{T \times C_s \times H_s \times W_s}$ for the fusion with the features determined in the spatial stream.

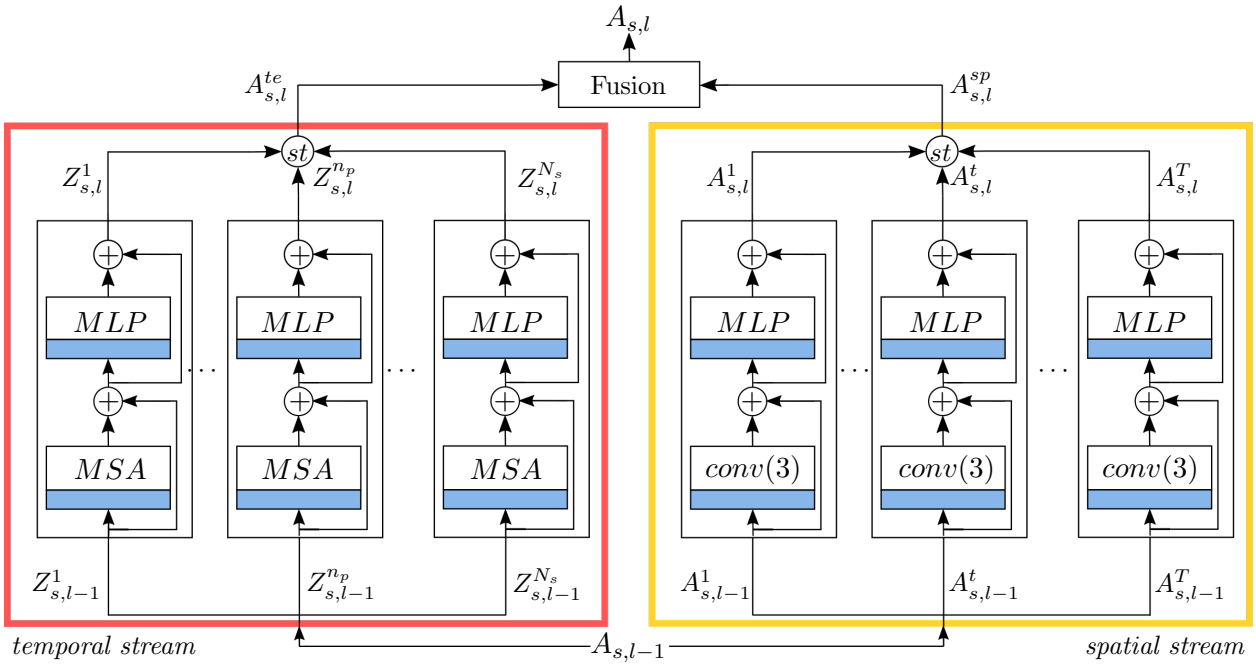

Figure 4.6: Structure of the FE_{hyb} module. Blue rectangles: Layer normalisation, $\oplus$: Element-wise addition, $\textcircled{st}$: Stacking of outputs to one feature matrix, $conv(k)$: Convolutional layer with kernel size k. MSA: Multi-head self-attention, MLP: Multilayer Perceptron.

Spatial stream: In the spatial stream of the FE_{hyb} module, 2D convolutional layers are applied to the input patches of the same acquisition date t. The number of used acquisition dates T is, therefore, equal to the number of convolutional blocks applied separately in this stream. The number of patches that describe the same acquisition date in stage s is denoted by $N_s = H_s \cdot W_s$, which is the number of patches processed in one block in the spatial stream. The structure of the spatial stream is depicted on the right-hand side of figure 4.6.

The input $A_{s,l-1} \in \mathbb{R}^{T \times C_s \times H_s \times W_s}$ to the FE_{hyb} module is separated into T feature maps $A^t_{s,l-1} \in \mathbb{R}^{C_s \times H_s \times W_s}$, each of which contains the features of the corresponding timestep t. $A^t_{s,l-1}$ serves as input to the t-th convolutional block, and new features are computed based on the following equation:

$$\begin{aligned} \hat{A}^t_{s,l} &= conv(LN(A^t_{s,l-1})) + A^t_{s,l-1} \\ A^t_{s,l} &= MLP(LN(\hat{A}^t_{s,l})) + \hat{A}^t_{s,l}, \end{aligned} \tag{4.5}$$

with $\hat{A}^t_{s,l} \in \mathbb{R}^{C_s \times H_s \times W_s}$ representing the output of the convolutional layer after applying the residual connection and $A^t_{s,l} \in \mathbb{R}^{C_s \times H_s \times W_s}$ representing the output of the block after applying the MLP

and another residual connection. The symbol $conv()$ represents a convolutional layer with kernel size $k = 3$ and C_s output dimensions. The weights of the kernel matrices are shared across time, i.e. in each of the T convolutional blocks the same kernels are used. Similar to the processing in the temporal stream, the resulting features from all T convolutional blocks are stacked (st in figure 4.6) to form the final output of the spatial stream $A_{s,l}^{sp} \in \mathbb{R}^{T \times C_s \times H_s \times W_s}$.

Feature fusion: In the final step of the FE_{hyb} module, the features extracted by the temporal and spatial streams of the module are combined. For this purpose, the image-like structures of $A_{s,l}^{te} \in \mathbb{R}^{T \times C_s \times H_s \times W_s}$ and $A_{s,l}^{sp} \in \mathbb{R}^{T \times C_s \times H_s \times W_s}$ of the outputs of the temporal and spatial streams are used. The spatial features $A_{s,l}^{sp}$ and the temporal features $A_{s,l}^{te}$ (rearranged from $Z_{s,l}^{te}$) are concatenated, resulting in $A_{s,l}^{con} \in \mathbb{R}^{T \times 2C_s \times H_s \times W_s}$. To obtain a feature dimension of C_s again, a fully connected layer is applied in the feature dimension to the map $A_{s,l}^{con}$ to obtain the final output $A_{s,l} \in \mathbb{R}^{T \times C_s \times H_s \times W_s}$. $A_{s,l}$ serves as the input to the next FE_{hyb} module, or, in the case of the last module in a stage, it serves as input to the patch merging layer. Additionally, the outputs of the spatial and temporal streams of the last module in stage s serve as input to the temporal weighting module TW for the skip connections of stage s (cf. section 4.2.7).

4.2.6.2 Attention-based embedding of spatial and temporal dependencies

The attention-based module FE_{att} to encode spatial and temporal dependencies is the first out of two module variants that are used to analyse the performance of the proposed FE_{hyb} module in comparison to modules that are solely based on self-attention. Comparing the FE_{hyb} module to the FE_{att} module will help to analyse the impact of using convolutional layers instead of self-attention in the spatial stream. The FE_{att} module has a similar structure as the FE_{hyb} module, but instead of using convolutions in the spatial stream, self-attention is used in both streams. All other aspects are identical, including the splitting of the features in both streams and the feature fusion module. As the structure of the temporal stream and the feature fusion is the same as described for the FE_{hyb} module in section 4.2.6.1, only the adapted spatial stream is described in the following.

Spatial stream based on self-attention: In the spatial stream of the FE_{att} module, the input features $A_{s,l-1}$ are split into T feature maps $A_{s,l-1}^{t}$ with $t \in [1, ..., T]$ in the same way as in the FE_{hyb} module. $A_{s,l-1}^{t}$ is then rearranged into the sequence-like structure $Z_{s,l-1}^{t}$, i.e. the two spatial dimensions $(H_s \cdot W_s)$ are combined into one dimension $N_s = H_s \cdot W_s$. Each input $Z_{s,l-1}^{t}$ is processed by a self-attention block in which a window-based multi-head self-attention (W-MSA) layer is applied as introduced in section 2.2.2.4. In the W-MSA layer, the spatial dimensions are again partitioned into windows of $M \times M$ patches to reduce the computational complexity of the layer. In each block, the W-MSA module is applied, resulting in the following computations:

$$\begin{aligned} \hat{Z}_{s,l}^{t} &= W\text{-}MSA(LN(Z_{s,l-1}^{t})) + Z_{s,l-1}^{t} \\ Z_{s,l}^{t} &= MLP(LN(\hat{Z}_{s,l}^{t})) + \hat{Z}_{s,l}^{t}, \end{aligned} \tag{4.6}$$

with $\hat{Z}_{s,l}^{t} \in \mathbb{R}^{C_s \times N_s}$ representing the output of the W-MSA layer after applying the residual connection and $Z_{s,l}^{t} \in \mathbb{R}^{C_s \times N_s}$ representing the output of the block. After applying the MLP and another residual connection. The resulting features from all T blocks are stacked to form the final output

of the spatial stream $Z_{s,l}^{sp} \in \mathbb{R}^{T \times C_s \times N_s}$. As in the original Swin Transformer, the used windows are shifted by $M/2$ in the following block, i.e. block $l+1$, to integrate global spatial context.

4.2.6.3 Joint consideration of spatio-temporal dependencies

Both modules introduced so far, FE_{hyb} and FE_{att}, consider spatial and temporal dependencies in separate streams before the results are combined in the last step of the modules. However, some works, e.g. (Arnab et al., 2021), have shown that extracting spatio-temporal features in one self-attention layer can result in improved performance. The major challenge in such a module is that the computational complexity is considerably increased because T times as many patches serve as input to one self-attention layer. Furthermore, convolutions can not be applied in such a scenario, which can impact the models accuracy.

To be able to compare FE_{hyb} to a module using self-attention to jointly extract spatial and temporal dependencies, the FE_{full} module is introduced. A visualisation of patches considered in one layer that jointly computes spatio-temporal features is shown in figure 4.7, while a detailed visualization of the module components is given in figure 4.8. The FE_{full} module consists of two parts: First, spatial dependencies are encoded for all timesteps in parallel similarly to the spatial stream of the FE_{att} module. Afterwards, the features are combined and serve as input to the second part, in which spatio-temporal features are computed from all patches of all timesteps inside a defined spatial window based on self-attention. Both components of the FE_{full} module are described in the following.

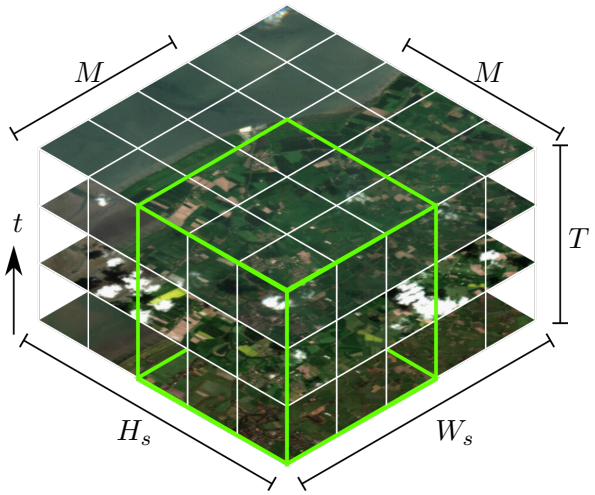

Figure 4.7: Visualization of the principle of feature extraction in the second part of the FE_{full} module. The green cuboid indicates the patches considered in one self-attention block that uses the $W\text{-}MSA_{3D}$ layer to compute spatio-temporal features. (H_s, W_s): Height and width of the feature map in stage s, M: Window size, t: Index for timesteps. Note that the satellite images are only used for visualisation purposes because commonly, the input to a FE_{full} module are the features from the previous FE_{full} module or the patch generation and not the original spectral input bands.

Considering spatial context with self-attention: The embedding of spatial context in the FE_{full} module is identical to the one used in the spatial stream in the FE_{att} module: The input $A_{s,l-1} \in \mathbb{R}^{T \times C_s \times H_s \times W_s}$ is separated into T feature maps, resulting in $Z_{s,l-1}^{t} \in \mathbb{R}^{C_s \times H_s \times W_s}$ as the

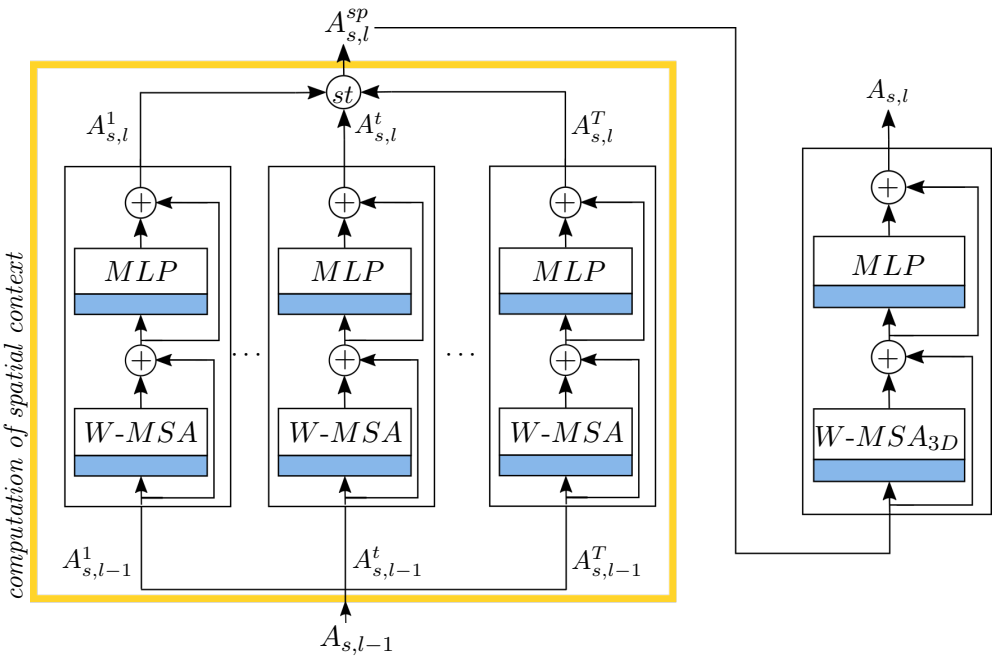

Figure 4.8: Overview of the FE_{full} module. $W\text{-}MSA$ indicates that features are computed within a window of $M \times M$ patches to consider spatial context while $W\text{-}MSA_{3D}$ indicates that joint spatio-temporal features are computed within a window of size $M \times M \times T$. Blue rectangles: Layer normalisation, $\oplus$: Element-wise addition, $\textcircled{st}$: Stacking of outputs to one feature matrix.

input to the t-th self-attention block. Afterwards, in each block, a W-MSA layer is applied to the sequence-like structure of the input features. The overall procedure is described by equation 4.6. At the end, the features resulting from all T blocks are stacked (st in figure 4.8) to form the output of the first part of the FE_{full} module, which is denoted by $A^{sp}_{s,l} \in \mathbb{R}^{T \times C_s \times H_s \times W_s}$.

Joint computation of spatio-temporal features with self-attention: The second part of the FE_{full} module is the block that jointly considers spatio-temporal dependencies, which is shown on the right side in figure 4.8. $A^{sp}_{s,l}$ serves as input to this module. In contrast to the modules FE_{hyb} or FE_{att}, the input directly serves as input to the $W\text{-}MSA_{3D}$ layer, and is not divided into several inputs of small dimensionalities. To be able to compute self-attention across both spatial and temporal dimensions, the W-MSA is adapted to handle multi-temporal input data, which is indicated by the symbol $W\text{-}MSA_{3D}$ in figure 4.8. The $W\text{-}MSA_{3D}$ layer is an extension of the W-MSA layer described in section 2.2.2.4. The main extension is an adaptation of the window partitioning layer of the Swin Transformer (cf. section 2.2.2.4). In the original window partitioning, an input of size $C_s \times H_s \times W_s$ is arranged into non-overlapping regions, each having a size of $C_s \times M \times M$, followed by applying self-attention to the sequence-like structure of these features of size $C_s \times M^2$. In the 3D window partitioning layer, the input $A^{sp}_{s,l}$ of size $T \times C_s \times H_s \times W_s$ is rearranged to non-overlapping windows, each of size $T \times C_s \times M \times M$. To be able to encode information from the spatial and temporal dimensions jointly, these features are rearranged to the sequence-like structure, by combining the spatial and temporal dimensions to $Z^{sp,w}_{s,l} \in \mathbb{R}^{C_s \times TM^2}$, with w indicating that these features belong to one of the windows. Figure 4.7 shows an exemplary visualisation of the input patches that are considered in one of the windows, indicated by the green cuboid. Afterwards, multi-head self-attention is computed for all windows in parallel using

equations 2.22 and 2.23. The number of input patches to one self-attention layer is $N = T \cdot M^2$ as all timesteps are used in window partitioning as well as the self-attention layer. This results in query, key and value matrices $Q_{3D}, K_{3D}, V_{3D} \in \mathbb{R}^{TM^2 \times C_s}$ serving as input to equations 2.22 and 2.23. The number of considered dependencies increases from M^4 (standard $W\text{-}MSA$) to M^4T^2, as self-attention is computed between all combinations of patches that serve as input for one window. Increasing the number of input patches by the factor of T therefore results in a significant increase in the number of computations, leading to a longer training and inference process. The remaining parts of the joint feature extraction block are similar to those of the W-MSA, resulting in the following computations:

$$\begin{aligned} \hat{Z}_{s,l} &= W\text{-}MSA_{3D}(LN(Z^{sp}_{s,l})) + Z^{sp}_{s,l} \\ Z_{s,l} &= MLP(LN(\hat{Z}_{s,l})) + \hat{Z}_{s,l}, \end{aligned} \tag{4.7}$$

with $Z^{sp}_{s,l}$ denoting the output of the spatial self-attention, $\hat{Z}_{s,l}$ representing the output of the spatio-temporal $W\text{-}MSA_{3D}$ layer and $Z_{s,l} \in \mathbb{R}^{T \times C_s \times N_s}$ being the output of the MLP after applying a residual connection. $Z_{s,l}$ is rearranged to $A_{s,l}$, which serves as the final output of the FE_{full} module. In the FE_{full} module, spatial and temporal dependencies are not considered separately, and therefore, the temporal weighting module cannot be used directly when this module is used in a model variant.

4.2.7 Temporal weighting in skip connections

Skip connections are used to combine features from the encoder stages with those from the decoder. The main advantage of these connections is that features extracted at finer spatial resolutions, i.e. the earlier encoder layers, serve as additional input to the later layers of the model decoder, which is expected to increase the classification accuracy, especially near object boundaries or for fine structures. In all encoder stages of the new method, spatial and temporal dependencies are considered separately before the results of the two streams are combined for the next stage, which involves a reduction of the feature dimension. The idea of the new temporal weighting module TW is the usage of the outputs of the spatial and temporal streams directly in the skip connections. This idea builds on the temporal weighting module introduced by Garnot and Landrieu (2021), who interpret the features from the temporal streams as weights for the individual timesteps, indicating which of the timesteps contain the most important information. The temporal features are then used as weights for the spatial features. However, this weighting could also be interpreted the other way around, i.e. that the features from the spatial stream are used to weight the features from the temporal stream, because the weighting is realised by an element-wise matrix multiplication. As the TW module needs separated sets of features computed by the temporal and spatial streams, this module can only be used with encoder modules that provide such outputs. Therefore, in the context of this thesis, the TW module can be used with the proposed FE_{hyb} module as well as with the FE_{att} module. The FE_{full} module cannot be used in combination with the TW module, because it computes a single set of spatio-temporal features. The general structure of the TW module is shown in figure 4.9.

In the proposed model, the outputs of the temporal (A_s^{te}) and spatial (A_s^{sp}) streams of the last FE_{hyb} module in stage s serve as input to the TW module of stage s (cf. figure 4.2). In the TW module, first, A_s^{te} and A_s^{sp} are multiplied element-wise, which can be interpreted as a weighting of the spatial features by the temporal features. Afterwards, a convolutional layer with $k = 3$, followed by BN and an activation function is applied to map the features to C_{dec} dimensions that are needed in the UPer-Net decoder. Similarly to the original UPer-Net, the output features $A_s^{TW} \in \mathbb{R}^{T \times C_{dec} \times H_s \times W_s}$ are combined with the decoder features by element-wise addition in the skip connections (cf. figure 4.2).

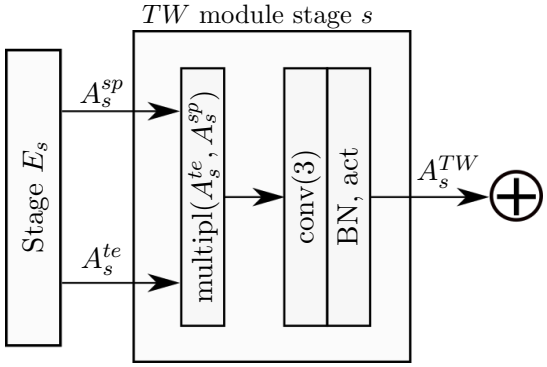

Figure 4.9: Structure of the temporal weighting module TW. multipl(·): Element-wise multiplication of features in brackets, conv(k): Convolutional layer with kernel size k, BN: Batch normalization, act: activation function. ⓧ: Element-wise addition with upsampled features from previous skip connection.

4.3 Training

The new model introduced in section 4.2 is trained by minimising a weighted cross entropy loss $\mathcal{L}_{CE\omega}(\mathcal{T}_{tr}, \Theta)$ based on the training dataset $\mathcal{T}_{tr}$ and the model parameters Θ that are updated in the training process. The training dataset $\mathcal{T}_{tr} = \{X_j, Y_j\}_{j=1}^{n_{\mathcal{T}_{tr}}}$ consists of a number $n_{\mathcal{T}_{tr}}$ of multi-temporal input images $X \in \mathbb{R}^{T \times B \times H_0 \times W_0}$ with corresponding multi-temporal label maps $Y \in \mathbb{R}^{T \times H_0 \times W_0}$ (cf. section 4.1). The weights Θ_0 are initialised randomly using variance scaling as introduced in section 2.1.4.3. During the iterative training process, mini-batch stochastic gradient descent (Bishop, 2006) with adaptive momentum (Kingma and Ba, 2015) as introduced in section 2.1.4.2 is used to update the model parameters Θ based on the computed loss and its gradients. In each training iteration, only a minibatch $\mathcal{T}_{mb} = \{X_k, Y_k\}_{k=1}^{mb} \subset \mathcal{T}_{tr}$ consisting of mb image-time-series $X_k \in \mathbb{R}^{T \times B \times H_0 \times W_0}$ and corresponding label maps $Y_k \in \mathbb{R}^{H_0 \times W_0}$, is used to compute the loss $\mathcal{L}_{CE\omega}(\mathcal{T}_{mb}, \Theta_{i-1})$ based on the weights of the previous iteration $i - 1$:

$$\mathcal{L}_{CE\omega}(\mathcal{T}_{mb}, \Theta_{i-1}) = -\frac{1}{n_{mb}} \sum_{n=1}^{n_{mb}} \sum_{c=1}^{n_c} O_n^c \cdot log(a_n^c) \cdot \omega_c, \tag{4.8}$$

where a_n^c is the predicted softmax score of the n-th pixel in the current minibatch to belong to class $y_n = C^c$, $n_{mb} = mb \cdot T \cdot H_0 \cdot W_0$ is the number of pixels in the current minibatch that is used to compute the loss, n_c is the total number of classes, ω_c is a weight assigned to class C^c, and O_n^c is a binary indicator that equals 1 if the nth pixel belongs to class C^c and equals 0 otherwise. Note

that the loss is computed based on the predicted maps for all timesteps by comparing them to the corresponding label map and all timesteps contribute to the loss in the same way. Training is conducted for a number n_{it} of iterations, which is referred to as one epoch. After an epoch finished, the current accuracies on the validation dataset are computed before the next epoch starts.

The class weights $\omega = [\omega_1, ..., \omega_{n_c}]$s are used to compensate for the imbalanced class distribution that is commonly included in RS datasets. Without any weighting, the loss will be dominated by the most frequent classes, while underrepresented classes only contribute a marginal fraction. As a result, the model weights are primarily updated to correctly predict samples of the dominant classes, while pixels having labels from underrepresented classes have a much smaller impact. In this thesis, the standard cross-entropy loss (cf. section 2.1.4.1) is therefore extended by class-dependent weights ω_c based on the adaptive cross-entropy loss $\mathcal{L}_{CE\omega}$ introduced by Wittich and Rottensteiner (2021), which is based on the current ability of the classifier to predict the class labels correctly. This is done by assigning a weight to each class after each epoch based on a class-wise accuracy metric. In this way, not only class imbalance is counteracted, but furthermore, classes that are difficult to distinguish also obtain a higher weight when the corresponding class-wise accuracies are low. The class weights $\omega = [\omega_1, ..., \omega_{n_c}]$ are initialised with ones for the first training epoch, which corresponds to minimizing an unweighted cross-entropy loss. After each training epoch, n_ω randomly chosen minibatches from the training dataset are classified based on the model parameters Θ_e from the previous epoch e, and the obtained predictions are used to compute class-based accuracy metrics. In this thesis the intersection over union is chosen:

$$IoU_c = \frac{TP_c}{TP_c + FP_c + FN_c}. \tag{4.9}$$

In equation 4.9, TP_c, FP_c, and FN_c refer to the number of pixels that are true positives, false positives, and false negatives, respectively, with respect to class $c \in [1, ..., n_c]$. The results highly depend on the minibatches chosen for the calculation of the IoUs. In general, a higher number of minibatches n_ω would be beneficial to obtain IoU scores that are representative for the whole training dataset. In practice, the process of computing the accuracy scores slows down the training process, and thus, a relatively small value is chosen for n_ω. To counteract this problem, starting from epoch two, the $IoUs$ from the last $n_{e\omega}$ epochs (or from all available ones before epoch $n_{e\omega}$) are averaged. Following (Wittich and Rottensteiner, 2021), these IoU scores are then used to determine the class weights ω_c for the next epoch:

$$\omega_c = (1 - \Delta IoU_c)^\kappa = [1 - (IoU_c - \frac{1}{n_c}\sum_{h}^{n_c} IoU_h)]^\kappa, \tag{4.10}$$

where ΔIoU_c is the difference between the IoU of class C^c and the mean IoU of all classes, n_c denotes the total number of classes, and the hyperparameter κ is used to scale the influence of class weights for classes with a lower IoU on the results. These class weights are then used in the loss (equation 4.8) during the following epoch.

Afterwards, the moving averages of the gradients (cf. equation 2.9) and the squared gradients (cf. equation 2.10) are updated, which are then used to update the weights by using the ADAM optimiser as described with equation 2.11. One epoch finishes when a pre-defined number n_{it} of iterations is conducted, and after each epoch the overall accuracy on the validation dataset

is computed to monitor it during the training process for early stopping. In early stopping the training process is stopped when the validation accuracy is saturated for a fixed number of epochs n_{es}. If this is not the case, training continues until the maximum number of epochs n_e is reached. After the training process is finished, the model weights from the epoch with the best validation accuracy are considered to be the result of the training process and, thus, used for the evaluation process.

Dropout: For the introduced method, two types of dropout (cf. section 2.1.4.6) are adapted from the original Swin Transformer for regularisation purposes. The first type, which is referred to as attention dropout, is applied to the output tensor after the softmax function in the computation of any multi-head self-attention block (cf. equation 4.2). The attention dropout rate adr can be interpreted as the probability that a value in the attention matrix, i.e. QK^T, is dropped. Practically, this is realised by setting the corresponding values in the attention matrix (QK^T) to zero, which means that these dependencies are not considered. Dropout is only applied in the training process. To obtain output values with a same scaling during training and inference, the output values of the dropout layer are scaled with $\frac{1}{1-adr}$ in the training process.

The second type of dropout is stochastic depth dropout, which is applied to the output of the multi-head self-attention layers (e.g. $\hat{Z}_{s,l}^{n_p/t}$ in FE_{hyb}) as well as to the MLP layers in each self-attention block (e.g. $Z_{s,l}^{n_p/t}$ in FE_{hyb}). The stochastic depth dropout rate sdr_s refers to the probability that an output patch of the corresponding layer in stage s is dropped (i.e. set to zero) and is not considered in the forward pass. The stochastic depth dropout rate is increased with the model depth from $[0, .., sdr_S]$ with a number of steps that is equal to the number of stages in the model. Similarly to the attention dropout, the stochastic depth dropout is only used in the training process and the output values of a stochastic depth dropout layer is scaled by $\frac{1}{1-sdr_S}$.

5 Experimental Setup

In this chapter, overviews of the used datasets, the evaluation strategy, and the experimental setup to analyse the proposed method for multi-temporal SITS classification is given. In section 5.1, the datasets used for pixel-wise LC classification are introduced. Afterwards, the objectives and the structure of the different sets of experiments are described in section 5.2 before the evaluation strategy and the quality metrics used to evaluate the results are introduced in section 5.3.

5.1 Datasets

In the context of this thesis, three different datasets for LC classification are used. The first two datasets consist of Sentinel-2 SITS. One of them covers the whole federal state of Lower Saxony in Germany for a period of four years. The image data is combined with label data from a topographic database providing information about land cover. The usage of these class labels results in some wrong class labels (*label noise*). For this reason, a small part of the dataset is manually corrected and used as an additional test dataset. The second Sentinel-2 dataset is the Kleve dataset. This dataset is located in the neighbouring federal state of North-Rhine-Westphalia and consists of 1600 manually labelled pixels for different LC classes. This dataset is only used as an additional test dataset for the models trained on the area of Lower Saxony to evaluate its generalisation abilities. The third dataset consists of Planet images for 75 different areas spread around the whole Earth. The images cover a period of two years and have a GSD of 3 m. The corresponding class labels were generated manually. In the following, all datasets are described in detail.

5.1.1 Lower Saxony & Kleve datasets

Lower Saxony dataset: The Lower Saxony dataset covers the whole area of the German federal state of Lower Saxony (47600 km^2) as shown in the overview in figure 5.1. The dataset comprises Sentinel-2 images acquired within four years, between January 2019 and December 2022, in combination with pixel-wise class labels that are derived from the official German landscape model ATKIS (AdV, 2008). Image and label data are both provided for multiple timesteps that are combined to form the multi-temporal input data for the model proposed in section 4.

Image data: The used Sentinel-2 images are preprocessed Level-2A data, which contain georeferenced bottom-of-atmosphere reflectance and cloud masks from the top-of-atmosphere reflectance of every pixel (Drusch et al., 2012). The used images contain four spectral bands (red, green, blue and near infrared) with a GSD of 10 m. The six spectral bands having 20 m resolution are excluded, as previous experiments did not show an improvement when these bands were used as well. All

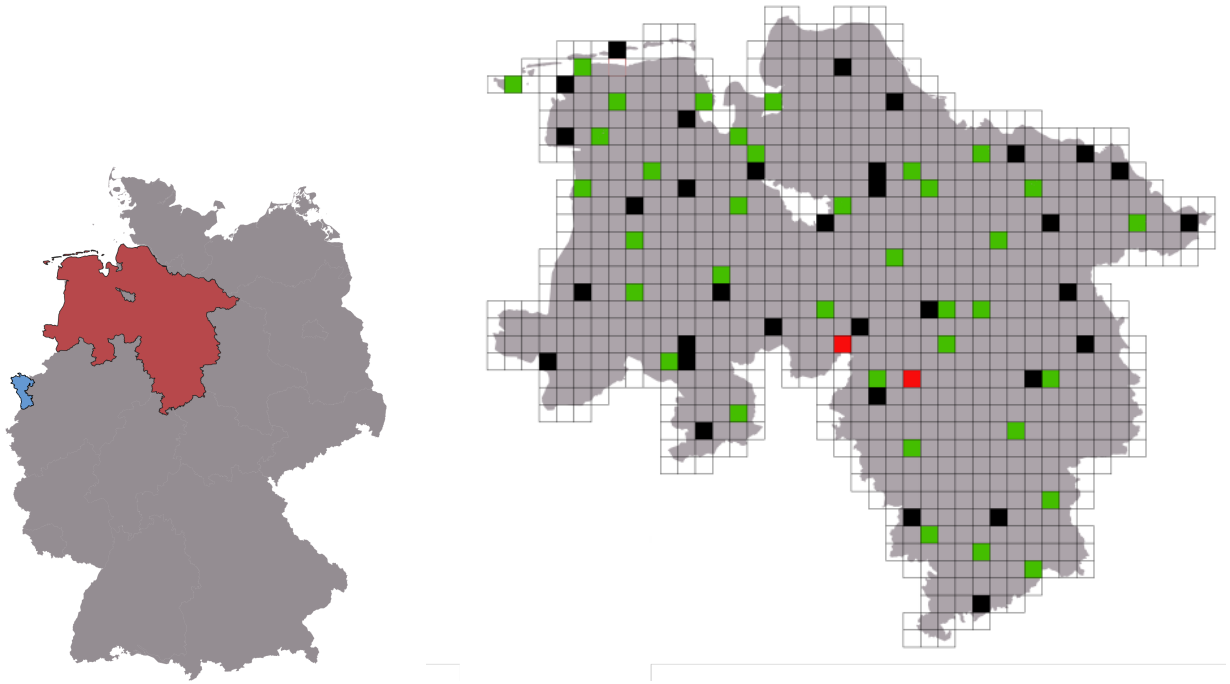

Figure 5.1: Overview of the dataset located in Lower Saxony in Germany. The figure on the left shows where Lower Saxony is located in Germany (red). Additionally, the district of Kleve, situated in the German federal state of North-Rhine-Westphalia, is shown in blue. The area of Lower Saxony is shown on the right, where squares show the available tiles of 8×8 km^2 each. Grey/green: potential training/validation tiles. Red: test tiles with manually corrected reference (dataset $\mathcal{T}^L_{cor}$). Black: test tiles without corrected reference (dataset $\mathcal{T}^L_{te}$).

bands are normalized to zero-mean and unit standard deviation by using $x^n_{h,w,b} = (x_{h,w,b} - \mu_b)/\sigma_b$, where $x^n_{h,w,b}$ and $x_{h,w,b}$ correspond to the normalized and the original grey value of the pixel at position (h, w) for band b of an image, respectively. μ_b and σ_b denote the mean and standard deviation of band b, respectively, and are computed based on all Sentinel-2 images acquired in 2019 and 2020 before the training process started.

Label data: The class labels are obtained from the official German landscape model ATKIS (AdV, 2008). This database contains information about 113 different land use classes in vectorised format. As this high number of classes is too detailed for automatic classification, a suitable class structure for LC classification had to be defined by merging several ATKIS classes into one LC class. The selected class structure is based on the AdV land cover scheme (AdV, 2024) which defines 15 LC classes. These classes were further reduced to 8 LC classes with regard to the recognisability and differentiability in satellite images with 10 m GSD. In this process, for example, the classes herbaceous and woody vegetation are combined and the class ice is excluded as it does not occur in the selected area. This strategy resulted in the following eight LC classes that are used in the experiments: *Settlement (stl.), Sealed area (sld.), Agriculture (agr.), Vegetation (veg.), Deciduous forest (dfor.), Coniferous forest (cfor.), Water (wat.)* and *Barren land (bar.)*. In addition, the class *others* is used for areas without useful label information. This includes areas outside the state borders that are included in the satellite imagery (white areas in the squares in figure 5.1) and ATKIS classes that cannot be clearly assigned to one of the LC classes (e.g. the ATKIS class *mixed*

forest). Samples assigned to this class are disregarded during training (no loss is computed) and also during evaluation. The ATKIS database is updated based on in-situ surveys and aerial flights that take place every three years. The updates are integrated into the available database every three months (ends of March, June, September and December), resulting in four label maps in a year. To be able to combine these reference labels with the satellite imagery, the available polygons are rasterized at the GSD of the satellite imagery (10 m). The class *Sealed area* mainly includes line objects in ATKIS (roads/railway tracks). To be able to use this information in a rasterized format, the lines are buffered with a specific width to both sides before rasterisation. Afterwards, if a pixel is covered at least 50% by this buffer region, it is assigned to the class *Sealed area*. The road width is chosen based on the type of the line and varies between 20 m for country roads and railway tracks and 40 m for highways. Due to the spatial resolution of 10 m, smaller streets are not rasterized when their buffer region does not cover 50% of surrounding pixels. For the experiments in this thesis, each Sentinel-2 image is combined with the label image closest in time to its acquisition date. The whole procedure leads to some *label noise* because some more recent changes visible in the images are not yet contained in the ATKIS database.

Tiling: The image and label data is provided in image tiles of $8 \times 8\,\mathrm{km}^2$ (800×800 pixels). For the whole area of Lower Saxony this results in a total number of 885 tiles that are shown as squares with black outlines in figure 5.1. For each tile, multi-temporal image and label data is provided. The label data is available every three months, resulting in twelve label maps for each tile over the time period of four years. The satellite images are available for different numbers of acquisition dates, because a cloud filter is applied to filter out images with a high cloud coverage. The cloud filter is based on the cloud masks that are provided in addition to the spectral bands for every acquisition date and contain the probability that a pixel is covered by a cloud. With this mask, the average cloud coverage for one tile is computed and afterwards, those tiles that contain more than 5% cloud coverage are excluded from the dataset. This process is done independently for each acquisition date and tile and results in a variable number of available images for different regions.

Train-test split: The splitting into a training, validation, and test dataset is based on a spatial separation, based on the introduced tiles. This means that the images and label maps of all timesteps for one tile are assigned to the selected dataset. During the training/validation or test phases, all available images and labels of a tile are then used to create one input to the model, which is dependent on the number of used timesteps, cf. section 4.1.1. The dataset is split into a training dataset $\mathcal{T}_{tr}^{L}$ of 813 tiles, a validation dataset $\mathcal{T}_{val}^{L}$ (green in figure 5.1) and a test dataset $\mathcal{T}_{te}^{L}$ (black and red in figure 5.1), both containing 36 tiles. The superscript L indicates that training is conducted on the Lower Saxony dataset to make it easily distinguishable from the other datasets introduced later in this section and in section 5.1.2. The test dataset $\mathcal{T}_{te}^{L}$ consists of label maps from the ATKIS database, which results in some *label noise* as discussed before. Therefore, the accuracy metrics achieved for $\mathcal{T}_{te}^{L}$ are not reliable. For this reason, an additional manually corrected test dataset $\mathcal{T}_{cor}^{L}$ is used. For two test tiles (red in figure 5.1) the corresponding label maps of several timesteps were corrected manually, resulting in eight corrected label maps (two corrected timesteps for the first tile plus six for the second tile). This manual work was carried out to obtain a reference for the evaluation that is not or only slightly affected by label noise. Note that the two tiles that are included in $\mathcal{T}_{cor}^{L}$ are also included in the test dataset $\mathcal{T}_{te}^{L}$, but with the ATKIS labels instead of

the corrected ones. $\mathcal{T}^L_{cor}$ is not affected by the errors in the labels, but it contains a limited number of reference labels so that small changes in the classification results might have a relatively large impact on the quality indices determined for evaluation based on this test dataset.

The distribution of the class labels in the training, validation, and test datasets is shown in table 5.1. Some examples, including Sentinel-2 images of the different seasons, labels from the test dataset $\mathcal{T}^L_{te}$ and the corrected test dataset $\mathcal{T}^L_{cor}$ are shown in figure 5.2.

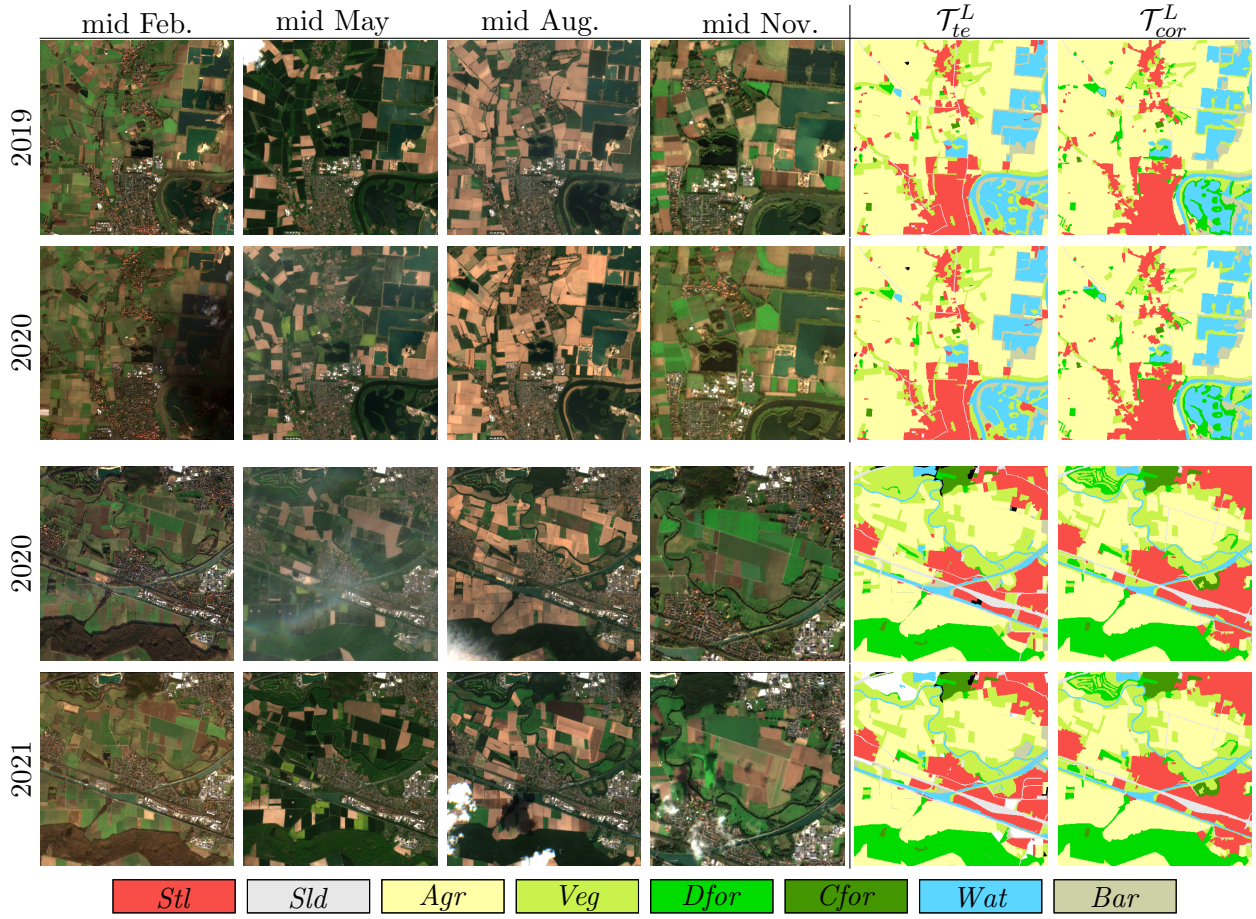

Figure 5.2: Exemplary image and label data for the Lower Saxony dataset. All of the samples show an area of approx. 4 km × 4 km (400 px × 400 px, a quarter of a tile). The label maps shown for $\mathcal{T}^L_{te}$ are those from the end of June of the corresponding year, which were manually corrected for $\mathcal{T}^L_{cor}$.

Kleve dataset: In addition to the corrected test dataset $\mathcal{T}^L_{cor}$ of the Lower Saxony dataset, $\mathcal{T}^L_{kleve}$ is used as an additional manually labeled test dataset. It is used in addition to the test datasets $\mathcal{T}^L_{te}$ and $\mathcal{T}^L_{cor}$. This dataset, referred to as the Kleve dataset, is based on 1600 manually labeled points that are randomly spread over the area of the district Kleve, belonging to the German federal state North-Rhine-Westphalia. The area of Kleve is shown in blue in figure 5.1. The labeling process was based on digital orthophotos acquired on 24/03/2020 and initially conducted for the project Cop4All, which had the goal of using freely available satellite data in combination with high-resolution digital orthophotos to automatically derive land cover maps that can be used by the surveying authorities in North-Rhine-Westphalia and other federal states in Germany to update their topographic databses (Sandmann et al., 2022). As shown in coloumn $\mathcal{T}^L_{kleve}$ in table 5.1, the

class distribution for Kleve is different from the one of the Lower Saxony dataset, and it is also more balanced. The provided reference points are combined with the available Sentinel-2 images for the Kleve district for the year 2020. These are processed similarly to the Sentinel-2 images of Lower Saxony and also a tiling of 8 km×8 km is used. Overall, 35 tiles cover the area of Kleve. In figure 5.3, some example tiles (8×8 km) are shown for the Kleve dataset. In comparison to the Lower Saxony dataset, where one tile already contains 640.000 class labels only for one timestep, the number of 1600 points is very small . However, these points are inspected manually in orthophotos of higher resolution, which results in very reliable labels for the Kleve dataset.

In the context of this thesis, the Kleve dataset is used as an additional test dataset to evaluate the results obtained by classifiers trained on the Lower Saxony dataset $\mathcal{T}_{tr}^{L}$. For this purpose, input timeseries are generated for all tiles covering the Kleve district in 2020 as described in section 4.1.1, and these input timeseries are classified. As the labels are mono-temporal, the obtained multi-temporal classification maps are combined by a majority voting mechanism, meaning that for each pixel in the timeseries, the class that is predicted most frequently is selected as the representative LC class for the whole year. One focus of the evaluation on the Kleve dataset is the generalisation performance, because the area is completely located outside of Lower Saxony. This is not the case for the validation and test datasets $\mathcal{T}_{val}^{L}$ and $\mathcal{T}_{te}^{L}$, which are located in different areas in the state of Lower Saxony, but are surrounded by tiles belonging to the training dataset $\mathcal{T}_{tr}^{L}$.

Dataset	$\mathcal{T}^{L}$	$\mathcal{T}_{tr}^{L}$	$\mathcal{T}_{val}^{L}$	$\mathcal{T}_{te}^{L}$	$\mathcal{T}_{cor}^{L}$	$\mathcal{T}_{kleve}^{L}$	Colour
Settlement (Stl.)	9.1	9.0	9.5	9.8	9.7	8.3	
Sealed area (Sld)	1.9	1.9	1.9	1.9	2.0	7.8	
Agriculture (Agr)	39.2	39.5	35.6	37.7	63.3	21.5	
Vegetation (Veg)	23.5	23.3	25.7	24.2	8.2	18.7	
Deciduous forest (Dfor)	7.0	7.1	6.3	6.2	10.9	15.0	
Coniferous forest (Cfor)	12.5	12.5	13.4	11.7	2.0	9.2	
Water (Wat)	5.5	5.4	5.8	6.9	3.2	14.2	
Barren land (Bar)	1.3	1.3	1.8	1.6	0.7	5.3	
Nb. tiles / points ($\mathcal{T}_{kleve}^{L}$)	885	813	36	36	2	1600	

Table 5.1: Class label distribution in [%] for the Lower Saxony and Kleve dataset. $\mathcal{T}^{L}$: Whole dataset, $\mathcal{T}_{tr}^{L}$: Training dataset, $\mathcal{T}_{val}^{L}$: Validation dataset, $\mathcal{T}_{te}^{L}$: Test dataset, $\mathcal{T}_{cor}^{L}$: Corrected test dataset, $\mathcal{T}_{kleve}^{L}$: Kleve test dataset. The last row shows the number of tiles (nb. tiles) belonging to the individual datasets of Lower Saxony and the total number of points for the Kleve dataset.

5.1.2 Dynamic Earth dataset

Toker et al. (2022) published the Dynamic Earth dataset, intending to provide daily satellite images in combination with high-quality labels. For 75 areas, distributed over the whole globe, daily satellite images are provided in combination with monthly labels over a time period of two years that covers 2018 and 2019. In the remainder of this thesis, these different areas are referred to as different tiles, similar to the Lower Saxony & Kleve datasets.

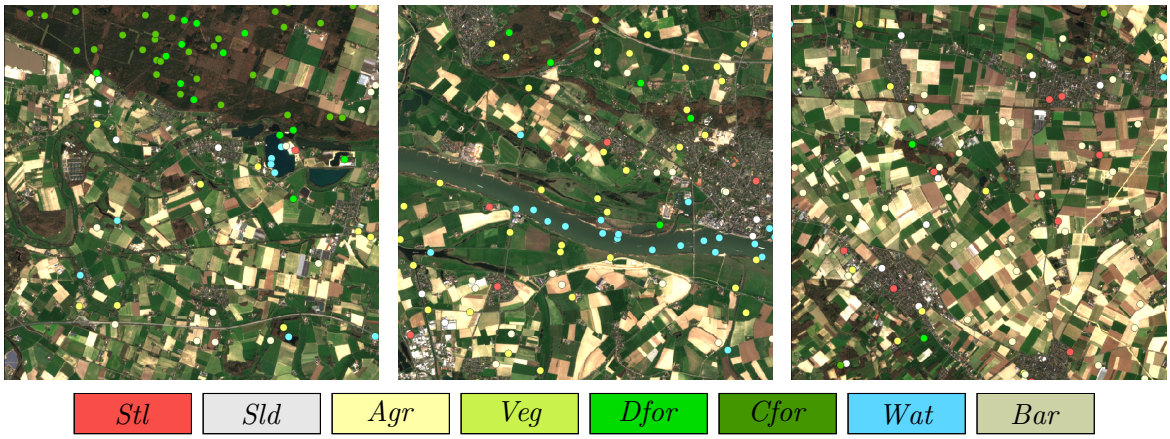

Figure 5.3: Exemplary Sentinel-2 images and label data (points) for the Kleve dataset. The shown images cover 8 km × 8 km (800 px×800 px - one tile).

Image data: The provided SITS are derived from the Fusion Monitoring product[1] from Planet Labs. The images contain four spectral bands (red, green, blue, and near-infrared) with a GSD of 3 m. Each of the 75 tiles contains 1024 × 1024 pixels, which is equivalent to 3072 × 3072 m on the ground. The images are pre-processed by Planet Labs, including the removal of clouds and artifacts. Gaps are filled with suitable observations from the closest acquisition in time. Additionally, the spectral bands are calibrated to the harmonized Landsat-Sentinel[2] spectrum, which makes them compatible with images from the Landsat or Sentinel satellites. In addition to the Planet images, monthly Sentinel-2 images are provided to encourage the usage of data fusion methods. However, these Sentinel-2 images are not further used in the context of this thesis.

Label data: Monthly pixel-wise annotations are provided, corresponding to the first day of each month. Seven land cover classes are defined: *Impervious Surfaces (imp)*, *Agriculture (agr)*, *Forest & vegetation (veg)*, *Wetlands (wet)*, *Soil (sol)*, *Water (wat)*, *Snow & ice (ice)*. The label maps were manually annotated for the first image of the whole timeseries for each tile. Afterwards, all following label maps are annotated based on the label map from the previous month by just annotating changes. To ensure accurate annotations, three annotators checked each label map before publication (Toker et al., 2022).

Train-test split: The separation of the whole dataset into training, validation, and test dataset used in this thesis is based on the same spatial split as introduced by Toker et al. (2022). The training dataset $\mathcal{T}_{tr}^{D}$ consists of 55 tiles, while the validation ($\mathcal{T}_{val}^{D}$) and test dataset ($\mathcal{T}_{te}^{D}$) consist of 10 tiles each. While some of the tiles contain only two of the introduced LC classes (in most cases, these tiles are covered mainly by vegetation), some tiles include up to six of the seven LC classes. The split into training, validation and test datasets is chosen in a way that the class distribution is as equal as possible in all three datasets. The class *Snow & ice* is not included in the validation or test sets and, therefore, this class is also not used during the training process. The distribution of the class labels in the different datasets is shown in table 5.2. The class distribution of the dataset

[1] https://www.planet.com/pulse/planet-announces-powerful-new-products-at-planet-explore-2020, accessed on 26-11-2024

[2] https://www.earthdata.nasa.gov/data/projects/hls, accessed on 26-11-2024

T^D slightly differs from the distribution presented in (Toker et al., 2022) because two of the tiles for training were added to the published dataset later. Similarly to the Lower Saxony dataset the class distribution is highly imbalanced.

Dataset	$\mathcal{T}^D$	$\mathcal{T}^D_{tr}$	$\mathcal{T}^D_{val}$	$\mathcal{T}^D_{te}$	Colour
Impervious Surfaces (Imp)	7.2	6.9	7.5	8.4	
Agriculture (Agr)	9.5	11.2	3.1	7.0	
Forest & vegetation (Veg)	45.5	44.6	59.0	36.9	
Wetlands (Wet)	0.7	0.3	2.9	0.6	
Soil (Sol)	28.1	28.0	20.9	35.6	
Water (Wat)	8.0	7.7	6.6	11.5	
Snow & ice (Ice)	1.0	1.3	0.0	0.0	
Nb. areas of interest	75	55	10	10	

Table 5.2: Class label distribution in [%] for the Dynamic Earth dataset. $\mathcal{T}^D$: Whole dataset, $\mathcal{T}^D_{tr}$: Training dataset, $\mathcal{T}^D_{val}$: Validation dataset, $\mathcal{T}^D_{te}$: Test dataset.

5.1.3 Discussion

The Lower Saxony dataset as well as the Dynamic Earth dataset provide SITS with pixel-wise annotations for LC for multiple timesteps and the method introduced in section 4.2 can be evaluated on them. Nevertheless, the datasets come with some differences that can have an impact on the training process and on the selection of some model parameters. The major differences are the covered area and the difference in the GSD. These characteristics are discussed in the following paragraph. The Kleve dataset is not included in this discussion, as the provided labels are based on points for one timestep only.

The Lower Saxony dataset $\mathcal{T}^L$ covers an area of 47600 km^2 while the Dynamic Earth dataset $\mathcal{T}^D$ only covers a total area of 735 km^2 if all 75 tiles are combined, which is only about 1.6% of $\mathcal{T}^L$. At the same time, the tiles of $\mathcal{T}^D$ are spread all over the globe, resulting in a larger diversity of appearances, even within the same LC class (cf. figure 5.4). In combination, these properties are expected to result in a more difficult training process for the Dynamic Earth dataset, caused by the much smaller number of available tiles and the higher variability within the training tiles, but also between the training and test tiles. This is expected to result in lower accuracy scores in comparison to the Lower Saxony dataset.

The Lower Saxony dataset is based on Sentinel-2 images with a GSD of 10 m, whereas the Dynamic Earth dataset is based on Planet Labs images that have a GSD of 3 m. It is clearly visible, e.g. by comparing the exemplary Planet images in figure 5.4 with the exemplary Sentinel-2 images in figure 5.3, that finer details, such as building or street shapes, are more clearly defined in the Planet images. For the Sentinel-2 images, object boundaries will often fall into the 10 m $\times$10 m covered by a single pixel, resulting in mixed pixels, i.e. pixels that contain more than one of the defined LC classes. Mixed pixels also appear in the Dynamic Earth dataset, however, due to the smaller GSD of 3 m much less pixels are affected. These mixed pixels are expected to result in some difficulties to assign those pixels to the correct class and can result in a lower performance

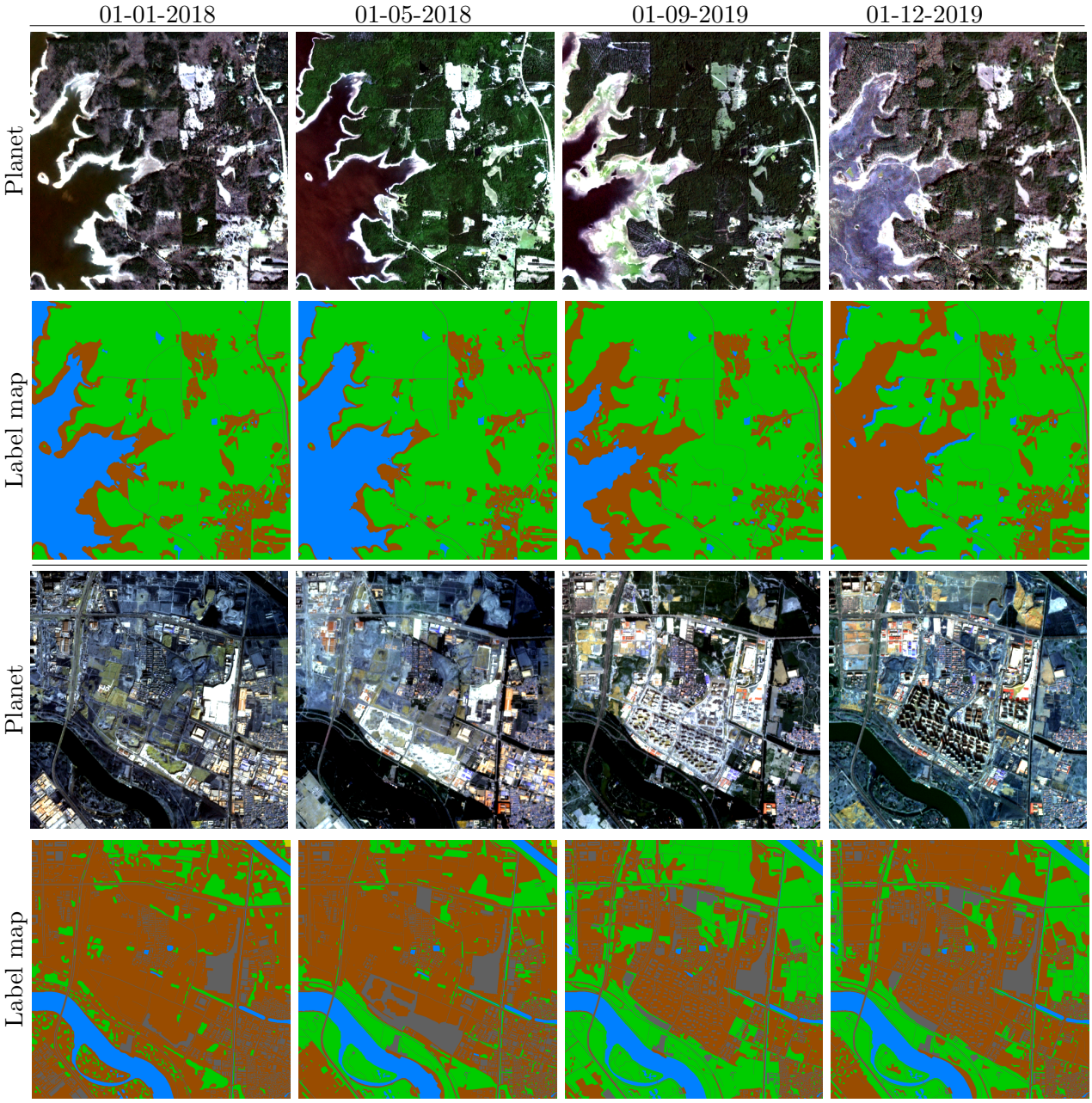

Figure 5.4: Exemplary image and label data for the Dynamic Earth Net dataset. The data correspond to two of the 75 tiles, each having a size of 1024× 1024 pixels at a GSD of 3 m. The colors correspond to the different LC classes introduced in table 5.2.

in these regions and for thin objects. A second aspect of the different GSDs is the size of the area in object space that corresponds to the receptive field of a used model. With a similar size of the receptive field in terms of observed pixels in the input images for the Dynamic Earth dataset, the observed area in object space would only cover about 10% of the area covered in the Lower Saxony dataset. Therefore, a larger receptive field might be beneficial for the Dynamic Earth dataset. Consequently, the ablation studies for this dataset include the usage of more model stages and larger convolutional kernels, which both increase the receptive field of the model.

5.2 Objectives and structure of the experiments

To answer the research questions formulated in section 1, different series of experiments are conducted, which are described in this section. First, the general training setup and the hyper-parameter studies are described in section 5.2.1. The hyper-parameters are set for the Lower Saxony dataset and the Dynamic Earth dataset separately and are then fixed for all experiments. Afterwards, the experiment set designed for analysing the new hybrid feature extraction module FE_{hyb} is presented in section 5.2.3. In section 5.2.4 the experiments regarding the convolutional patch generation are described, before in section 5.2.5 the experiment set regarding the temporal weighting module in the skip connections is introduced. Finally, the setup of the experiments conducted to compare the new model to other approaches is described in section 5.2.6.

5.2.1 General training setup and parameter studies

The general training procedure follows the strategy described in section 4.3. While several hyper-parameters are set empirically by testing different parameter values and evaluating the classification performance on the validation datasets, other hyper-parameters are set based on preliminary experiments (Voelsen et al., 2022, 2023; Voelsen et al., 2024). The remaining hyper-parameters are selected based on suggestions from related works or are dictated by limitations regarding the hardware. Note that all hyper-parameter studies reported in this section were conducted before the experiments for answering the research questions were performed. In the latter, the best parameter settings based on the hyperparameter tuning experiments were used.

During training, the input images are randomly cropped windows of size $(H_0, W_0) = (256, 256)$ pixels that are chosen from all available training tiles as explained in section 4.3. For all experiments in this thesis, a number of $T = 12$ timesteps and a number of $B = 4$ spectral bands are used. Using twelve timesteps resulted in a good trade-off between classification performance and training time, as previous experiments have shown improved performance when using more timesteps, but also an increase in the time for training. Data augmentation is applied for the whole timeseries during the training process, applying random rotations by 90°, 180°, 270° and horizontal and vertical flipping. During training, the ADAM optimizer (Kingma and Ba, 2015) is used with the parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Training is conducted for a maximum number of $n_e = 100$ epochs and early stopping is used when the validation accuracy is not improving for $n_{es} = 10$ epochs. Each epoch consists of a fixed number n_{it} of iterations, each considering a minibatch of mb input timeseries. Due to limitations of the available GPU resources, the minibatch size was set to $mb = 2$. Due to the large difference in the size of both training datasets, the number of iterations per epoch is set individually for the datasets.

For the Lower Saxony dataset, a number of $n_{it} = 5000$ is chosen, while for the Dynamic Earth dataset, n_{it} is set to $n_{it} = 1000$. In this way, the accuracy on the validation datasets are computed for both datasets, before too many images are used for updating the weights multiple times. Some parameters were found in preliminary experiments. The parameter κ of the adaptive cross-entropy loss (cf. equation 4.8) is set to $\kappa = 1$ and the parameter for the temporal encoding is set to $\tau = 10000$. Regarding the proposed model, the feature dimension of the UPer-Net decoder is set to

$C_{dec} = 512$ and the dropout rates are set to $adr = 0.4$ for the attention dropout and to $sdr = 0.2$ for the stochastic depth dropout. Furthermore, the patch size was set to $P = 4$, which is adapted from the Swin Transformer (Liu et al., 2021). The number n_{emb} of 3D convolutional layers in the $P/2$ patch generation blocks in the 3D-PG module is set to $n_{emb} = 2$, based on preliminary experiments with this hyper-parameter. The remaining parameters are set individually for both datasets, which is discussed in the following paragraph. This includes the initial learning rate $\eta_{e=0}$, with $e = 0$ indicating the learning rate for the first epoch, the factor η_f which is multiplied with the learning rate every ten epochs to slowly reduce the learning rate $\eta_e = \eta_f \cdot \eta_{e-1}$, the activation function, the number of stages S and the input feature dimension C_{in}.

Parameter studies regarding the Lower Saxony dataset: For the Lower Saxony dataset only a small set of hyper-parameter studies is conducted, as this dataset was already used in previous experiments (Voelsen et al., 2022, 2023; Voelsen et al., 2024) and most values are taken from these experiments. For the remaining hyper-parameters several values were tested. The tested hyper-parameters include the initial learning rate $\eta_{e=0}$, the factor for the learning rate η_f, the used activation function for the convolutional layers, different combinations of the number of used stages S in combination with the number of modules per stage L_S, as well as the input feature dimension C_{in}. An overview of all tested values for these hyper-parameters is given in table 5.3, with the best-performing values regarding the mF^1-scores on the validation dataset T^L_{val} indicated in bold.

Parameter	Symbol	Tested values (best in **bold**)
Learning rate	$\eta_{e=0}$	$3 \cdot 10^{-5}$; $6 \cdot 10^{-5}$; $12 \cdot 10^{-5}$; $\mathbf{2 \cdot 10^{-4}}$
Factor for learning rate	η_f	0.7; **1.0**
Activation function	-	ReLU; **lReLU**; GELU
Number of stages, modules per stage	S, L_s	$\mathbf{S = 2}$ with $[L_1\text{-}L_2] = [2, 2]; [2, 6]; [\mathbf{6}, \mathbf{2}]; [6, 6]$
		$S = 3$ with $[L_1\text{-}L_3] = [2, 2, 6]; [\mathbf{2}, \mathbf{6}, \mathbf{2}]$
Input feature dimension	C_{in}	48; **96**

Table 5.3: Hyper-parameter studies for the Lower Saxony dataset $\mathcal{T}^L$. Best parameter values based on the mF^1-score on $\mathcal{T}^D_{val}$ are indicated in **bold**. L_s denotes the number of self-attention blocks in stage s, i.e. $[L_1\text{-}L_S] = [2, 4]$ corresponds to a model with two stage ($S = 2$) with two Fe_{hyb} modules in stage 1 and four modules in stage 2.

Parameter studies regarding the Dynamic Earth Net dataset: The Dynamic Earth dataset comes with some differences compared to the Lower Saxony dataset as discussed in section 5.1.3. Due to these differences, it is expected that several hyper-parameters should differ from these for the Lower Saxony dataset. Additionally, this dataset was not used in any previous experiments. Therefore, a larger number of hyper-parameter studies was conducted to find the best parameter settings regarding the mF^1-score on the validation dataset $\mathcal{T}^D_{val}$. The initial values regarding the model are taken from the experiments of the Lower Saxony dataset and those for the training process (e.g. learning rate) are taken from Toker et al. (2022). For each parameter setting, only the investigated parameter is changed while all others stay constant. In addition to the hyper-parameters that are studied for the Lower Saxony dataset, for the Dynamic Earth dataset the parameter κ of the adaptive cross-entropy loss, the kernel size k used in the convolutional layers, and the number of

iterations per epoch n_{it} are analysed. An overview of the hyper-parameter studies is shown in table 5.4, with the best-performing values on the validation dataset T^D_{val} indicated in bold.

Parameter	Symbol	Tested values (best in **bold**)
Param. of adaptive Cr. Entr. loss	κ	**1**; 3; 5
Learning rate	$\eta_{e=0}$	$3 \cdot 10^{-5}$; $\mathbf{6 \cdot 10^{-5}}$; $12 \cdot 10^{-5}$
Factor for learning rate	η_f	0.5; **0.7**; 1.0
Activation function	-	ReLU; **lReLU**; GELU
Number of stages, modules per stage	S, L_s	$S = 2$ with $[L_1\text{-}L_2] = [2, 2]$; $[\mathbf{2}, \mathbf{6}]$; $[6, 2]$
		$S = 3$ with $[L_1\text{-}L_3] = [4, 4, 12]$; $[\mathbf{2}, \mathbf{2}, \mathbf{6}]$; $[2, 2, 18]$
		$[2, 6, 18]$; $[2, 2, 24]$
		$\mathbf{S = 4}$ with
		$[L_1\text{-}L_4] = [2, 2, 2, 2]$; $[2, 2, 6, 2]$, $[2, 2, 18, 2]$
		$[2, 2, 12, 6]$; $[2, 2, 12, 2]$; $[2, 2, 2, 6]$
		$[2, 2, 12, 12]$; $[2, 2, 18, 6]$; $[\mathbf{2}, \mathbf{2}, \mathbf{6}, \mathbf{6}]$
kernel size (convolutions)	k	3; **5**
Iterations per epoch	n_{it}	250; **1000**; 5000
Input feature dimension	C_{in}	**48**; 96

Table 5.4: Hyper-parameter studies for the Dynamic Earth Net dataset $\mathcal{T}^D$. Best parameter values based on the mF^1-score on $\mathcal{T}^D_{cal}$ are indicated in **bold**. L_s denotes the number of self-attention blocks in stage s,i.e. $[L_1\text{-}L_S] = [2, 4]$ corresponds to a model with two stage ($S = 2$) with two Fe_{hyb} modules in stage 1 and four modules in stage 2.

Discussion: An overview of all hyper-parameter values for both datasets is given in table 5.5. In most cases, the differences in the achieved validation accuracies between the tested hyper-parameter values are small. An exception is the usage of four stages ($S = 4$) in combination with the number of modules per stage $L_S = [2, 2, 6, 6]$ on the Dynamic Earth dataset, which performs significantly better than using two stages. This can be explained by the increase in the receptive field, which is equivalent to 24 m×24 m on the ground for $S = 2$ and to 96 m×96 m on the ground for $S = 4$ for $\mathcal{T}^D$. In contrast, the receptive field for the Lower Saxony dataset is already 80 m×80 m for two stages ($S = 2$), which leads to the best results on $\mathcal{T}^L$. These results meet the expectation that a larger receptive field, based on the number of pixels that contribute to one feature in the deepest stage, is beneficial for the Dynamic Earth dataset, as discussed in section 5.1.3. If this receptive field is converted into the spatial area on the ground, it is in a similar range.

The selected hyper-parameter values are used for the experiments in section 6.2 - 6.4. For most hyper-parameters, the values that performed best on the validation dataset are selected (cf. tables 5.3 and 5.4). The combination of all parameter values that achieve the best performance results in an increase in the computational complexity, because for most of the tested hyper-parameters the values that result in a larger model perform better (e.g. more stages or a larger kernel size for the Dynamic Earth dataset). This increase in the computational complexity results in longer training times and the fact that the model could not be trained on some of the available GPU resources anymore. Therefore, the selected value for the number of stages S is reduced from $S = 4$ with $L_S = [2, 2, 6, 6]$ to $S = 3$ with $L_S = [2, 2, 6]$ for the Dynamic Earth dataset, which performs

marginally worse (-0.1% in the mF^1-score) than the model with four stages. For the Lower Saxony dataset the number of modules is reduced from $S = 2$ with $L_S = [6, 2]$ to $S = 2$ with $L_S = [2, 2]$, with the latter performing slightly worse (0.5% in the mF^1-score) than the best variant. For the comparison of the new model to the baseline variants (section 6.5), the models with the best parameter values for S and L_S are additionally shown for the proposed method.

	Parameter	Symbol	Value	
			$\mathcal{T}^L$	$\mathcal{T}^D$
Input	Nr. of classes	n_c	8	6
	Input size	H_0, W_0, B, T	$H_0 = 256, W_0 = 256, B = 4, T = 12$	
	Minibatch size	mb	2	
Training	Learning rate	$\eta_{e=0}$	$2 \cdot 10^{-4}$	$6 \cdot 10^{-5}$
	Factor for learning rate	η_f	1.0	0.7
	Iterations per epoch	n_{it}	5000	1000
	Total nr. of epochs	n_e	100	
	Nr. of epochs for early stopping	n_{es}	10	
	Adam	β_1, β_2	$\beta_1 = 0.9, \beta_2 = 0.999$	
	Param. of adaptive Cr. Entr. loss	κ	1	
	Nr. of mb for class weights	n_ω	100	
	Nr. of epochs for class weights	$n_{e\omega}$	10	
Model	Nr. of stages, modules per stage	S, L_S	$S = 2,\ L_S = [2, 2]$	$S = 3,\ L_S = [2, 2, 6]$
	Nr. of heads in the stages	n_{h_s}	$[3, 6]$	$[3, 6, 12]$
	Kernel size (convolutions)	k	3	5
	Input feature dimension	C_{in}	96	48
	Patch size	P	4	
	Nr. of 3D conv. in PE module	n_{emb}	2	
	Activation function	-	lReLU	
	Parameter initialization	-	Variance scaling	
	Feature dimension UPer-Net	C_{dec}	512	
	Attention dropout	adr	0.4	
	Stochastic depth dropout	sdr	0.2	
	Param. for temporal encoding	τ	10000	

Table 5.5: Hyper-parameter values for both datasets that are used for the experiments.

5.2.2 Experiment set 1: Proposed method for multi-temporal LC classification

In the first set of experiments, the proposed model for multi-temporal LC classification is analysed in detail to answer the first research question stated in section 1:

1. How does the proposed model, which incorporates temporal context, perform for the task of multi-temporal land cover classification? Is the model able to detect class changes over time, even if it is not explicitly trained for change detection?

To answer research question 1, the proposed model, which will be referred to as $V\text{-}FE_{hyb}$, is trained and evaluated on the available datasets. The $V\text{-}FE_{hyb}$ model is based on the FE_{hyb} module in all stages of the model, which means that spatial and temporal context is considered separately and the outputs are combined afterwards (cf. section 4.2.6.1). Convolutions are used to consider spatial context, while temporal context is encoded based on self-attention. The proposed spatio-temporal PG module based on 3D convolutions (cf. section 4.2.4) is used for patch generation and the temporal weighting module is used in the skip connections (cf. section 4.2.7).

The evaluation of the proposed model is structured into two parts. First, the accuracies obtained for multi-temporal LC classification are analysed in detail to be able to assess the model performance on the different datasets, which is related to the first part of research question 1. Afterwards, the prediction of LC changes is evaluated by exemplary qualitative and quantitative results with the goal to answer the second part of research question 1. As the datasets only contain a small percentage of change, some areas that include a higher percentage are manually selected for this comparison. Changes of LC are considered implicitly in the model in the form of cases in which the used LC maps in the training datasets show different classes for the same pixels on the ground for different timesteps. However, the classification of changes is not explicitly considered in the model or in the training process, because the loss function is only computed based on the predicted LC maps. This analysis shall help to show in which scenarios LC changes are already classified by the model and in which cases there are problems.

5.2.3 Experiment set 2: Evaluation of the hybrid feature extraction module

The second set of experiments focuses on evaluating the performance of the proposed hybrid spatio-temporal feature extraction module, FE_{hyb}, introduced in section 4.2.6.1. The analysis of this module is related to research question 2:

2. Does the proposed hybrid feature extraction module, considering spatial context with convolutions and temporal context with self-attention, lead to better results than modules solely relying on self-attention for feature extraction?

To answer this research question, two model variants are introduced that serve as comparative variants to the proposed method $V\text{-}FE_{hyb}$. Both variants are described in the following.

The first aspect that is analysed in this experiment set is the usage of convolutional layers in the spatial feature extraction stream of the FE_{hyb} module. Specifically, it shall be investigated if convolutional layers are better suited for the encoding of spatial context than self-attention layers. To be able to analyse this aspect, variant $V\text{-}FE_{att}$ is introduced. $V\text{-}FE_{att}$ uses the FE_{att} module (cf. section 4.2.6.2) instead of the FE_{hyb} module in all stages of the model. In the FE_{att} module, spatial context is encoded based on the W-MSA, which combines the shifted window approach from Liu et al. (2021) and the multi-head self-attention from Vaswani et al. (2017). In the temporal stream of both modules multi-head self-attention layers are used to encode temporal context.

The separation into the spatial and temporal streams has the advantage that the computational complexity is drastically reduced in comparison to a strategy that jointly extracts spatial and

temporal context. This raises the question of how a model that jointly extracts spatio-temporal features in a single computation performs. To be able to analyse this aspect, another variant, which is referred to as $V\text{-}FE_{full}$, is introduced. In $V\text{-}FE_{full}$, spatio-temporal features are jointly computed by self-attention. To achieve this, all FE_{hyb} modules are replaced by FE_{full} modules, which are described in section 4.2.6.3. In the FE_{full} module, dependencies between patches from different acquisition dates and different spatial positions are additionally taken into account, which is not the case in the proposed model $V\text{-}FE_{hyb}$ or variant $V\text{-}FE_{att}$. By comparing the classification performance of the variants $V\text{-}FE_{att}$ and $V\text{-}FE_{full}$, the impact of these additional dependencies can be analysed because, apart from this change, the variants are completely the same. By using the FE_{full} module, the number of patches between which self-attention is computed increases to $M^2 \cdot T$. This affects the training and inference time that is also analysed in the evaluation of this model variant. Variants $V\text{-}FE_{hyb}$ and $V\text{-}FE_{full}$ cannot be compared directly to answer research question 2 because differences in the classification performance can be caused by one of two differences: First, FE_{hyb} separates spatial and temporal feature analysis. Second, FE_{hyb} uses convolutions instead of self-attention in the spatial dimension of the input data.

5.2.4 Experiment set 3: Evaluation of the convolutional patch generation

In the third set of experiments, the impact of the spatio-temporal patch generation module introduced in section 4.2.4 is analysed. This analysis allows to answer the third research question related to the 3D patch generation module, which is formulated as follows (cf. section 1.2):

3. Does the proposed patch generation module, utilizing 3D convolutions to encode spatio-temporal dependencies, achieve better performance compared to standard patch generation strategies such as the one in the Vision Transformer (Dosovitskiy et al., 2021) or when compared to using 2D convolutions to consider spatial context only?

The usage of the introduced patch generation module allows to use additional skip connections at higher spatial resolution than the one after the patch generation process ($H/P, W/P$), because spatial and temporal dependencies are already extracted at higher spatial resolution, which can be used in the corresponding decoder stages. This aspect is addressed by research question 4:

4. Does an additional skip connection at a higher spatial resolution than the one of the feature maps after patch generation lead to improved classification performance?

To answer these research questions, the 3D patch generation module, which is used in the proposed model $V\text{-}FE_{hyb}$, is adapted, resulting in four additional model variants. $V\text{-}FE_{hyb}$ serves as the baseline to which these variants are compared. The patch size of the $V\text{-}FE_{hyb}$ model is set to $P = 4$, resulting in spatial dimensions of size $(H_0/4, W_0/4)$ for the feature maps after the 3D patch generation module is applied. In combination with the number of stages S, defined in the hyper-parameter studies for each dataset (cf. section 5.2.1), this results in a specific receptive field. In all variants that are introduced in the following, the receptive field is the same as for model $V\text{-}FE_{hyb}$. This means that the spatial extent after patch generation is $(H_0/4, W_0/4)$.

The first adapted model variant for this experiment set is based on the standard patch generation strategy used by Dosovitskiy et al. (2021) in the Vision Transformer and by Liu et al. (2021) for the Swin Transformer. This variant is referred to as $V\text{-}PE_{swin}$. The used standard strategy for patch generation is explained in detail in section 2.2.2.1. It is based on stacking all pixel values within a patch (e.g. 4×4 pixels if $P = 4$). A linear layer transforms this input to a feature vector of dimension C_{in} for each patch. The main reason to perform this reduction in the spatial dimension is a reduction of the computational complexity, but, the positions of fine details are not preserved with this operation, which leads to difficulties to predict the exact position of class borders. This challenge was already discussed in earlier works, for instance, by Strudel et al. (2021), who investigate the effect of using different patch sizes. By comparing the results of variant $V\text{-}PE_{swin}$ to the proposed model $V\text{-}PE_{hyb}$, the influence of extracting spatio-temporal features by 3D convolutions can be investigated, which answers the first part of research question 3.

The second adaptation introduced for this experiment set uses 2D convolutions in a patch generation module that is otherwise equally structured as the 3D PG introduced in section 4.2.4. This model variant is referred to as $V\text{-}PE_{2D}$. In this variant, 2D convolutions are applied in the two spatial dimensions of the input images. This is done in parallel for all input timesteps with shared weights. Using this variant, it can be answered if the usage of 3D convolutions in model $V\text{-}PE_{hyb}$ leads to an improved classification performance compared to only using 2D convolutions in the patch generation module. This analysis answers the second part of research question 3.

The third model variant of this experiment set uses the 3D patch generation module of $V\text{-}PE_{hyb}$, but with an additional skip connection that connects the patch generation module with the last decoder layer of the model. This model variant is referred to as $V\text{-}PE_{skip}$. Due to hardware limitations, the additional skip connection is integrated at the spatial resolution of $(H/2, W/2)$ and not at the original spatial resolution (H, W), which is depicted in figure 5.5 (green arrow). For the new skip connection, the features after the last 3D convolutional layer (including BN and activation function), but before the final downsampling step, are used. For the used patch size of $P = 4$, these features have a spatial dimension of $(H/2, W/2)$. To integrate these features into the last decoder layers (cf. table 4.1), the upsampling process from $(H/P, W/P)$, i.e. in this case $(H/4, W/4)$, is split into two upsampling steps to be able to combine the encoder and decoder features at the spatial resolution of $(H/(P/2), W/(P/2))$, i.e. in the case of $P = 4$ of $(H/2, W/2)$. For this purpose, the output of layer CB1 (cf. table 4.1) in the output generation process of the decoder serves as input to the first upsampling layer with a factor of 2. Afterwards, the new skip connection is applied by concatenating these features, having a dimension of C_{dec}, with those from the patch generation module. This process results in features with a dimension of $T \times (C_{dec} + C_{in}) \times H/2 \times W/2$. After the skip connection, the combined features are processed by another convolutional layer with a kernel size of $k = 3$, followed by BN and activation function before a final convolutional layer with $k = 1$ maps the features to n_c classes. Similarly to variant $V\text{-}PE_{hyb}$, the feature maps are then upsampled to the original spatial resolution and normalised by the softmax function as depicted on the right side of figure 5.5. By comparing variant $V\text{-}PE_{skip}$ with the proposed model, research question 4 can be answered.

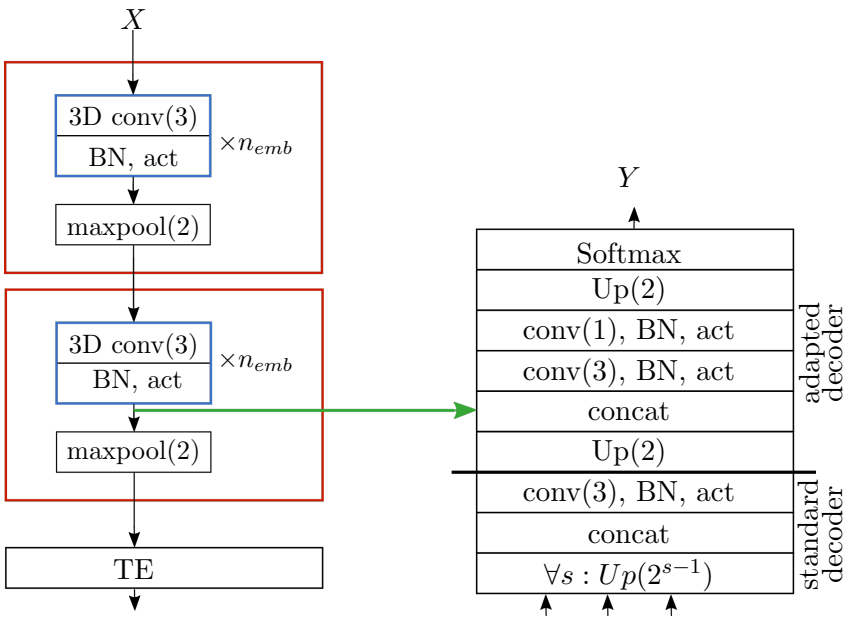

Figure 5.5: Overview of how the additional skip connection (green arrow) is integrated into the output generation of the decoder for a patch size of $P = 4$. 3D conv(k): 3D convolution with kernel size k, maxpool(f): Maximum pooling by factor f, TE: Temporal position encoding. The standard decoder layers are the same as the first three layers of the output generation in table 4.1, the adapted decoder splits the upsampling process for the new skip connection, i.e. instead of upsampling directly by a factor of P, two upsampling layers with a factor of $P/2$ are applied.

Using the fourth and last model adaptation of this experiment set, the patch generation and its impact on the classification performance is analysed on a general level. This is done by using a variant without any patch generation. Instead, a patch size of $P = 1$ is used. This model variant is referred to as $V\text{-}PE_{P1}$. Due to limitations in the available GPU resources, the input image size is reduced to $(H_0, W_0) = (64, 64)$ for this model variant, which is a reduction by a factor of four and, therefore, equivalent to the feature map size after the patch generation modules for all other variants. However, the receptive field is smaller by a factor of the patch size in height and width, i.e. a factor of 4 with the defined patch size of $P = 4$, which will have an impact the classification results. Furthermore, if no patches are generated, the number of input feature vectors to the FE_{hyb} modules is larger by a factor of P^2 compared to a model using patch generation, which will significantly increase the time to process the same amount of images. To counteract this effect in the training process, the number of used image timeseries in one epoch is not increased, resulting in an area covered by the images used in one training epoch that is four times smaller.

5.2.5 Experiment set 4: Evaluation of the temporal weighting module

In the fourth set of experiments the temporal weighting module, which is used in the skip connections of the proposed model $V\text{-}PE_{hyb}$ (cf. section 4.2.7), is evaluated. The temporal weighting is based on the weighting strategy combining features extracted in the spatial and temporal dimensions of the input timeseries, introduced by Garnot and Landrieu (2020). They use it in the bottleneck layer to weight the features considering spatial context with those considering temporal context. In practice, the idea of weighting is realised by an element-wise matrix multiplication

that is interpreted as using the features considering temporal context as weights. However, this operation can also be interpreted the other way around, meaning that also the features considering spatial context can be interpreted as weights. In the proposed model $V\text{-}FE_{hyb}$, this weighting strategy is extended to all stages of the model, because temporal and spatial dependencies are considered anyway in all encoder stages. The analysis of the temporal weighting module uses the model $V\text{-}FE_{hyb}$ as a baseline and shall answer the following research question:

5. Does integrating the proposed temporal weighting module into the skip connections throughout all model stages enhance the classification performance compared to not using any temporal weighting?

To be able to answer research question 5, a variant that does not use any temporal weighting module is introduced. This variant is referred to as variant $V\text{-}TW_{no}$. Instead of weighting the temporal and spatial features, in variant $V\text{-}TW_{no}$, the fused output features A_s of the FE_{hyb} module of the corresponding stage are used as input to the skip connections. These input features are then processed based on the standard strategy in the UPer-Net decoder. This involves a convolutional layer that maps the features from the encoder to C_{dec} feature dimensions before the features from the encoder and decoder are combined by element-wise addition. By evaluating this variant in comparison to $V\text{-}FE_{hyb}$, the impact of the temporal weighting module can be analysed and research question 5 can be answered. Note that except for the temporal weighting module, the same setting as for the $V\text{-}FE_{hyb}$ variant is used for $V\text{-}TW_{no}$.

5.2.6 Experiment set 5: Comparison to other works

In the experiments introduced in sections 5.2.1 - 5.2.5, different components of the proposed model are evaluated by only changing specific modules in the architecture. In the fifth and last set of experiments, the proposed model is compared to other architectures from the literature that can be used for SITS classification. The requirement of such methods is the possibility to handle multi-temporal input images and generate multi-temporal output maps. The goal of the conducted experiments in this section is to answer the last research question stated in section 1:

6. Does a variant of the proposed methods outperform approaches from literature, specifically those using a multitemporal FCN (Voelsen et al., 2023), the Utilise model presented by Stucker et al. (2023) and a U-Net (Toker et al., 2022)?

The first method used for comparison is a multi-temporal FCN that was already used as a baseline in (Voelsen et al., 2022) and (Voelsen et al., 2023). Additionally, the Utilise model from Stucker et al. (2023) is adapted to a classification task. Finally, the accuracy scores that Toker et al. (2022) achieved with a U-Net architecture for the Dynamic Earth dataset are compared to the proposed model. For the U-Net, several differences regarding the input and the training process occurred in comparison to the training procedure in this thesis, which limits the conclusiveness of the obtained results. The details about the three methods from literature are described in the following.

Multitemporal FCN: The multi-temporal FCN that serves as a baseline model for the comparison is purely based on convolutional layers. It was introduced in (Voelsen et al., 2023) for the

task of LC classification and was already used for experiments on the Lower Saxony dataset. It has to be mentioned that the Lower Saxony dataset in (Voelsen et al., 2023) has a slightly different set of classes and distribution of the tiles in the training and test datasets because the class structure was rearranged later. For that reason, the results of Voelsen et al. (2023) cannot be used directly for comparison, so that the FCN was trained again on both datasets.

Regarding the FCN training setup, the best values from Voelsen et al. (2023) were adopted. This includes the learning rate $\eta_{e=0} = 0.001$, the reduction factor for the learning rate $\eta_f = 0.7$ and the number of input features $C_{in} = 64$. The weighted cross entropy loss with $\kappa = 1$ and the ADAM optimizer is used. The number of stages S is chosen in a way that the receptive field is similar to the one of the proposed model for the corresponding dataset. As the FCN does not have any patch generation module at the beginning and the spatial resolution is reduced by a factor of two between subsequent stages, two additional stages are applied to compensate the patch generation with a patch size of $P = 4$. This results in a number of $S = 4$ stages for the Lower Saxony dataset, which results in feature vectors at stage 4 that represent 8×8 pixels (80 m×80 m) from the input images. For the Dynamic Earth dataset and a number of $S = 5$ stages is used, which means that one feature vector represents 16×16 pixels (48 m×48 m) from the input images.

***Utilise*:** The second model used for a comparison is *Utilise* based on (Stucker et al., 2023) (cf. section 2.2.2.4. Stucker et al. (2023) introduced it for cloud removal from SITS data, which requires multi-temporal input and output images. From the model point of view, the major difference is that the solved task is a regression task as the output maps contain predicted grey values for occluded pixels. Stucker et al. (2023) published the code of *Utilise*[3] and in the context of this work, the code is adapted for the task of LC classification. To adapt this model for multi-temporal LC classification, the last convolutional layer of the Utilise model is modified to transform the feature maps into n_c raw class scores, which is followed by the softmax function to obtain normalised class scores. The ADAM optimiser is used with a initial learning rate of $\eta_{e=0} = 6 \cdot 10^{-5}$.

The *Utilise* model is purely based on convolutions in the encoder and decoder that are applied successively by using the same weights for all input timesteps. The L-TAE module is used in the bottleneck layer to consider temporal dependencies in the timeseries. The comparison between the Utilise model and the $V\text{-}FE_{hyb}$ model makes it possible to analyse the aspect of extracting temporal context in all stages of the model. In the *Utilise* model, temporal dependencies are computed only at the coarsest spatial resolution, which are then upsampled to all spatial resolutions, which are then used in the skip connections. Note that the weighting applied in the L-TAE module and the temporal weighting module in this thesis is only used at the coarsest resolution in the Utilise model, which is different from the proposed model $V\text{-}FE_{hyb}$.

U-Net (Toker et al., 2022): Toker et al. (2022) apply several models to the Dynamic Earth dataset to introduce baselines for different tasks that can be solved with this dataset. Besides some models for change detection, they also test different models for supervised pixel-wise classification for SITS data. The only model that is trained with similar input data as the method proposed in this thesis is a U-Net architecture (Ronneberger et al., 2015) trained with monthly SITS data. In

[3]https://github.com/prs-eth/U-TILISE, accessed on 14-04-2025

contrast to the method proposed in this thesis, the generated output map is mono-temporal, as for the whole input timeseries one output label map is generated. During training, the first image from each month is used and combined with the corresponding reference, but the authors do not explain in detail to which label map the predicted mono-temporal map is compared. This can have an effect on the results in cases where the LC class changes with time as changes appear in 5% of the pixels in the Dynamic Earth dataset (Toker et al., 2022); this is already a relatively high percentage of class change. Furthermore, Toker et al. (2022) do not mention if the whole available timeperiod of two years is used to generate one input timeseries or if it is divided into several smaller timeseries, which can have an impact on the performance of the model.

To be able to compare the results of the U-Net used in (Toker et al., 2022) to those achieved by the method proposed in this thesis, the multi-temporal output maps are combined to one output map by a majority voting strategy. This only applies to the computation of the performance for inference; the training process remains unchanged. To generate the corresponding mono-temporal label map, only the pixels that do not change within the timeseries are used to compute the accuracy metrics. As Toker et al. (2022) only publish the class-wise accuracy scores on the validation dataset and only the mean scores for the test dataset, the discussion of these results focuses on the validation data. Overall, this comparison cannot answer any research question as the training and inference process contains some differences that can significantly impact the outcome. However, the results can show if the performance of the U-Net and the proposed method point to the same direction.

5.3 Evaluation

After training, the performance of the model is obtained by comparing the predicted label maps $\hat{Y} \in \mathbb{R}^{T \times H_0 \times W_0}$ to the reference labels $Y \in \mathbb{R}^{T \times H_0 \times W_0}$ from the test datasets. For all datasets, the provided images are larger in height and width than the input size to the model (H_0, W_0). For this reason, a sliding window approach is applied, using a horizontal and vertical shift of $(H_0/2, W_0/2)$ pixels. This means that after classifying the first input X with spatial size (H_0, W_0), e.g. starting in the upper left corner of the image, the window is shifted by $H_0/2$ and the next input is classified. This results in redundant predictions per pixel in the areas where the windows overlap. The probabilistic class scores for the same pixel in these areas are averaged before the class with the largest probability is chosen as the final class prediction. Note that the whole process is based on multi-temporal input images X, i.e. the acquisition dates for each temporal interval are chosen based on the strategy introduced in section 4.1.1. The accuracy metrics that are explained in the following are based on the comparison of Y and $\hat{Y}$, resulting in T predictions and corresponding labels for each pixel at one spatial position that are all used to compute the accuracy metrics.

Based on the obtained classification results, a confusion matrix is generated by comparing each class prediction (i.e., for each pixel and each timestep) with the reference from the corresponding test dataset. From the confusion matrix, the number of pixels that correspond to true positive (TP_c), false positive (FP_c), true negative (TN_c) and false negative (FN_c) predictions can be computed for each class C^c. From these values, several quality indicators can be determined. One global metric is the Overall Accuracy (OA) that is defined as the percentage of pixels that were

assigned to the correct class. If the class distribution is imbalanced, as it is the case for all datasets used in this thesis (cf. tables 5.1 and 5.2), the OA is biased towards classes with more samples, as each pixel is weighted similarly. For this reason, the F1-Score (F_c^1) is computed as a metric that is computed for each class C^c separately:

$$F_c^1 = 2 \cdot \frac{precision_c \cdot recall_c}{precisision_c + recall_c} \tag{5.1}$$

with

$$precision_c = \frac{TP_c}{TP_c + FP_c}, \quad recall_c = \frac{TP_c}{TP_c + FN_c}. \tag{5.2}$$

The *precision* describes the percentage of the predictions assigned to a class that belongs to that class in the corresponding reference and the *recall* is defined as the percentage of the samples that belong to a class in the reference that are classified correctly. From the individual F^1-scores, the mean F^1-score (mF^1) over all classes is obtained as a second global metric. The mean F^1-score is not influenced by the class imbalance, because the impact of F_c^1 on mF^1 is equal for all classes, independently from the number of pixels corresponding to that class.

For the sake of a comparison to other methods, the Intersection over Union (IoU) is also reported. The IoU is another performance metric that is based on the confusion matrix and is computed for each class individually based on the following equation:

$$IoU_c = \frac{TP_c}{TP_c + FN_c + FP_c}. \tag{5.3}$$

From the IoU-scores, the mean IoU ($mIoU$) over all classes is obtained as a global metric.

In addition to the evaluation of the LC classification, the ability of the model to correctly predict LC changes over times is analysed in the first experiment set. This analysis is based on binary change maps of the prediction and the reference. These change maps have the same spatial dimensions as the input X and are generated by comparing the label/predicted map from the first and the lasttimesteps of the classified timeseriees and inserting no change (0) or change (1) into the corresponding pixel position of the change map. Similar to (Toker et al., 2022), afterwards, the IoU is computed for the *change* class, i.e. for all pixels assigned to change (1) by comparing the reference change map with the predicted change map, resulting in the binary change score BC:

$$BC = \frac{TP_{ch}}{TP_{ch} + FN_{ch} + FP_{ch}}, \tag{5.4}$$

with TP_{ch}, FN_{ch} and FP_{ch} as the true positives, false negatives and false positives for the *change* class, respectively.

All experiments are repeated three times with the goal to assess the impact of random components on the classification performance. In the conducted experiments, this includes the random parameter initialization and random selection of the training areas and acquisition dates (cf. section 4.3). The results of these test runs are averaged and only the average metrics and standard deviations are reported. To compare the achieved accuracy metrics and analyse whether improvements are significant, the respective accuracies are used for a one-sided two-sample T-Test with a significance level of 5%. Therefore, whenever a result is declared to be significantly better or worse than another, this is based on this hypothesis test.

6 Results and Discussion

In this chapter the results of the experiments introduced in section 5.2 are reported and discussed. The structure of the analysis is the one introduced in section 5.2, starting with the evaluation of the proposed method on the Lower Saxony, Kleve and Dynamic Earth datasets in section 6.1. In section 6.2 the evaluation of the hybrid feature extraction module is reported, which is followed by the results of the experiments comparing different variants of the patch generation (section 6.3) and temporal weighting modules (section 6.4). The chapter ends with the comparison of the proposed methods to other baseline models for SITS classification in section 6.5. For all experiments, first, the results on the Lower Saxony and Kleve datasets are described, followed by the results on the Dynamic Earth dataset. After analysing the results on each dataset separately, the achieved results are discussed on a general level by summarising the results and discussing them with respect to the corresponding research questions.

6.1 Evaluation of the proposed method for multi-temporal LC classification (Exp. 1)

In this section, the classification performance of the proposed method and its ability to detect class changes in time is analysed in detail in sections 6.1.1 and 6.1.2, respectively. The general classification performance is analysed based on all available test datasets (Lower Saxony, Kleve and Dynamic Earth), while the analysis of the capability of the method to detect changes in time is based on the Lower Saxony and Dynamic Earth datasets, because the Kleve dataset only provides mono-temporal class labels. In the end, the results on all datasets are summarized and research question 1 is answered in section 6.1.3.

6.1.1 Classification performance

Model $V\text{-}FE_{hyb}$ is proposed to predict land cover maps for multiple timesteps and improve the classification performance in comparison to other methods. In order to effectively compare the proposed method to other variants, first a detailed analysis of the achieved accuracies on the used datasets is necessary to identify under which conditions the method already performs well and to identify potential challenges. Therefore, this section analyses in detail the accuracy scores obtained for $V\text{-}FE_{hyb}$ on the Lower Saxony, Kleve, and Dynamic Earth datasets.

6.1.1.1 Results on the Lower Saxony dataset

Table 6.1 reports the overall accuracy (OA), mF^1-scores and the individual F^1-scores for all classes on the two test datasets $\mathcal{T}^L_{cor}$ and $\mathcal{T}^L_{te}$. In the following, first the results on both datasets are discussed individually before the differences in the achieved accuracies is analysed in more detail. Afterwards, the results on the Kleve datasets are analysed in section 6.1.1.2. Note that for all datasets in this section, the training of model $V\text{-}FE_{hyb}$ is conducted on the training dataset of Lower Saxony $\mathcal{T}^L_{tr}$ and only the test images differ.

Datas.	OA	mF^1	*Stl*	*Sld*	*Agr*	*Veg*	*Dfor*	*CFor*	*Wat*	*Bar*
$\mathcal{T}^L_{te}$	85.7 ±0.1	78.7 ±0.2	87.9 ±0.1	61.0 ±0.2	88.7 ±0.0	78.5 ±0.4	83.7 ±0.3	92.7 ±0.1	95.7 ±0.1	41.9 ±0.3
$\mathcal{T}^L_{cor}$	86.3 ±0.1	72.9 ±0.4	87.9 ±0.1	57.8 ±0.7	93.8 ±0.1	50.6 ±0.7	78.8 ±0.8	74.4 ±0.3	89.0 ±0.1	50.9 ±2.3

Table 6.1: OA, mF^1 and individual F^1-scores for all classes in % for the proposed method $V\text{-}FE_{hyb}$ on the test datasets $\mathcal{T}^L_{te}$ and $\mathcal{T}^L_{cor}$. All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations.

General classification results on the Lower Saxony dataset: When trained on the Lower Saxony training dataset $\mathcal{T}^L_{tr}$, the model is provided with a large variety of training samples as the images cover a large region (47 700km^2) and four different acquisition years. Eight LC classes are differentiated as reported in table 6.1, and some example predictions are shown in figure 6.1. Training is conducted based on more than 90% of the available tiles (cf. section 5.1.1), testing is based on the two test datasets $\mathcal{T}^L_{te}$ and $\mathcal{T}^L_{cor}$. While $\mathcal{T}^L_{te}$ contains 36 of the available tiles with labels provided from the ATKIS database which is also used to generate the labels used during training and contains some label noise, test dataset $\mathcal{T}^L_{cor}$ only comprises two tiles which have been corrected manually.

With this setting, the OA is about 86% for both datasets, and the achieved mF^1-scores are 78% and 73% for $\mathcal{T}^L_{te}$ and $\mathcal{T}^L_{cor}$, respectively. In all cases, the mF^1 is lower (by almost 13% for $\mathcal{T}^L_{cor}$ and 7% on $\mathcal{T}^L_{te}$) than the OA. The main reason for this behaviour is the imbalanced class distribution as reported in table 5.1, in combination with lower F^1-scores for classes having a smaller number of pixels compared to other classes. The individual class accuracies in table 6.1 and the examples shown in figure 6.1 show examples of this behaviour: The classes *Sealed area* and *Barren land* achieve comparatively low F^1-scores (61% and 42% on $\mathcal{T}^L_{te}$, respectively) and comprise relatively few pixels. According to the confusion matrix, the class *Barren land* is often misclassified as *Vegetation*, or, in some cases as *Agriculture*. In other cases, only parts of areas assigned to *Barren land* are classified correctly, shown exemplary in circle 1 in figure 6.1. For the class *Sealed area*, a challenge is the correct prediction of streets, which are rather fine lines in the satellite images with 10 m GSD. This challenge is depicted particularly in the second example in figure 6.1. Here, prominent roads, such as the highway traversing the whole image (near circle 1), are predicted to be thicker than the corresponding street in the reference. In contrast, rural roads, as exemplarily shown in the area indicated by the arrows in figure 6.1, are not predicted at all. Especially for the example in the second row the predicted streets are thicker than those in the corresponding reference. In other cases, streets are interrupted or they are mixed up with *Settlement* in cases where a street is near a settlement area (circle 2). Additionally, most of the pixels of *Sealed area*

are boundary pixels, meaning that a pixel belonging to this class can already include spectral information from the neighbouring LC classes. In these cases, the model has more difficulties in predicting the correct class, which can have a negative impact on the classification performance for *Sealed area*. The classes *Settlement*, *Agriculture* and *Coniferous forest* have comparatively many training samples, and the F^1-scores reach higher values of 88%, 89% and 93% on $\mathcal{T}_{te}^L$, respectively. In general, the outlines of forest areas, as shown exemplary in the second tile in figure 6.1, are detected very accurately, but sometimes the boundaries between *Deciduous forest* and *Coniferous forest* are not detected correctly. For the class *Water*, high accuracy scores are obtained (96% F^1-score on $\mathcal{T}_{te}^L$ and 89% F^1 on $\mathcal{T}_{cor}^L$), regardless of the small number of pixels assigned to this class. This is probably due to the different appearance of water and land surfaces that is easier to distinguish than other LC classes, which can also be seen in the Sentinel-2 image of the first tile in figure 6.1 in the areas of circles 4 and 6.

The mentioned examples show that the number of pixels that are available for a specific class is related to the performance of the class in most cases. However, if the appearance of a class is different from the other classes, also a relatively low number of samples is enough to distinguish this class from the others. During all the experiments, those test areas that contain larger areas, i.e. *Agriculture* or *Deciduous* and *Coniferous forest*, achieve the best performances. Low accuracy scores are obtained for areas in which it is hard to distinguish between *Vegetation* and *Agriculture*. In other cases, the label noise in the test dataset $\mathcal{T}_{te}^L$ can impact the accuracy scores, which is discussed in the following.

Analyis of the corrected test dataset $\mathcal{T}_{cor}^L$: Throughout all the experiments that were conducted for this thesis, a gap in the accuracy scores between $\mathcal{T}_{te}^L$ and $\mathcal{T}_{cor}^L$, e.g. 6% in the mF^1-score for variant $V\text{-}FE_{hyb}$, is observed, which raises the question for the cause of this gap. One possible reason is the obvious difference in the class distributions of both test datasets (cf. table 5.1). Another possible reason is the label noise included both in the training dataset $\mathcal{T}_{tr}^L$ and in the test dataset $\mathcal{T}_{te}^L$: the classifier might learn some wrong patterns that could also be available in $\mathcal{T}_{te}^L$, leading to over-optimistic quality metrics in an evaluation on that dataset. Before presenting the actual empirical evaluation in subsequent sections, in this section the accuracy metrics on $\mathcal{T}_{cor}^L$ and $\mathcal{T}_{te}^L$ are investigated in more detail in order to obtain a better idea about the possible reasons for the observed performance gap. This analysis is based on the proposed model $V\text{-}FE_{hyb}$. The obtained accuracy scores for $\mathcal{T}_{te}^L$ and $\mathcal{T}_{cor}^L$ are shown in table 6.1 and qualitative results for both corrected test tiles are shown in figure 6.1 for one timestep (14/02/2019) that was manually corrected. The F^1-scores for the individual classes differ between $\mathcal{T}_{te}^L$ and $\mathcal{T}_{cor}^L$, especially for the classes *Vegetation* (28% better on $\mathcal{T}_{te}^L$), *Coniferous Forest* (18% better on $\mathcal{T}_{te}^L$). For most classes, the accuracies on $\mathcal{T}_{cor}^L$ are worse than those on $\mathcal{T}_{te}^L$, but there are also exceptions, e.g. *Agriculture* (5% better on $\mathcal{T}_{cor}^L$) or *Barren land* (9% better on $\mathcal{T}_{cor}^L$). The qualitative examples in figure 6.1 allow to analyse these results in more detail.

There are several examples in which the class labels in $\mathcal{T}_{te}^L$ were wrong and had to be corrected for $\mathcal{T}_{cor}^L$. Most of these corrections involved the classes *Agriculture* and *Vegetation*, which are difficult to differentiate in some cases, even for a human operator. An example can be seen in the circles with number 3 (cf. figure 6.1). In $\mathcal{T}_{cor}^L$ these parts are assigned to the class *Agriculture* while these

areas are assigned to *Vegetation* in $\mathcal{T}_{te}^{L}$. In tile 1, the area covered by forest at the boundary of several lakes (circle 4 in figure 6.1) was labelled as *Vegetation* and *Barren land* in $\mathcal{T}_{te}^{L}$ and was corrected to *Forest* in $\mathcal{T}_{cor}^{L}$. Another example is shown in circle 5, in which an area of *Deciduous forest* ($\mathcal{T}_{te}^{L}$) was corrected to *Coniferous forest* for $\mathcal{T}_{cor}^{L}$. For all of these examples, the prediction is similar to the class labels from $\mathcal{T}_{te}^{L}$. Consequently, this erroneously improves the metrics for $\mathcal{T}_{te}^{L}$ while decreasing the metrics on $\mathcal{T}_{cor}^{L}$. If the errors in the database shown in these examples are representative, they could be explained by similar patterns occurring in the training data, where they would be considered to be label noise.

Of course, there are also examples in which the prediction agrees with label corrections in $\mathcal{T}_{cor}^{L}$. The lakes in tile 1 (circles 6 in figure 6.1) are not included in the database but predicted correctly. A part of the settlement area (circle 7 in figure 6.1) was relabelled as *Vegetation*, which is also the predicted label. As these areas are relatively small, they have only a small effect on the accuracy metrics. For some classes the performance on $\mathcal{T}_{te}^{L}$ and $\mathcal{T}_{cor}^{L}$ is similar, e.g. for *Settlement* (F^1-scores of 88% for both dataset), *Sealed area* (F^1-scores of 58% on $\mathcal{T}_{cor}^{L}$ and 61% on $\mathcal{T}_{te}^{L}$), *Agriculture* (F^1-scores of 94% for $\mathcal{T}_{cor}^{L}$ and 89% for $\mathcal{T}_{te}^{L}$) or *Decidous forest* (F^1-scores of 79% for $\mathcal{T}_{cor}^{L}$ and 84% for $\mathcal{T}_{te}^{L}$). As discussed before, the class *Sealed area* is correctly classified primarily for larger roads like highways or parking areas that only occur in tile 2 (cf. the large highway near circle 1 in tile 2 in figure 6.1). Rural roads contained in tile 1 are not predicted and, as they are too narrow, sometimes such roads are not included in rasterized labels as discussed in section 5.1.1. This results in lower performance of 58%-60% for this class compared to the mF^1-score, independently from the dataset, and shows the limitations of the used image data: due to the rather coarse spatial resolution of 10 m it is challenging and in some cases not possible to delineate fine structures such as streets that might have a width of less than 10 m. The other classes with similar F^1-scores on both datasets either have more samples (*Agriculture*) or are clearly distinguishable from the other classes (*Settlement*).

To summarise, the corrected test dataset $\mathcal{T}_{cor}^{L}$ is very small, so that relatively small areas of wrong classification results can have a large impact on the quality metrics. On the other hand, the analysis showed that the results on $\mathcal{T}_{te}^{L}$ are affected by the label noise in the reference, which leads to a positive bias in the performance metrics. Ultimately, the numbers achieved on $\mathcal{T}_{cor}^{L}$ are considered to be more reliable and, thus, are the main focus in the subsequent analyses. It may be noted that even though the values of the quality indices may differ between $\mathcal{T}_{te}^{L}$ and $\mathcal{T}_{cor}^{L}$, the tendency is very similar, in particular when ranking methods based on average indices.

6.1.1.2 Results on the Kleve dataset

In addition to the evaluation of the two test datasets $\mathcal{T}_{te}^{L}$ and $\mathcal{T}_{cor}^{L}$, the results of the models that are trained on the Lower Saxony dataset are evaluated on the Kleve dataset $\mathcal{T}_{kleve}^{L}$ described in section 5.1.1. The two other datasets both include some challenges, e.g. *label noise* that is included in $\mathcal{T}_{te}^{L}$ and a relatively small manually labelled reference in $\mathcal{T}_{cor}^{L}$; the additional evaluation on the Kleve dataset $\mathcal{T}_{kleve}^{L}$ can show if the results obtained on $\mathcal{T}_{te}^{L}$ and $\mathcal{T}_{te}^{L}$ can be confirmed. In addition, the Kleve dataset can indicate the model's generalisation performance. The achieved accuracy metrics for $\mathcal{T}_{kleve}^{L}$ are reported in table 6.2.

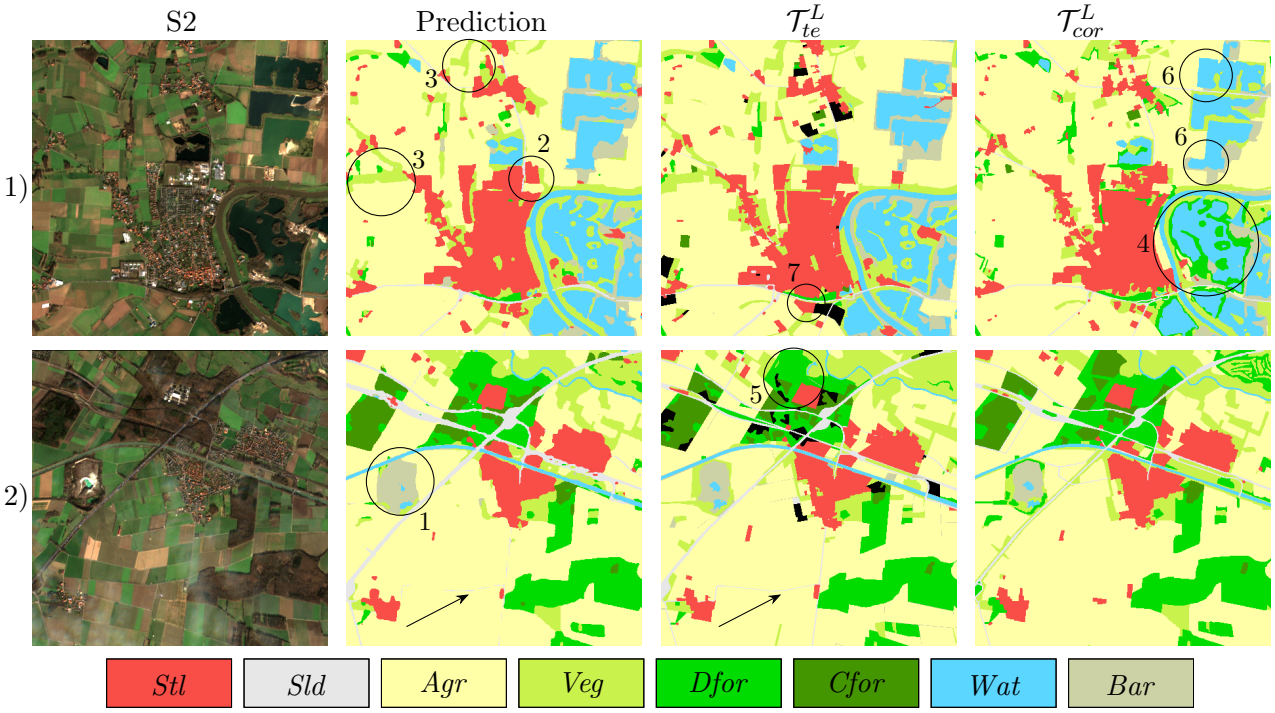

Figure 6.1: Examples of image patches from the corrected test dataset $\mathcal{T}_{cor}^{L}$ classified with the proposed model $V\text{-}FE_{hyb}$. The first column shows the S2 images from 2019-02-14, the second column the corresponding predictions of the proposed model $V\text{-}FE_{hyb}$, the third row the corresponding labels from the database ($\mathcal{T}_{te}^{L}$) and the last row the corrected class labels $\mathcal{T}_{cor}^{L}$. Black areas in $\mathcal{T}_{te}^{L}$ indicate areas that could not be assigned to one of the used LC classes, e.g. the areas assigned to *mixed forest* in the ATKIS database. The circles indicate examples that are mentioned in the analysis in section 6.1.1.1.

Datas.	OA	mF^1	*Stl*	*Sld*	*Agr*	*Veg*	*Dfor*	*CFor*	*Wat*	*Bar*
$\mathcal{T}_{kleve}^{L}$	72.3 ±1.0	64.8 ±0.8	59.2 ±0.6	18.4 ±0.5	86.1 ±1.7	69.6 ±2.3	73.7 ±1.8	81.2 ±0.8	88.7 ±0.3	41.3 ±0.6

Table 6.2: OA, mF^1 and individual F^1-scores for all classes in % for the proposed method $V\text{-}FE_{hyb}$ on the Kleve test dataset $\mathcal{T}_{kleve}^{L}$. All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations.

On the Kleve test dataset, a mF^1-score of 65% and an OA of 72% are obtained. These results are significantly lower than those on the other two test datasets for Lower Saxony. For instance, the mF^1-score on $\mathcal{T}_{kleve}^{L}$ is 7% lower than the one on $\mathcal{T}_{cor}^{L}$ and 14% lower than the one on $\mathcal{T}_{te}^{L}$. The included *label noise* in $\mathcal{T}_{te}^{L}$ is probably the main reason why the differences in the results on this dataset are higher. The largest drop in performance is obtained for the classes *Settlement* and *Sealed area*. The confusion matrix and the qualitative results shown in figure 6.2 help to analyse these results: Many false positive predictions for the class *Settlement* result in a relatively low precision score for this class (42%), while almost all points labelled as *Settlement* are correctly assigned to this class during classification (recall score of 97%). Many of the false positive predictions of the class *Settlement* belong to the class *Sealed area* in $\mathcal{T}_{kleve}^{L}$, resulting in a very low recall of 13% for *Sealed area*. This can be seen in the left example in figure 6.2, where all points that are labelled as *Sealed area* are classified as *Settlement* (circle 1). The wrong predictions of the pixels that are

labelled as *Sealed area* is probably caused by differences in the training data generation for $\mathcal{T}^L_{tr}$ and $\mathcal{T}^L_{kleve}$. For $\mathcal{T}^L_{tr}$, only larger streets such as highways or country roads with a larger width are used to create labels for this class as explained in section 5.1.1. The main reason to exclude streets with a smaller width is the GSD of 10 m of the Sentinel-2 images. This process excludes most roads inside settlement areas that are assigned to the class *Settlement* as one large urban area, as otherwise, many mixed pixels that include *Settlement* and *Sealed area* would occur. In contrast, the selected pixels that are manually labelled for $\mathcal{T}^L_{kleve}$ are labelled based on orthophotos with a resolution of 20 cm, and in this process also streets inside settlement areas can be distinguished from the surrounding. Therefore, the pixels that are surrounded by *Settlement* but labelled as *Sealed area* in $\mathcal{T}^L_{kleve}$ are mainly mixed pixels that contain both LC classes. These pixels are then predicted as *Settlement* as this was the assigned class for such pixels during the training process.

There is also quite some confusion between the classes *Agriculture* and *Vegetation*, which is a similar observation as discussed for the datasets $\mathcal{T}^L_{te}$ and $\mathcal{T}^L_{cor}$. The obtained F^1-score of 70% for the class *Vegetation* is located between its F^1-score on $\mathcal{T}^L_{te}$ and $\mathcal{T}^L_{cor}$, while the F^1-score of 86% for *Agriculture* is lower than its score on $\mathcal{T}^L_{te}$ (-2%) and $\mathcal{T}^L_{cor}$ (-8%). High F^1-scores are obtained for the classes *Water*, *Deciduous-* and *Coniferous forest*, with 89%, 74% and 81%, respectively. For these classes, the F^1-scores are relatively close to those on the corrected test dataset $\mathcal{T}^L_{cor}$ (differences between 0 and 4%), while the difference to $\mathcal{T}^L_{te}$ is higher (up to 11%).

Overall, the achieved results on the Kleve dataset show a drop in the overall performance (e.g. -8% compared to $\mathcal{T}^L_{cor}$), which was expected because the area is located outside of the training area, i.e. Lower Saxony. Furthermore, the model is trained to predict the labels from the training dataset $\mathcal{T}^L_{cor}$, which was generated differently from the labelling process of the Kleve dataset; this is another reason for the worse results. The results show some challenges, e.g. the classification of streets, which might be improved when a different procedure is followed to produce labels for the class *Sealed area* for the training dataset $\mathcal{T}^L_{tr}$. On the other hand, the achieved results also show that for several classes the performance is similar to the results achieved with the test dataset that are located in Lower Saxony ($\mathcal{T}^L_{cor}$ and $\mathcal{T}^L_{te}$), which indicates the ability of the model to generalise to new data in some respects.

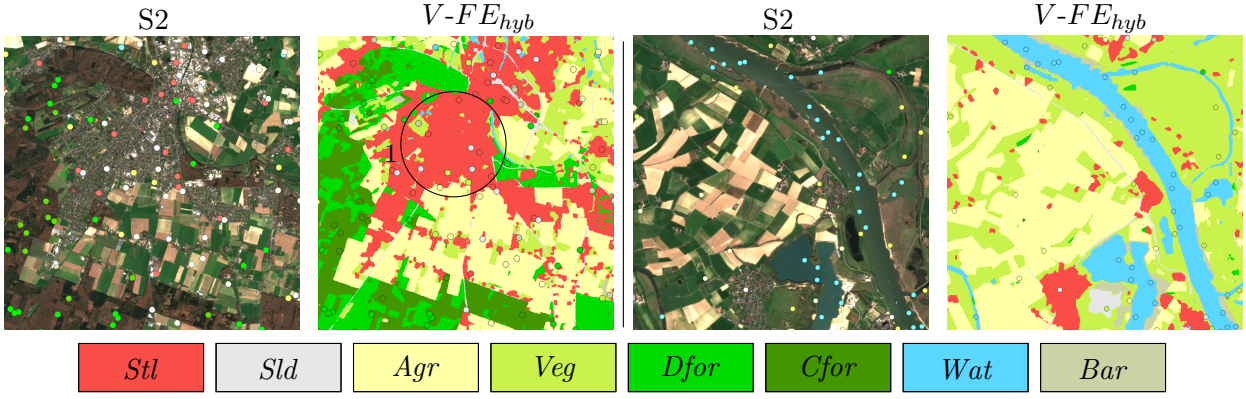

Figure 6.2: Two examples of classification results for the Kleve dataset $\mathcal{T}^L_{kleve}$. The first and third columns show S2 images, the second and fourth columns the corresponding predictions obtained using the proposed model $V\text{-}FE_{hyb}$. The points show the labels from the Kleve test dataset $\mathcal{T}^L_{kleve}$.

6.1.1.3 Results on the Dynamic Earth dataset

Table 6.3 reports the OA, mF^1-score as well as the individual F^1-scores for all classes predicted by the proposed model $V\text{-}FE_{hyb}$ on the test dataset $\mathcal{T}_{te}^D$ of the Dynamic Earth dataset. They are discussed in the following.

Variant	OA	mF^1	*Imp*	*Agr*	*Veg*	*Wet*	*Sol*	*Wat*
$V\text{-}FE_{hyb}$	66.4 ± 2.3	43.8 ± 0.5	30.4 ± 1.5	1.8 ± 0.7	75.2 ± 0.2	2.0 ± 2.0	62.7 ± 4.7	90.9 ± 2.5

Table 6.3: OA, mF^1 and individual F^1-scores for all classes in % for the proposed methods $V\text{-}FE_{hyb}$ one the Dynamic Earth test datasest $\mathcal{T}_{te}^D$. All quality metrics are averages over three experiments, the numbers behind the accuracy scores indicate the corresponding standard deviations.

On the Dynamic Earth dataset, the model $V\text{-}FE_{hyb}$ achieves an overall accuracy of 66.4% and a mF^1-score of 43.8%. These scores are much lower than those achieved on the Lower Saxony dataset, which is expected given that the training and test tiles are distributed across the entire Earth. This global distribution inherently leads to larger variability in the appearance of the same LC class. The differences between the OA and mF^1-score of almost 20% are much larger than the differences on the Lower Saxony dataset. The main reason for this large difference is the wide range of the F^1-scores for the individual classes as reported in table 6.3. The classes *Agriculture* and *Wetland* are barely classified correctly at all, with F^1-scores below 5% for all scenarios. These results are similar to those reported by Toker et al. (2022). However, Toker et al. (2022) only publish the IoU scores on the validation dataset for all classes; a comparison of the proposed model to their results is reported in section 6.5.

A closer analysis of the F^1-scores, the qualitative results in figure 6.3 and the confusion matrix show the cases in which the classifier has difficulties to predict the correct class. The classes *Forest & vegetation*, *Soil* and *Water* are classified best, with F^1-scores of 75%, 63% and 91%, respectively. For the class *Water*, some pixels at the borders of rivers or coastal areas are misclassified as shown in examples 1) and 2) in figure 6.3, but in general, coast lines or rivers are predicted at the correct positions. *Forest & vegetation* is the class with the highest number of pixels, as 37% of the pixels in $\mathcal{T}_{te}^D$ belong to this class. There are large regions in most of the test tiles that belong to that class and most of them are classified correctly, which results in recall scores of about 94%. The main reason for the F^1-score of only 75% is the comparably low precision of about 60%, caused by many false positive pixels that belong to *Agriculture* or *Soil* in $\mathcal{T}_{te}^D$, which can be seen in example 4) in figure 6.3. Pixels belonging to the class *Agriculture* in the reference are mostly misclassified as *Forest & vegetation* or *Soil*, as exemplarily shown in example 4) in figure 6.3, which explains the very low F^1-score for the class *Agriculture*. These misclassifications also impact the F^1-score for *Forest & vegetation*, but due to the higher percentage of pixels (37% in $\mathcal{T}_{te}^D$) for this class combined with a high number of true positives, the F^1-score for *Forest & vegetation* is still high (75%). Pixels belonging to the class *Wetland* are commonly misclassified as *Water* or *Soil* as shown in examples 1) and 2) in figure 6.3. This class is also the one with the smallest number of pixels in the dataset. Additionally, the similar appearance of wetlands compared to some water areas can explain that this class is only classified correctly for a handful of pixels. For the class *Impervious surfaces*, a F^1-score of 30% is achieved. There are many fine structures including single buildings

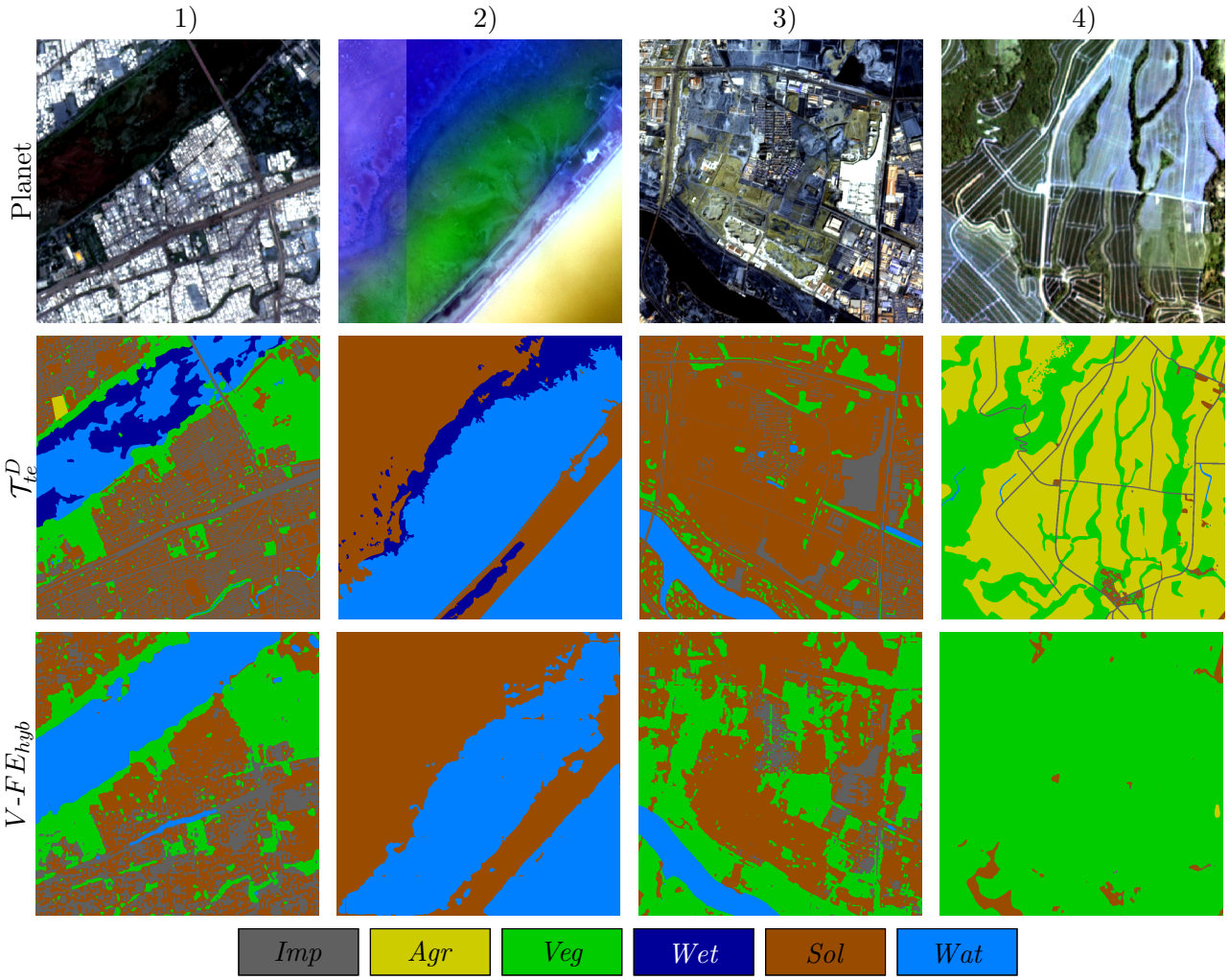

Figure 6.3: Examples of classification results from the Dynamic Earth test dataset $\mathcal{T}_{te}^{D}$ classified with the proposed model $V\text{-}FE_{hyb}$. The first row shows the S2 composite, the second row the labels from $\mathcal{T}_{te}^{D}$ and the following row the predictions of $V\text{-}FE_{hyb}$.

and streets, as shown in example 1) in figure 6.3, labelled as *Impervious surfaces*. In the predicted label maps, single buildings are not classified at the same level of detail as in the reference, and instead larger regions of *Impervious surfaces* are predicted. A main reason for this is probably the patch generation process with a patch size of $P = 4$.

To summarise, the Dynamic Earth dataset is much more challenging than the Lower Saxony dataset. This results in low accuracies for several classes that are influenced by several factors, e.g. the imbalanced class distribution and different appearances of pixels belonging to the same class at different places in the world. Classes such as *Water*, *Forest & vegetation* and *Soil* are differentiated best, probably due to the higher number of pixels for these classes during training, so that the training data represent the different appearances of these classes in a better way.

6.1.2 Analysis of predicted land cover changes

The proposed method $V\text{-}FE_{hyb}$ predicts land cover maps for each timestep that is used in the input timeseries. In the training process, each of the generated output maps is included in the computation of the loss function by combining corresponding training label maps with the predictions of each timestep. Therefore, if land cover changes occur between different timesteps in the label maps, the model can potentially learn to predict these changes by assigning the same pixel on the ground to different classes over time. However, the number of pixels that are changing the class of LC is low in comparison to the pixels that are assigned to the same LC class over all timesteps. For instance, for the Dynamic Earth dataset around 5% of the class labels change within the used timeperiod of two years, which is already a high percentage. In this section, the multi-temporal predictions of the proposed model $V\text{-}FE_{hyb}$ are analysed by focusing on the ability of the model to correctly classify changes in LC that happen within the used timeseries. This analysis is based on the score for the binary change (BC) used in (Toker et al., 2022) as a quantitative metric for change (cf. section 5.3). The binary change score can be compared with the results achieved by Toker et al. (2022) on the Dynamic Earth dataset. It has to be mentioned that this way of computing a metric for changes does not consider that the correct classes are predicted before and after a class change occurs. Furthermore, only the differences between the two used dates are integrated and the timesteps in between are not used. Therefore, in this section the analysis is extended to qualitative results of areas containing LC changes that help to understand in which cases changes are already classified correctly and in which cases challenges occur. In the following, first the results on the Lower Saxony dataset are discussed in section 6.1.2.1, which is followed by an analysis of the results on the Dynamic Earth dataset in section 6.1.2.2.

6.1.2.1 Results on the Lower Saxony dataset

The analysis of LC changes for the Lower Saxony dataset is based on the results achieved on the corrected test dataset $\mathcal{T}^L_{cor}$. The binary change (BC) score on $\mathcal{T}^L_{cor}$ is shown in table 6.4. In addition to the average score for the two tiles contained in $\mathcal{T}^L_{cor}$, the BC-score for two individual examples, that correspond to the images shown in figures 6.4 and 6.5 are reported in table 6.4. The BC-score for the complete dataset is computed based on the change maps generated by comparing the reference and the predictions from the first available date in the dataset (2019-02-14) and the last available date in the four-year timeseries (different for both tiles in $\mathcal{T}^L_{cor}$). For the computation of the binary change (BC) score for example 1 and 2, the LC maps from those dates (i.e. month) for which manually corrected maps ($\mathcal{T}^L_{cor}$) are available are used. These are highlighted by the black outlines in figure 6.4 and 6.5. For the two qualitative examples, the Sentinel-2 imagery, the corresponding label map from $\mathcal{T}^L_{cor}$ and the corresponding prediction are shown for six timesteps from the whole available timeseries of four years. Note that to classify the whole timeseries, four input timeseries of one year each need to be classified. The relatively few LC changes occurring in this dataset are the reason to show the complete time period for this dataset.

An overall BC-score of 8.9% is achieved on $\mathcal{T}^L_{cor}$, which is due to the large number of both, false positive and false negative predictions. For comparison: the BC-scores reported in (Toker et al., 2022) on the Dynamic Earth dataset achieve values between 10-11%. The two examples in

		Examples	
	$\mathcal{T}_{cor}^{L}$	1	2
BC	8.9 ± 0.5	8.8	8.8

Table 6.4: Binary change (BC) scores in % for the complete Lower Saxony corrected test dataset $\mathcal{T}_{cor}^{L}$ and for the two examples shown in figures 6.4 and 6.5. All results are obtained with model variant $V\text{-}FE_{hyb}$. The overall BC-score on $\mathcal{T}_{cor}^{L}$ is based on the change maps between 2019-02-14 and 2022-11-15, which are the first and last available satellite images in the dataset, and averaged over the three conducted experiments, with the number behind the score indicating the corresponding standard deviation. The BC-scores for the two examples are those obtained by comparing the change maps from the acquisition dates that were manually corrected and shown in figure 6.4 and 6.5.

figure 6.4 and 6.5 help to analyse this score. In the first qualitative example (figure 6.4) the main change is the extension of several lakes over time. In this area, gravel is extracted, which results in several new water areas over the observed years. For two acquisition dates, i.e. the 2019-06-14 and 2020-11-23, the satellite images were manually corrected (highlighted by the black outlines in figure 6.4). These two dates are used to generate the label change map ($\mathcal{T}_{cm}^{L}$) and the predicted change map that are shown in the last row. In the predicted change map, there are many false positive predictions for *Change*. Most of these false positives are fine details at class boundaries that can be categorised as noise of the classification. The new areas of *Water* are mostly classified correctly, e.g. the growth of the lake in circle 1. However, not all of the predicted changes look exactly like they were labelled in the reference, as often the boundaries are classified thicker or the surroundings of the lakes are classified different from the reference. Some larger areas that are false positives are also visible in the predicted change map. For instance, the area of *Vegetation* in circle 2 does not change the LC class in $\mathcal{T}_{cor}^{L}$, but the prediction changes from *Vegetation* to *Barren land*.

The second example is shown in figure 6.5 and shows an area with some urban development, highlighted in circle 1. For this tile, the manual labelling process was based on the satellite images from 2019-02-14 and 2021-03-05 (highlighted with the black outlines in figure 6.5). These dates are used to generate the reference for change and the predicted change map shown in the last row. Two larger areas of change are visible in the reference change map, highlighted by circles 1 and 2 in $\mathcal{T}_{cor}^{L}$. In the first area (circle 1), a new *Settlement* area was built, which is labelled correctly as a class change in 2021 and also predicted correctly. The second area shows a class change from *Vegetation* to *Agriculture*. This change is not classified in the corresponding predictions, in which this area is always classified as *Vegetation*. These classes are often mixed during classification as discussed earlier, and these results show that the detection of changes is, not surprisingly, even more challenging for classes that have a similar appearance. Similar to the first qualitative example, many false positives for *Change* are visible in the predicted change map.

Overall, the analysis of the two qualitative examples shows several reasons that may lead to the low BC-score. For both examples, a BC-score of 8.8% is obtained. The main reasons for this low score are many false positives for *Change* caused by wrong classification results at class boundaries, which occur between classes with similar appearance. However, the correct prediction

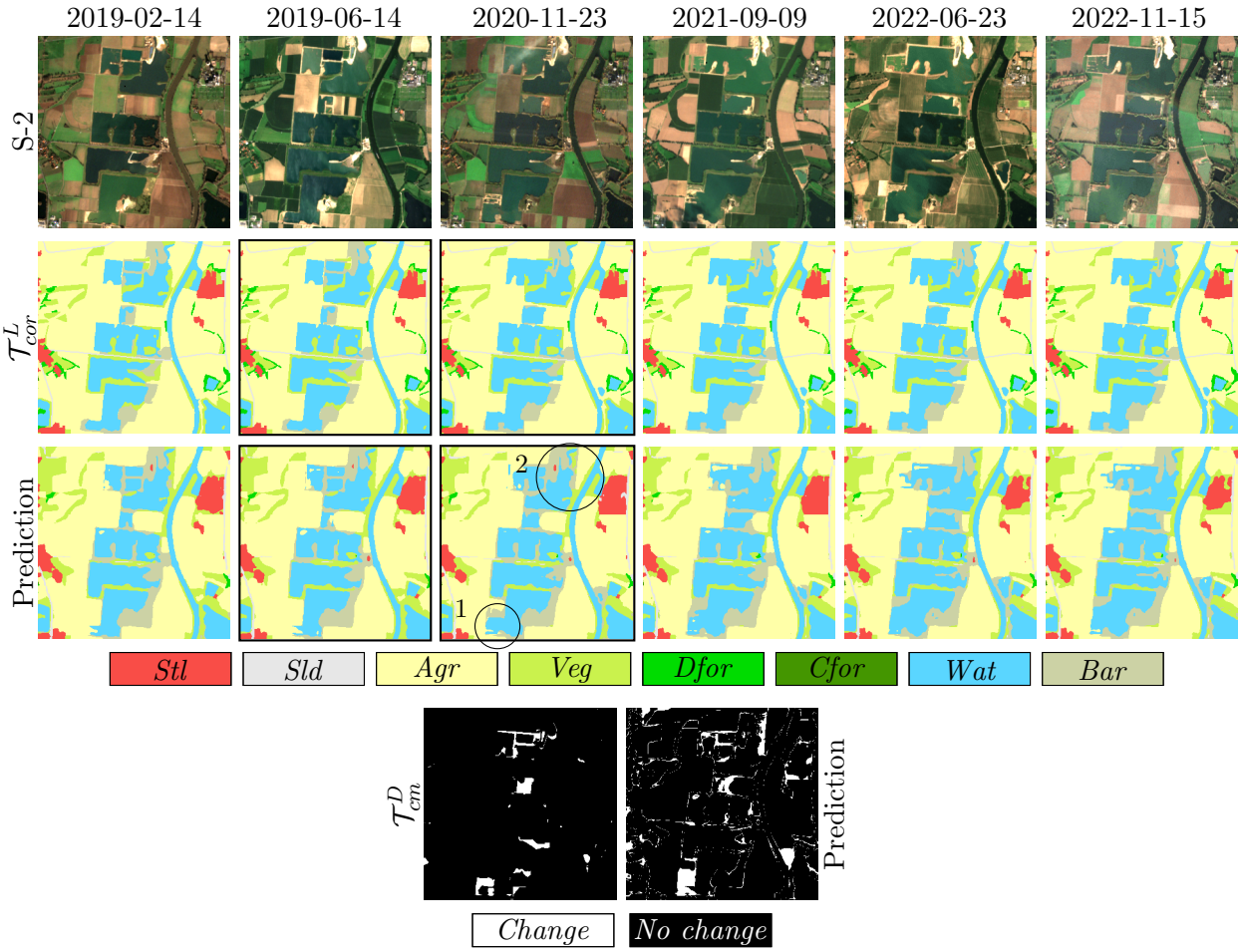

Figure 6.4: Qualitative examples of multi-temporal predictions of $V\text{-}FE_{hyb}$ on the Lower Saxony dataset. Row one shows the Sentinel-2 images, row two the corresponding manually generated reference, and row three the predictions of $V\text{-}FE_{hyb}$. In addition, the generated binary change maps between the 2019-06-14 and 2020-11-23 acquisition dates, which are those dates that were manually corrected, are shown in row four.

of larger areas affected by change is possible for classes like *Settlement* or *Water*, which are relatively easy to distinguish from their surroundings.

6.1.2.2 Results on the Dynamic Earth dataset

The binary change (BC) scores for the complete Dynamic Earth test dataset $\mathcal{T}^D_{te}$ as well as for three individual examples are shown in table 6.5. The BC-score for the complete dataset is based on the change maps generated by comparing the reference and the predictions from the first available date in the dataset (2018-01-01) to the last available date (2019-12-01). For the three examples, the BC-score is obtained by comparing the change maps from the first and last shown acquisition date in the corresponding example. In this way the obtained scores can be directly compared with the classification maps. The corresponding qualitative examples of three regions are shown in figures 6.6 - 6.8. For each example, the Planet image, the corresponding label map, and the prediction are shown for six of the twelve timesteps that are used within one input timeseries for the model. Every

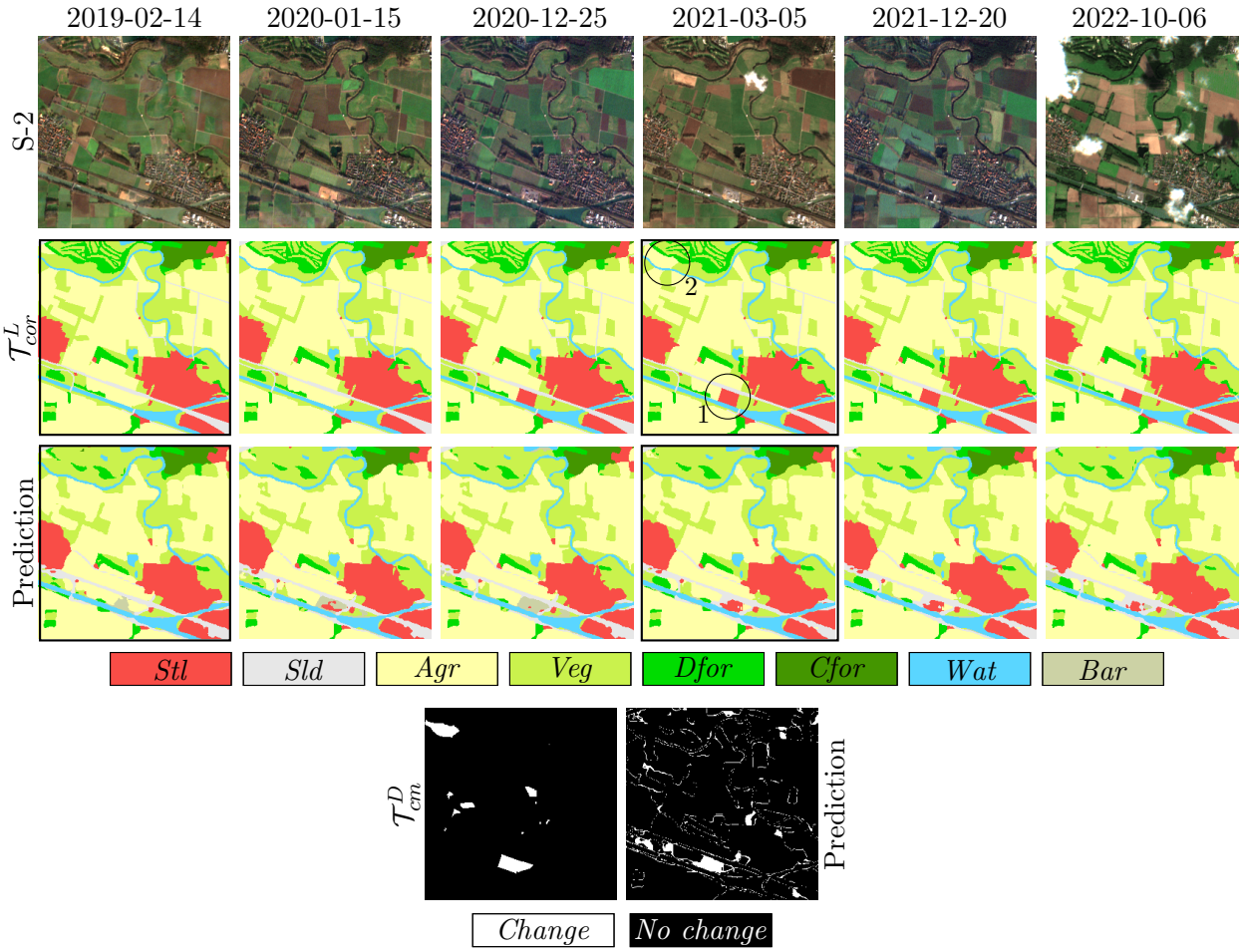

Figure 6.5: Qualitative examples of multi-temporal predictions of $V\text{-}FE_{hyb}$ on the Lower Saxony dataset. Row one shows the Sentinel-2 images, row two the corresponding manually generated reference, and row three the predictions of $V\text{-}FE_{hyb}$. In addition, the generated binary change maps between the first 2019-02-14 and 2021-03-05 acquisition date, which are those dates that were manually corrected, are shown in row four.

second timestep is shown, resulting in results shown for January, March, May, July, September, and November of the year.

		Examples		
	$\mathcal{T}^D_{te}$	1	2	3
BC	13.3 ± 1.7	47,7	14.6	12.9

Table 6.5: Binary change (BC) scores in % for the complete Dynamic Earth test dataset $\mathcal{T}^D_{te}$ and the three examples shown in figures 6.6 - 6.8. All results are obtained with model variant $V\text{-}FE_{hyb}$. The BC-score on $\mathcal{T}^D_{te}$ is based on the change maps between 2018-01-01 and 2019-12-01 and averaged over the three conducted experiments, with the number behind the score indicating the corresponding standard deviation. The BC-scores for the three examples are those obtained by comparing the change maps from the first and last shown acquisition date in the corresponding figures 6.6 - 6.8.

The achieved overall BC-score is 13.3%, which shows that some changes are already predicted correctly, but there is a large potential for improvement. The achieved score is a bit higher than

those reported by Toker et al. (2022), who achieve a BC-score of 10-11% for comparable scenarios. To be able to interpret this result, the three exemplary multi-temporal LC predictions and corresponding change maps shown in figures 6.6 - 6.8 are discussed in the following.

The example in figure 6.6 shows a comparably simple distribution of the three LC classees *Soil*, *Water* and *Wetland*. Over time, a large area of *Water* in the middle of the images disappears. In the predictions, the large *Water* area is classified correctly, only areas belonging to *Wetland* are wrongly assigned to *Soil* in the prediction. The transition from *Water* to *Soil* takes place between May and July. In the corresponding predictions, parts of the area are still classified as *Water* in May and July, resulting in a transition that takes longer in time than in the reference (circles 1 and 2). However, in July and September, the larger area of *Water* that is visible in January and March is almost completely classified as *Soil*. From July to November, a new elongated area of *Water* and *Wetland* appears in the satellite imagery (circle 3). This area is not recognised by the model at all, even though it is clearly visible in the satellite image. The permanent *Water* area at the bottom right corner of the shown images is, however, always predicted correctly. The change maps, computed between the first (2019-01-02) and the last (2019-11-01) shown acquisition date show a large area in which the LC class changed was detected correctly between these two dates. This is visible in the predicted change map in figure 6.6. However, the corresponding change map generated by comparing the labels from the first and last shown date shows an even larger region of change. Mainly, the pixels labelled as *Wetlands* in January and November are not predicted correctly, which can be explained by the very low F^1-score for this class. Overall, a BC-score of 47.7% is achieved in the shown example, which is much higher than the overall BC-score of 13% for the complete test dataset and is caused by the correct classification of class change from the *Water* area in January to *Soil* in November.

The example in figure 6.7 shows a river that is surrounded by a settlement area. The changes in LC mainly belong to the river in the centre of the images and some areas on top of the river, as shown in the change map of the reference $\mathcal{T}_{cm}^{D}$. However, not all changes in LC are available in the reference change map, as some changes happen faster on a timescale of some months rather than over the entire year. For example, several areas in the river are assigned to the classes *Wetlands* or *Forest & vegetation* during March, May and July (circles 1 and 2), but in the last reference map in November, these areas are again labelled as *Water*. Some of these areas are classified correctly as *Forest & vegetation*, e.g. in March (circle 3), whereas others are only partly predicted correctly (circle 4). The achieved change maps show only some similarities. For instance, the area of the river on the left side of the image is partly predicted correctly in addition to some smaller areas on the right top corner of the change maps. In the predicted change map there are many false positive predictions for change, which can be seen on both sides of the river. This is caused by a large number of misclassifications between the classes *Soil* and *Impervious surfaces*, as discussed in section 6.1.1. In combination, this results in the low BC-score of 14.6% for the shown example.

In the third example, shown in figure 6.8, larger areas of *Soil* are wrongly assigned to the class *Forest & vegetation* in the predictions (circle 1). In $\mathcal{T}_{te}^{D}$ this area of *Soil* is changing to *Forest & vegetation* in May (circle 2), and therefore these wrong classifications during January and March impact the generated change map from the predictions which shows *No change* for this area. In

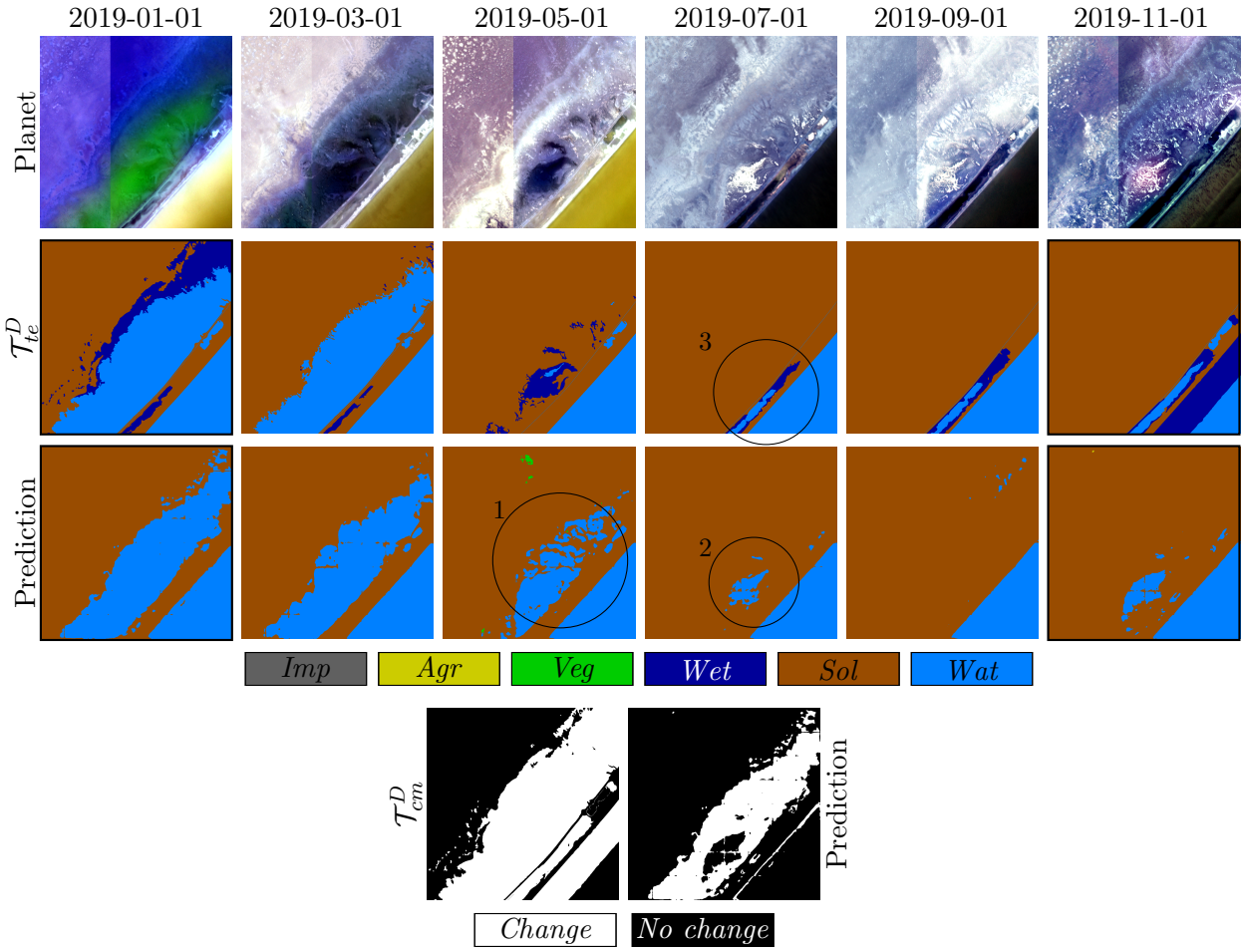

Figure 6.6: Qualitative examples of multi-temporal predictions of $V\text{-}FE_{hyb}$ on the Dynamic Earth dataset. Row one shows the Planet images for the first day of the month, row two the corresponding manually generated reference, and row three the prediction of $V\text{-}FE_{hyb}$. In addition, the generated binary change maps between the first (2019-01-01) and the last (2019-11-01) acquisition date are shown in row four.

the centre of the image (circle 3), several new constructions are labelled as *Impervious surfaces* in $\mathcal{T}_{te}^{D}$, but most of them are not classified correctly by the model. Overall, the third example shows a challenging area for the model, resulting in a lot of false LC classifications. These impact the predicted change map, as it is generated by a simple comparison of the predictions for two timesteps and results in a low BC-score of 12.9 % for the shown example.

In summary, the examples have shown several cases in which the prediction of class changes is already possible by comparing the predicted maps from two timesteps, e.g. for larger areas of *Water* (examples 1 and 2). On the other hand, there are many false positives for areas in which the models has difficulties to predict the correct LC class, e.g. the differentiation between *Impervious surfaces* and the surrounding class in settlement areas. Especially when new constructions are not predicted correctly, this has a larger impact on the classification of changes (example 3). In comparison to the BC-score reported by Toker et al. (2022) the achieved score of 13.3% is slightly higher, but this score leaves room for improvement, e.g. by integrating the classification of changes directly into the model.

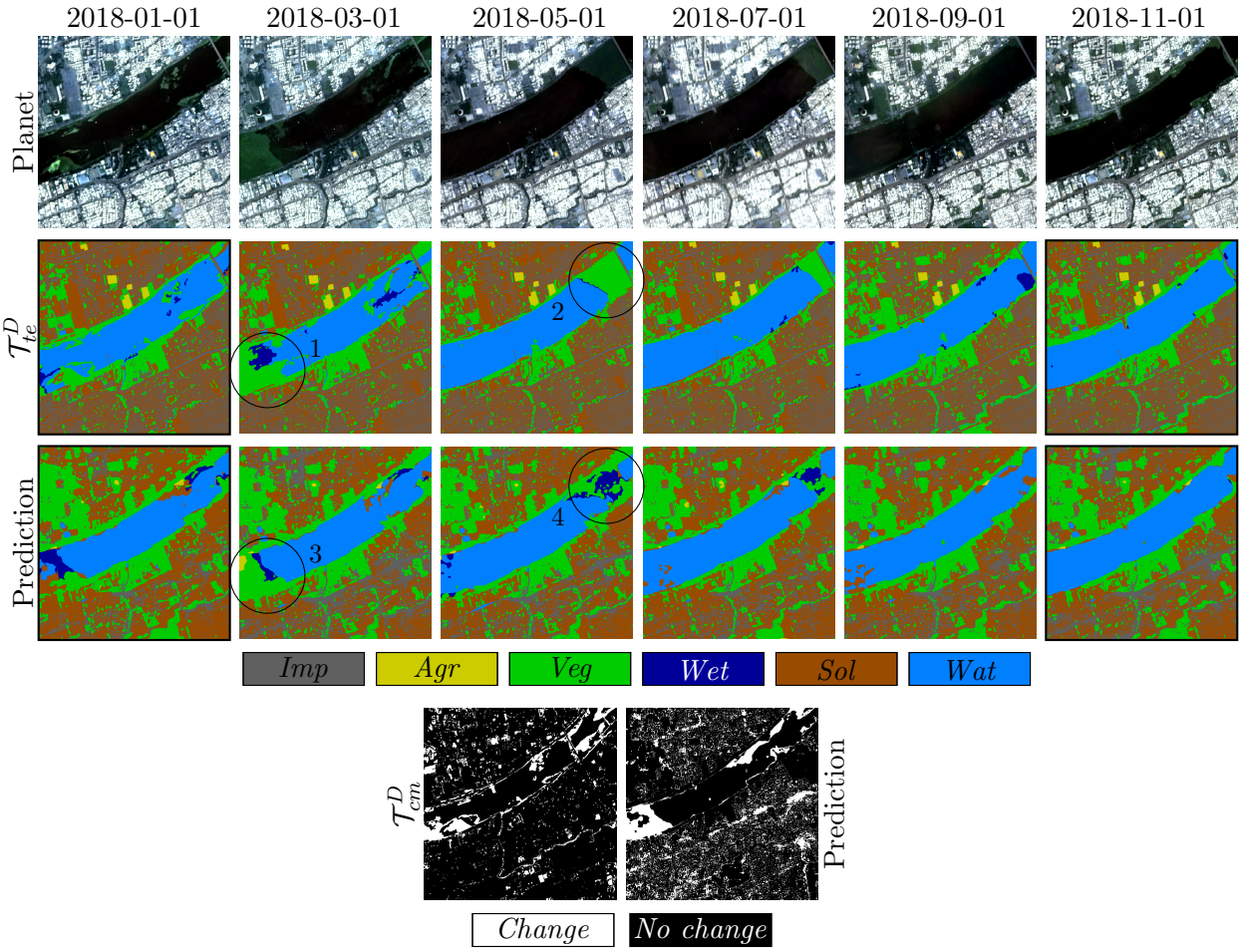

Figure 6.7: Qualitative examples of multi-temporal predictions of $V\text{-}FE_{hyb}$ on the Dynamic Earth dataset. Row one shows the Planet images for the first day of the month, row two the corresponding manually generated labels from the test dataset, and row three the prediction of $V\text{-}FE_{hyb}$. In addition, the generated binary change maps between the first (2019-01-01) and the last (2019-11-02) acquisition dates are shown in row four.

6.1.3 Discussion

Based on the detailed analysis of the proposed model $V\text{-}FE_{hyb}$, the first research question stated in chapter 1 can be answered:

1. How does the proposed model, which incorporates temporal context, perform for the task of multi-temporal land cover classification? Is the model able to detect class changes over time, even if it is not explicitly trained for change detection?

The model's performance varies on the different datasets, achieving mF^1-scores between 73% and 79% on the Lower Saxony dataset, 65% on the Kleve dataset, and 44% on the Dynamic Earth dataset. The main reasons for lower F^1-scores are fewer training samples for several classes and similarities in the appearance of LC classes. On the other hand, for classes to which a higher percentage of pixels belong or whose appearance differs significantly from those of other classes, higher F^1-scores are achieved. In addition, on the Lower Saxony dataset, having many more training samples and having test tiles that are located in areas in the vicinits of tiles that are used

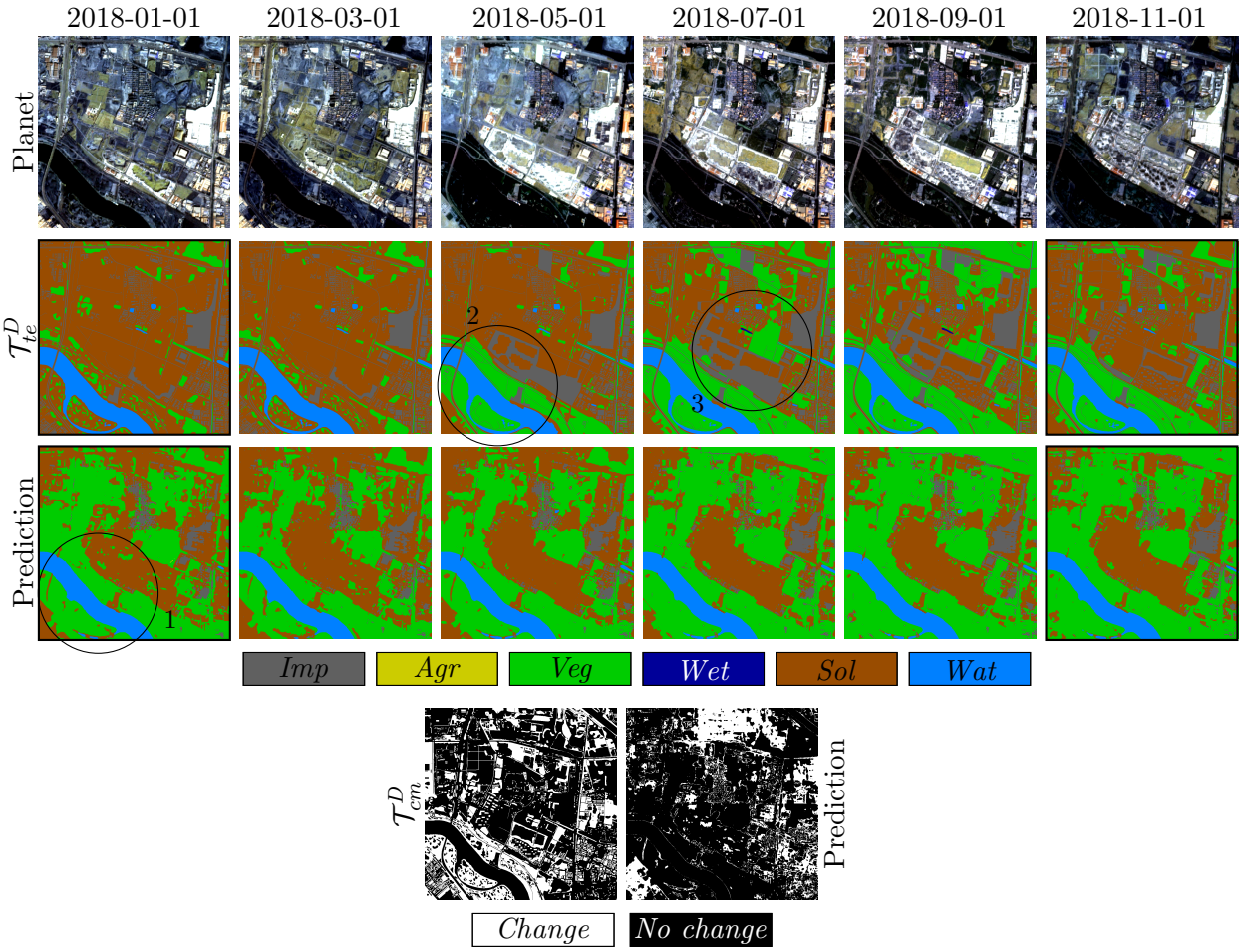

Figure 6.8: Qualitative examples of multi-temporal predictions of $V\text{-}FE_{hyb}$ on the Dynamic Earth dataset. Row one shows the Planet images for the first day of the month, row two the corresponding manually generated labels from the test dataset, and row three the prediction of $V\text{-}FE_{hyb}$. In addition, the generated binary change maps between the first (2019-01-01) and the last (2019-11-02) acquisition dates are shown in row four.

in training, the highest accuracies are achieved. In contrast, for the Dynamic Earth dataset, in which the training and test tiles are spread across the whole Earth, the lowest scores are achieved. In summary, and to answer the first part of research question 1, the proposed model performs well for the multi-temporal classification of most LC classes. Problems occur for classes with few training samples and similar appearance of different LC classes.

The detection of LC changes over time is still challenging for the proposed model. As the computed change maps are based on the comparison of two classified maps, the result for changes is directly dependent on the results of the LC classification. For classes that are difficult to predict, or regions where the classifier has problems, this results in areas that are wrongly predicted as change, or in areas that are not recognised as change. Additionally, wrong classification results at class boundaries result in many false positives for *Change* that are visible in all predicted change maps. In summary, these wrongly predicted pixels drastically decrease the BC-score, especially in combination with the small number of pixels that change the LC class over time. On the other hand, larger areas of class changes are predicted correctly for both datasets. Especially for classes that

have a different appearance from other classes, such as *Water* or *Settlement*, the predicted change map already includes these areas, which shows the potential to also detect class changes with the proposed model for multi-temporal LC classification. To conclude, and to answer the second part of research question 1, the new model is able to detect changes for classes that are easy to distinguish from the other classes and already achieve high F^1-scores for in the LC classification. However, in the proposed model, the prediction of changes is only implicitly included as the detection of changes is not integrated directly, e.g. in the loss function. It is expected that the integration of class changes into the model will have the side-effect of improving the general LC classification accuracy, as the model would then be forced to differentiate between real class changes and difficult samples that belong to the same class for several timesteps. Other adaptations can include morphological filters, e.g. based on an opening operation to remove misclassifications at class borders that are now counted as class changes. This topic is an interesting direction for future research.

6.2 Evaluation of the hybrid feature extraction module (Exp. 2)

In this section, the results of the variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{full}$ and $V\text{-}FE_{att}$ introduced in section 5.2.3 are compared. Variant $V\text{-}FE_{hyb}$ represents the proposed model, which was already analysed in section 6.1. It uses the introduced hybrid spatio-temporal feature extraction module, while variant $V\text{-}FE_{att}$ considers spatial and temporal dependencies separately by using self-attention and variant $V\text{-}FE_{full}$ uses the module that jointly models spatio-temporal dependencies with self-attention. First the results on the Lower Saxony and Kleve datasets are reported and discussed in section 6.2.1, which is followed by those from the Dynamic Earth dataset in section 6.2.2. Finally, the conclusions from the evaluation of all datasets are summarised to answer research question 2 stated in section 1.

6.2.1 Results on the Lower Saxony and Kleve datasets

Results on the Lower Saxony test datasets: Table 6.6 reports the OAs and mF^1-scores on the test datasets $\mathcal{T}^L_{te}$ and $\mathcal{T}^L_{cor}$ for the three model variants compared in this section. Additionally, table 6.7 shows the F^1-scores for all classes on the corrected test dataset $\mathcal{T}^L_{cor}$, while the F^1-scores on $\mathcal{T}^L_{te}$ are reported in appendix A. Note that the numbers for model $V\text{-}FE_{hyb}$ are identical to those reported in table 6.1.

The overall accuracy as well as the mF^1-score of the proposed model $V\text{-}FE_{hyb}$ are better than these achieved by the two other variants $V\text{-}FE_{att}$ and $V\text{-}FE_{full}$ with a mF^1-scores of 72.9% on $\mathcal{T}^L_{cor}$ and 78.7% on $\mathcal{T}^L_{te}$, as shown in table 6.6. The score on $\mathcal{T}^L_{cor}$ is 1.5% higher than the mF^1-score of variant $V\text{-}FE_{att}$ and 1.2% higher than the one of variant $V\text{-}FE_{full}$. On $\mathcal{T}^L_{te}$, model variant $V\text{-}FE_{hyb}$ improves the mF^1-score by 2% compared to variant $V\text{-}FE_{att}$ and by 1% compared to variant $V\text{-}FE_{full}$. The proposed model $V\text{-}FE_{hyb}$ performs significantly better based on the mF^1-scores on the test dataset $\mathcal{T}^L_{te}$; the improvements on the corrected test dataset are not significant, mainly due to the slightly higher standard deviations, but $V\text{-}FE_{hyb}$ still achieves the best results in combination with low standard deviations, which indicate stable results over multiple experiment runs. $V\text{-}FE_{full}$ achieves the second best results, whereas $V\text{-}FE_{att}$ achieves the lowest mF^1-scores

and also the highest standard deviations. These results indicate that the additional connections between patches of different dates and spatial locations, which are only used in variant $V\text{-}FE_{full}$, slightly improve the results in comparison to variant $V\text{-}FE_{att}$. Furthermore, the results show that convolutions used in variant $V\text{-}FE_{hyb}$, to embed spatial context are better suited than the window-based self-attention that is used in both other variants.

Table 6.7 provides the individual F^1-scores on $\mathcal{T}^L_{cor}$ for all classes. Variant $V\text{-}FE_{hyb}$ achieves the highest F^1-scores for seven of the eight LC classes. For the classes *Settlement*, *Agriculture*, *Deciduous forest* and *Water* the F^1-scores are very close for all variants. These are also the classes that achieve the highest F^1-scores in general, which indicates that for these classes the samples in the training dataset are representative for their general appearance and all model variants are able to generalise well on the test datasets. The highest standard deviations of variant $V\text{-}FE_{att}$ are obtained for the classes *Vegetation* and *Barren land*. This indicates a higher instability of model variant $V\text{-}FE_{att}$ compared to the other variants. *Barren land* is also the class with the highest difference in the F^1-score, as $V\text{-}FE_{hyb}$ achieves an F^1-score for this class that is 5% higher than the one for thee two other model variants. The results on $\mathcal{T}^L_{te}$ show a similar trend.

Variant	$\mathcal{T}^L_{cor}$		$\mathcal{T}^L_{te}$	
	OA	mF^1	OA	mF^1
$V\text{-}FE_{hyb}$	**86.3** ± 0.1	**72.9** ± 0.4	**85.7** ± 0.1	**78.7** ± 0.2
$V\text{-}FE_{att}$	85.3 ± 0.6	71.4 ± 1.1	84.6 ± 0.5	76.8 ± 0.1
$V\text{-}FE_{full}$	85.8 ± 0.3	71.7 ± 0.3	85.2 ± 0.2	77.8 ± 0.0

Table 6.6: OA and mF^1 in % for the experiments regarding the hybrid spatio-temporal feature extraction module FE_{hyb} on the Lower Saxony datasets $\mathcal{T}^L_{te}$ and $\mathcal{T}^L_{cor}$. All quality metrics are averages over three experiments, the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

Variant	*Stl*	*Sld*	*Agr*	*Veg*	*Dfor*	*CFor*	*Wat*	*Bar*
$V\text{-}FE_{hyb}$	**87.9** ± 0.1	**57.8** ± 0.7	**93.8** ± 0.1	**50.6** ± 0.7	**78.8** ± 0.8	**74.4** ± 0.3	89.0 ± 0.1	**50.9** ± 2.3
$V\text{-}FE_{att}$	86.7 ± 0.9	56.1 ± 0.7	93.2 ± 0.3	48.8 ± 2.4	78.5 ± 0.5	72.6 ± 1.2	**89.3** ± 0.4	45.8 ± 3.8
$V\text{-}FE_{full}$	87.8 ± 0.2	**57.8** ± 0.1	93.6 ± 0.2	47.7 ± 0.6	**78.8** ± 0.2	73.4 ± 0.0	88.4 ± 0.6	45.6 ± 1.8

Table 6.7: Individual F^1-scores for all classes in % achieved by the compared model variants on $\mathcal{T}^L_{cor}$. All quality metrics are averages over three experiments, the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

Figure 6.9 shows three examples from the test dataset $\mathcal{T}^L_{te}$ and the corresponding classification results for all three model variants, indicated by the numbers 1) - 3). As the F^1-scores for the classes differ by a maximum of 6% (*Barren land*) between the model variants, the general impression is similar. In the first example in figure 6.9, the predicted areas of the class *Vegetation* are larger than in the reference $\mathcal{T}^L_{te}$ in which these pixels are assigned to *Agriculture* by all three variants. The differentiation between these two classes is challenging as already discussed in section 6.1.1.1, and all three variants face this difficulty. Variant $V\text{-}FE_{att}$ produces some artefacts in the sea near the coast in the first example, which is not the case for both other variants and is one example that underlines the decrease in the classification accuracy for this variant. Example two shows a case in

which the classification of *Barren land* was not successful (circle 1). The larger region of *Barren land* is only partially detected by the proposed model $V\text{-}FE_{hyb}$, while both other variants do not predict *Barren land* at all in that region. In example three, there is again a large region labelled as *Barren land* (circle 2) that is only marginally visible in the predictied maps of variants $V\text{-}FE_{att}$ and $V\text{-}FE_{full}$. In contrast, variant $V\text{-}FE_{hyb}$ predicts a much larger region as *Barren land* than the corresponding label image. Because the class *Barren land* has comparably few samples in all datasets, the wrong classification of a larger region of this class can already result in a difference in its F^1-score and can explain the difference in the F^1-score of 6% between the proposed model $V\text{-}FE_{hyb}$ and both other variants. Another example in which variant $V\text{-}FE_{hyb}$ outperforms both other variants it the row of buildings in example two in figure 6.9 (circle 3). Even if not all buildings are differentiated as detailed as in $\mathcal{T}^L_{te}$, the prediction of variant $V\text{-}FE_{hyb}$ is clearly better than the prediction of both other variants. In the third example of figure 6.9, a case in which the LC information of the database results in a wrong reference is shown (circle 4). The forest area in the centre of the image is assigned to the class *Vegetation* in the test dataset $\mathcal{T}^L_{te}$. All three models predict *Deciduous forest* in that area, which is correct when visually comparing it to the satellite image. To sum up, the visual results explain the obtained accuracy scores for the three variants to some extent and underline the superior performance of the proposed method. For most classes, such as *Agriculture*, *Settlement*, *Decidous forest* or *Water*, the differences in the F^1-scores and in the qualitative results are very small. Higher differences are obtained for the class *Barren land*, for which the qualitative examples show challenging cases in which the classification fails.

Results on the Kleve dataset: The accuracy scores achieved on the Kleve dataset by the three compared model variants are reported in tables 6.8 and 6.9. Note that the numbers for model $V\text{-}FE_{hyb}$ are identical to those reported in table 6.2.

Variant	$\mathcal{T}^L_{kleve}$	
	OA	mF^1
$V\text{-}FE_{hyb}$	72.3 ±1.0	**64.8** ±0.8
$V\text{-}FE_{att}$	71.9 ±0.3	64.7 ±0.7
$V\text{-}FE_{full}$	**72.6** ±0.4	**64.8** ±0.1

Table 6.8: A and mF^1 in % for the experiments regarding the hybrid spatio-temporal feature extraction module FE_{hyb} on the Kleve test dataset $\mathcal{T}^L_{kleve}$. All quality metrics are averages over three experiments, the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

Variant	*Stl*	*Sld*	*Agr*	*Veg*	*Dfor*	*CFor*	*Wat*	*Bar*
$V\text{-}FE_{hyb}$	59.2 ±0.6	18.4 ±0.5	86.1 ±1.7	**69.6** ±2.3	73.7 ±1.8	81.2 ±0.8	**88.7** ±0.3	**41.3** ±0.6
$V\text{-}FE_{att}$	57.4 ±1.1	**19.1** ±2.4	**86.5** ±1.4	67.2 ±0.7	75.1 ±0.6	82.7 ± 1.2	88.5 ±0.4	41.2 ±1.2
$V\text{-}FE_{full}$	**59.7** ±0.2	17.4 ±2.1	86.3 ±0.8	68.7 ±2.0	**76.4** ±0.1	**82.8** ± 0.3	88.6 ±0.1	38.5 ±1.8

Table 6.9: Individual F^1-scores for all classes in % achieved by the compared model variant on $\mathcal{T}^L_{kleve}$. All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

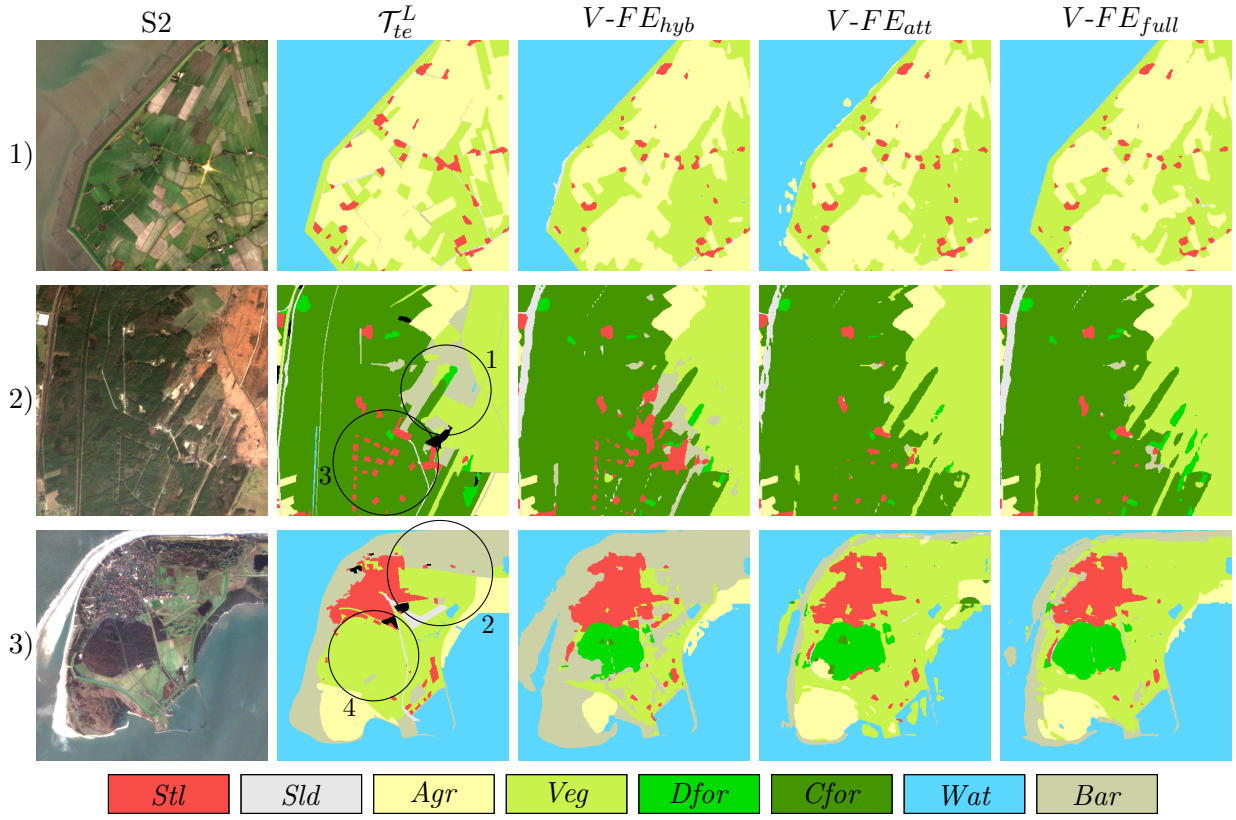

Figure 6.9: Examples for classification results of variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{att}$ and $V\text{-}FE_{full}$ trained on the Lower Saxony dataset in comparison to the reference from the test dataset $\mathcal{T}_{te}^{L}$. The first column shows the S-2 composite, the second column the labels from $\mathcal{T}_{te}^{L}$, and the following columns the predictions of the three model variants.

The mF^1-scores and OAs achieved on the Kleve test dataset $\mathcal{T}_{kleve}^{L}$ for the three model variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{att}$ and $V\text{-}FE_{full}$ are all very similar. There is no difference in the mF^1-score between variants $V\text{-}FE_{hyb}$ and $V\text{-}FE_{full}$, while $V\text{-}FE_{att}$ achieves a mF^1-score that is 0.1% lower. Slightly higher variations are obtained for the OA, nevertheless, these differences are not statistically significant either, as the maximum difference of 0.7% is in the range of the standard deviations. Regarding the individual F^1-scores of the classes, the proposed model $V\text{-}FE_{hyb}$ and variant $V\text{-}FE_{full}$ both achieve the highest scores for three of the eight LC classes and variant $V\text{-}FE_{att}$ for the remaining two. For several classes the F^1-scores of all variants are very close, e.g. for *Agriculture* and *Water*. For both forest classes, variant $V\text{-}FE_{hyb}$ achieves lower F^1-scores than both other variants, as the F^1-score for *Deciduous forest* is 2.7% lower and the F^1-score for *Coniferous forest* is 1.6% lower than the one for variant $V\text{-}FE_{full}$. For other classes, such as *Vegetation* (+0.9%) and *Barren land* (+2.8%) it is the opposite. It has to be mentioned that the Kleve dataset is even smaller than the corrected test dataset from the area of Lower Saxony. For the class *Agriculture*, which is the class with the highest number of points, 329 pixels are assigned to this class, while for the class *Barren land*, which is the class with the smallest number of samples, the evaluation is based on 81 pixels only and therefore, changes in the classification of a few pixels already have a relatively large impact on the F^1-scores.

To summarise, the comparison of the model variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{att}$ and $V\text{-}FE_{full}$ on the Kleve dataset did not result in significant differences. Overall, variant $V\text{-}FE_{att}$ achieved a slightly worse OA and mF^1-score than both other variants, which confirms the tendency of the experiments on the other test dataset for Lower Saxony.

6.2.2 Results on the Dynamic Earth dataset

Table 6.10 reports the OA and the F^1-scores on the test dataset $\mathcal{T}_{te}^{D}$ of the Dynamic Earth dataset for the three compared variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{att}$ and $V\text{-}FE_{full}$. In addition, figure 6.10 provides some qualitative results of classification results of all three model variants in comparison to the labels from the test dataset $\mathcal{T}_{te}^{D}$. Note that the numbers for model $V\text{-}FE_{hyb}$ are indentical to those reported in table 6.3.

Variant	OA	mF^1	*Imp*	*Agr*	*Veg*	*Wet*	*Sol*	*Wat*
$V\text{-}FE_{hyb}$	**66.4** ± 2.3	**43.8** ± 0.5	**30.4** ± 1.5	1.8 ± 0.7	**75.2** ± 0.2	**2.0** ± 2.0	62.7 ± 4.7	**90.9** ± 2.5
$V\text{-}FE_{att}$	65.6 ± 0.9	39.6 ± 1.5	7.7 ± 5.7	**4.2** ± 1.5	73.2 ± 0.5	0.0 ± 0.0	63.3 ± 2.2	89.1 ± 0.9
$V\text{-}FE_{full}$	66.3 ± 1.7	41.4 ± 1.9	22.0 ± 7.8	0.8 ± 0.5	74.2 ± 1.1	0.0 ± 0.0	**63.5** ± 2.5	88.0 ± 3.8

Table 6.10: OA, mF^1 and individual F^1-scores for all classes in % achieved for variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{att}$ and $V\text{-}FE_{full}$ on $\mathcal{T}_{te}^{D}$. All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

In the OA as well as the mF^1-score, the proposed model $V\text{-}FE_{hyb}$ achieves the highest scores on the test dataset $\mathcal{T}_{te}^{D}$ compared to the variants $V\text{-}FE_{att}$ and $V\text{-}FE_{full}$, as shown in table 6.10. While the overall accuracies are in a similar range for all three variants with only 1% variation, the mF^1-scores vary by 4%. Similar to the results on the Lower Saxony dataset, model variant $V\text{-}FE_{att}$ performs worst with -4.2% mF^1-score compared to variant $V\text{-}FE_{hyb}$, which is a statistically significant difference. Variant $V\text{-}FE_{full}$ performs second best with -2.4% mF^1-score compared to variant $V\text{-}FE_{hyb}$. This difference is not statistically significant, due to the higher standard deviation of 1.9% for variant $V\text{-}FE_{full}$.

Regarding the individual F^1-scores, the proposed model achieves the highest scores for four out of six LC classes. The classes *Forest & vegetation*, *Soil* and *Water* are the classes for which the best performance is achieved, as already discussed in section 6.1.1. For the classes *Forest & vegetation* and *Soil*, there are only small differences in the F^1-scores between all three model variants, and, similar as in the mF^1-score, the differences between variants $V\text{-}FE_{hyb}$ and $V\text{-}FE_{full}$ are very small (maximum 1%), while the differences in the F^1-scores between variants $V\text{-}FE_{hyb}$ and $V\text{-}FE_{att}$ are a little bit higher (maximum 2%).

Large differences in the F^1-scores are obtained for the class *Impervious Surface*, with a F^1-score of 30% for variant $V\text{-}FE_{hyb}$, 8% for variant $V\text{-}FE_{att}$ and 22% for variant $V\text{-}FE_{full}$. Additionally, the standard deviations for variants $V\text{-}FE_{att}$ and $V\text{-}FE_{full}$ are much larger (6% and 8%, respectively). The visual results in the second row of figure 6.10 show an example that explains the lower F^1-score for *Impervious Surface*, especially for variant $V\text{-}FE_{att}$. The urban area below the river

includes many streets and buildings that are labelled individually, i.e. as single buildings or fine streets, in the test dataset $\mathcal{T}_{te}^{D}$. Model variant $V\text{-}FE_{hyb}$ predicts several of these structures that are surrounded by the class *Soil*, but already for this variant many buildings are missing in the result and the wider road is partly misclassified. Both other variants predict drastically fewer building and road structures in this area. Instead, the urban area is mainly classified as *Soil*. The F^1-score of class *Agriculture* is very low for all variants. As discussed earlier, most of the pixels belonging to *Agriculture* are wrongly assigned to *Forest & vegetation* or *Soil* in the predictions. The higher score for *Agriculture* for variant $V\text{-}FE_{att}$ is unexpected as this variant achieves the worst results in the OA and mF^1-score; it can be explained by the exemplary LC predictions in figure 6.10. The agricultural area in the first row is only predicted correctly for variant $V\text{-}FE_{att}$, while both other variants predict *Forest & vegetation* or *Soil* in this area. However, it remains an open question why this model is able to predict the class *Agriculture* at least for some areas while both other variants achieve even lower F^1-scores. The second example in figure 6.10 visualises what happens for pixels belonging to the class *Wetland* (circle 1): for all three variants, all pixels assigned to *Wetland* in the test dataset are predicted as *Water*. In the corresponding satellite image of the shown timestep, it is also relatively difficult to distinguish clearly between these two classes, but it is not impossible as the areas assigned to *Wetland* are brighter than the surrounding areas of *Water*. Probably the low number of pixels assigned to this class in the training dataset does not represent this class well.

The qualitative results in figure 6.10 show some more differences between the model variants. For example, the long arm of class *Soil* (circle 2) is only predicted by the proposed variant $V\text{-}FE_{hyb}$, while $V\text{-}FE_{att}$ does not detect it at all and variant $V\text{-}FE_{full}$ only predicts some parts of it. Example 3) also shows that the small rectangular areas of the class *Impervious Surface* cannot be predicted by any of the models (circle 3), similar to the buildings in example 2). The detection of such fine structures is a main challenge for all compared variants in this section and is probably caused by the patch generation process in the first layers of the models.

6.2.3 Discussion

The proposed module FE_{hyb} for hybrid spatial-temporal feature extraction was evaluated based on three datasets for SITS classification with the task of LC classification. Besides the similar application, there are major differences between these datasets as discussed in section 5.1.3. In combination, the three datasets cover a variety of LC classes with one dataset having comparably few training samples that are labeled manually in comparison to a large dataset with some label noise.

On the Lower Saxony and Dynamic Earth test datasets the proposed model $V\text{-}FE_{hyb}$ achieves the highest mF^1-score and OA compared to variants $V\text{-}FE_{att}$ and $V\text{-}FE_{full}$. On the Kleve dataset the mF^1-score and OA are very close for all variants and while the proposed variant achieved the highest mF^1-score, the achieved OA is the second best. On the Lower Saxony dataset, the performance of variant $V\text{-}FE_{hyb}$ is significantly better on the test dataset $\mathcal{T}_{te}^{L}$ compared to both other variants, $V\text{-}FE_{att}$ and $V\text{-}FE_{full}$, while the performance difference is slightly smaller on $\mathcal{T}_{cor}^{L}$. For the Dynamic Earth dataset, only the performance difference to variant $V\text{-}FE_{att}$ is significant, the latter being the model achieving the lowest F^1-scores throughout all experiments. In the

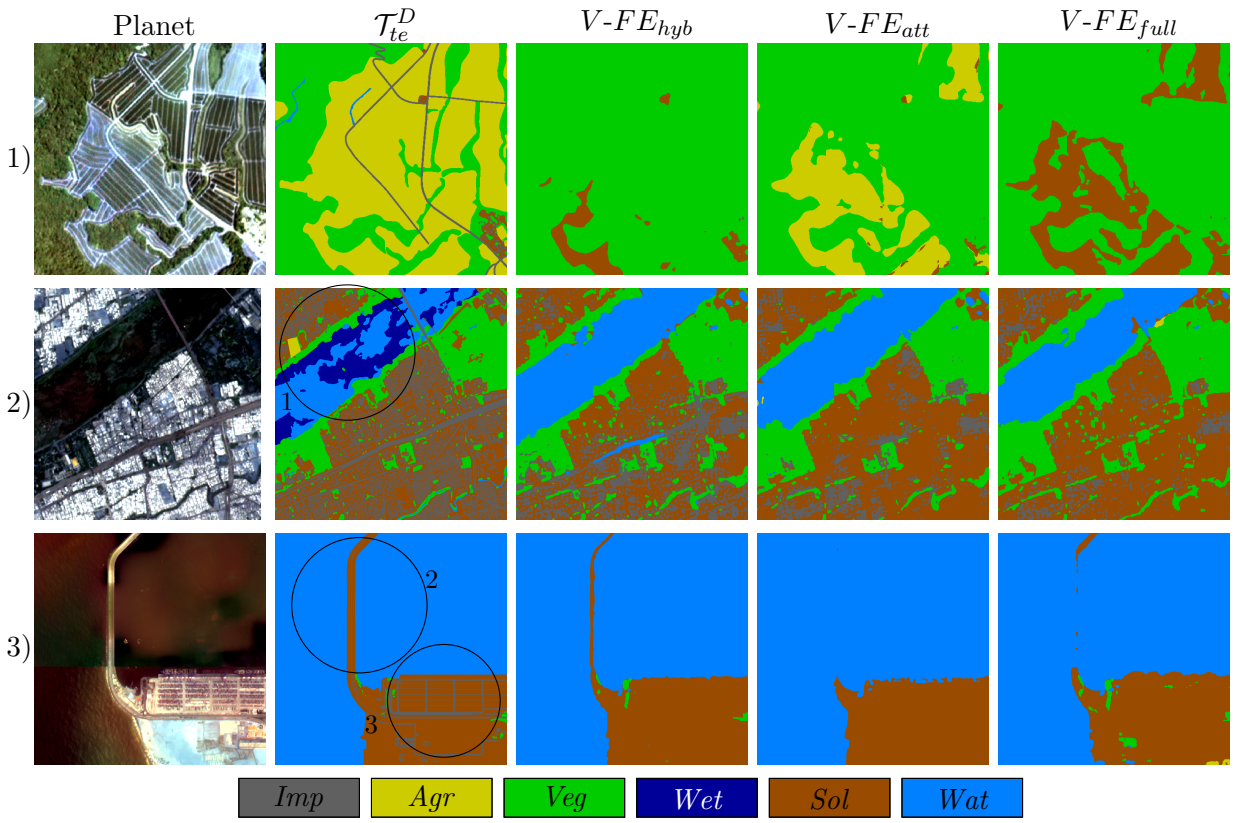

Figure 6.10: Examples of classification results of variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{att}$ and $V\text{-}FE_{full}$ in comparison to the labels from the test dataset $\mathcal{T}_{te}^{D}$ of the Dynamic Earth dataset. The first column shows the Planet composites, the second column the labels from $\mathcal{T}_{te}^{D}$ and the following columns the classifications of the three variants. The Planet images are acquired at 01/06/2018 and the classification results are those from the corresponding month in the input timeseries.

following, the obtained results are discussed with respect to the research question stated in section 5.2.3:

2. Does the proposed hybrid feature extraction module, considering spatial context with convolutions and temporal context with self-attention, lead to better results than modules solely relying on self-attention for feature extraction?

The comparison of the results of the proposed model $V\text{-}FE_{hyb}$ to those of variant $V\text{-}FE_{att}$ shows that convolutions are better suited for capturing spatial context than the window-based self-attention that is used in variant $V\text{-}FE_{att}$. In this regard, and to answer research question 2, the proposed hybrid features extraction module outperforms a module relying solely on self-attention in separated spatial and temporal streams. On the other hand, a comparison of the results achieved by the proposed model $V\text{-}FE_{hyb}$ and those of variant $V\text{-}FE_{full}$, which jointly consisers spatio-temporal dependencies with self-attention, shows a significant improvement for $V\text{-}FE_{hyb}$ only for one of the test datasets. However, the obtained performance for the proposed model $V\text{-}FE_{hyb}$ is still the same or better throughout all experiments, while the computational complexity is reduced. Therefore, to answer research question 2 conclusively, the usage of the proposed hybrid feature extraction module leads to similar or better results compared to modules that solely rely on self-

attention. While the proposed module significantly improves the results compared to a model based on self-attention with separate consideration of spatial and temporal context, the proposed module results in similar performance compared to a model considering joint spatio-temporal attention.

To conclude, the separation of the computation of spatial and temporal dependencies is possible without a drop in performance, under the condition that suitable methods are used for each dimension. For the used application of LC classification from SITS, convolutions are more suitable than window-based self-attention in the spatial dimension. Regarding the extraction of temporal dependencies, the usage of self-attention is not compared to other methods; this could be an interesting question for future research.

Commonly, the separation of the computation in spatial and temporal streams is done with the goal of reducing the computational complexity. To analyse this aspect for the compared model variants, table 6.11 shows the mean inference times during testing and the number of model parameters for both datasets and all three analysed model variants. A comparison during training is not possible as the conducted experiments are spread across multiple GPUs with different characteristics. The differences in the numbers of parameters for the models that are trained on the Lower Saxony and Dynamic Earth datasets are caused by the different hyper-parameters used in these experiments (cf. section 5.2.1).

	$\mathcal{T}_{te}^{L}$		$\mathcal{T}_{te}^{D}$	
Variant	t_i	$\#p$	t_i	$\#p$
$V\text{-}FE_{hyb}$	0.042	26.2	0.044	39.4
$V\text{-}FE_{att}$	0.038	25.3	0.035	33.3
$V\text{-}FE_{full}$	0.069	26.2	0.104	35.7

Table 6.11: Overview of inference times t_i [ms] and number of parameters $\#p$ [mio.] for the different model variants on the test dataset of Lower Saxony and Dynamic Earth. The inference times are averages over all input timeseries of the test dataset during evaluation on the same graphic card (Nvidia GeForce RTX 3090), i.e. the time t_i is the time needed to classify one input $X \in \mathbb{R}^{T\times B\times H_0\times W_0}$. A comparison of training times is not possible as different GPUs were used.

The inference times are relatively similar for variants $V\text{-}FE_{hyb}$ and $V\text{-}FE_{att}$, with a slightly faster processing for $V\text{-}FE_{att}$. This can be explained by the different computational settings for self-attention and convolution. While in variant $V\text{-}FE_{att}$ self-attention is computed in local windows that do not overlap, the convolutional kernels in variant $V\text{-}FE_{hyb}$ are applied to each pixel, shifting the kernel over the input feature map. Therefore, the convolutional operation is repeated many more times than the corresponding self-attention which can explain the slightly higher time for $V\text{-}FE_{hyb}$. As expected, the inference times for variant $V\text{-}FE_{full}$ are higher than for both other variants. On the Lower Saxony dataset, the inference time for variant $V\text{-}FE_{full}$ is 1.6 and 1.8 times higher than the ones for variants $V\text{-}FE_{hyb}$ and $V\text{-}FE_{att}$, respectively. On the Dynamic Earth dataset the difference is even higher, as the inference time for variant $V\text{-}FE_{full}$ is 2.4 and 3 times higher than the ones for variants $V\text{-}FE_{hyb}$ and $V\text{-}FE_{att}$, respectively. This behaviour is expected, because the self-attention operation in $V\text{-}FE_{full}$ is applied to three instead of two dimensions, increasing the number of input patches to one self-attention block from $M \times M$ to $T \times M \times M$. To conclude, the

proposed model $V\text{-}FE_{hyb}$ achieves better accuracy scores compared to variant $V\text{-}FE_{att}$ that needs similar inference times and achieves slightly higher or similar accuracy scores compared to variant $V\text{-}FE_{full}$, which requires inference times that are higher by a factor between 1.6 and 3.

6.3 Evaluation of the convolutional patch generation (Exp. 3)

In this section, the results regarding the 3D-convolutional patch generation module are discussed. This module is used in the proposed model $V\text{-}FE_{hyb}$ and is based on 3D convolutions to extract spatio-temporal features at the original spatial resolution of the input SITS. This module is compared to a patch generation module based on 2D convolutions (variant $V\text{-}FE_{2D}$), the original patch generation module from the Swin Transformer (variant $V\text{-}FE_{swin}$), a 3D patch generation module with an additional skip connection (variant $V\text{-}FE_{skip}$) and the usage of a patch size of $P = 1$ (variant $V\text{-}FE_{P1}$). In the following, the results on the Lower Saxony and Kleve dataset are discussed in section 6.3.1, before the results in the Dynamic Earth dataset are discussed in section 6.3.2. Afterwards, the results are summarised and discussed with regard to research questions 3 and 4 stated in chapter 1.

6.3.1 Results on the Lower Saxony and Kleve datasets

Results on the Lower Saxony dataset: Table 6.12 reports the overall accuracies as well as the mean F^1-scores on the two test datasets $\mathcal{T}^L_{cor}$ and $\mathcal{T}^L_{te}$ of Lower Saxony. In addition, table 6.13 shows the individual F^1-scores for all classes of the corrected test dataset. The F^1-scores on the other test datasets are reported in the appendix B.

Variant	$\mathcal{T}^L_{cor}$		$\mathcal{T}^L_{te}$	
	OA	mF^1	OA	mF^1
$V\text{-}FE_{hyb}$	86.3 ± 0.1	72.9 ± 0.4	**85.7** ± 0.1	78.7 ± 0.2
$V\text{-}FE_{swin}$	86.1 ± 0.1	72.3 ± 0.4	85.7 ± 0.1	**78.9** ± 0.2
$V\text{-}FE_{2d}$	85.7 ± 0.3	69.7 ± 1.1	84.2 ± 0.6	76.8 ± 0.7
$V\text{-}FE_{skip}$	**86.6** ± 0.0	**73.1** ± 0.4	**85.7** ± 0.0	78.7 ± 0.1
$V\text{-}FE_{P1}$	85.8 ± 0.3	70.7 ± 0.5	84.4 ± 0.1	75.5 ± 1.0

Table 6.12: OA and mF^1 in % for the experiments to evaluate the 3D-PG module that is used in variant $V\text{-}FE_{hyb}$, achieved on the Lower Saxony datasets $\mathcal{T}^L_{cor}$ and $\mathcal{T}^L_{te}$. All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations. Best scores are indicated in bold.

The proposed model $V\text{-}FE_{hyb}$, which uses the new patch generation module with 3D convolutions, the variant $V\text{-}FE_{swin}$, which uses the original patch generation module, and variant $V\text{-}FE_{skip}$, which uses the additional skip connection, all achieve a very similar performance. While $V\text{-}FE_{skip}$ achieves slightly better results on the corrected test dataset $\mathcal{T}^L_{cor}$ (+0.2% mF^1 compared to $V\text{-}FE_{hyb}$), variant $V\text{-}FE_{swin}$ achieves minimally better results on the test dataset $\mathcal{T}^L_{te}$ (+0.2% mF^1 compared to $V\text{-}FE_{hyb}$). However, these differences are small and none of them is statistically significant. Variant $V\text{-}FE_{2D}$, which uses 2D convolutions in the patch generation module, and

variant $V\text{-}FE_{P1}$ achieve lower accuracy scores than the other variants. Variant $V\text{-}FE_{2D}$ achieves a mF^1-score that is 3% lower on $\mathcal{T}_{cor}^L$ and 2% lower on $\mathcal{T}_{te}^L$. Both of these differences are statistically significant compared to model $V\text{-}FE_{hyb}$. The individual F^1-scores on the corrected test dataset $\mathcal{T}_{cor}^L$ reported in table 6.13 are very similar for the variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{swin}$ and $V\text{-}FE_{skip}$. Exceptions are the class *Barren land*, for which variant $V\text{-}FE_{hyb}$ achieves a 3% higher F^1-score than variant $V\text{-}FE_{swin}$, and the class *Vegetation*, for which variant $V\text{-}FE_{skip}$ achieves a 2% higher F^1-score than variant $V\text{-}FE_{swin}$. For variants $V\text{-}FE_{2D}$ and $V\text{-}FE_{P1}$, the F^1-scores for most classes are slightly worse than for variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{swin}$ and $V\text{-}FE_{skip}$. The largest drop in performance is obtained for the class *Coniferous forest* (-12% for variant $V\text{-}FE_{2D}$). The F^1-scores of *Agriculture* and *Water* are very close, differing by a maximum of 0.5%. On $\mathcal{T}_{te}^L$, variant $V\text{-}FE_{skip}$ achieves the highest F^1-scores for four of the classes and variant $V\text{-}FE_{swin}$ for two classes.

Figure 6.11 shows two examples (indicated by numbers in the figure) for qualitative results of the different model variants in comparison to the labels of the corrected test dataset $\mathcal{T}_{cor}^L$. Some of the results underline the obtained accuracy scores that were discussed in the previous paragraph. For instance, the outlines of forest and settlement areas look more blurred, i.e. edges are more round and straight lines are curved, for variant $V\text{-}FE_{2D}$, and the highway in example 2 is predicted to be thicker than the one predicted by the other variants or than it is in the reference. Furthermore, variant $V\text{-}FE_{2D}$ cannot distinguish between the two types of forest in example 2. These observations show why variant $V\text{-}FE_{2D}$ performs worse than the variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{swin}$ and $V\text{-}FE_{skip}$. The predictions of $V\text{-}FE_{hyb}$, $V\text{-}FE_{swin}$ and $V\text{-}FE_{skip}$ are very similar for most regions. There are some small differences, e.g. the street in the forest area in example 2 is interrupted in the prediction of variant $V\text{-}FE_{swin}$, which is not the case for variants $V\text{-}FE_{hyb}$ and $V\text{-}FE_{skip}$. Furthermore, the predictions of the two types of forest are more similar to the reference for variant $V\text{-}FE_{hyb}$. These qualitative examples underline the better performance of the new method, especially in comparison to variant $V\text{-}FE_{2D}$. In the following, the results of variant $V\text{-}FE_{P1}$ are analysed in more detail, because the obtained results contradict the expectation that a smaller patch size improves the general classification performance.

Evaluation of variant $V\text{-}FE_{P1}$: Variant $V\text{-}FE_{P1}$, which does not use any patch generation module, achieves worse results than $V\text{-}FE_{hyb}$, $V\text{-}FE_{swin}$ or $V\text{-}FE_{skip}$ on both test datasets. The differences in the mF^1-score are significant compared to the proposed model $V\text{-}FE_{hyb}$. The classes for which variant $V\text{-}FE_{P1}$ achieves significantly worse results are *Sealed area* and *Barren land*. Figure 6.11, showing some qualitative results of all model variants in comparison to the reference for $\mathcal{T}_{cor}^L$, helps to analyse these limitations in more detail. In comparison to the other variants, $V\text{-}FE_{P1}$ delivers sharper class borders, especially in comparison to variants $V\text{-}FE_{2D}$ and $V\text{-}FE_{skip}$, which can be seen clearly at object boundaries, e.g. for the settlement areas in circle 1. In some cases, variant $V\text{-}FE_{P1}$ classifies even finer structures than the reference. For instance, the highway access belonging to the class *Sealed area* in circle 2 is predicted very accurately by variant $V\text{-}FE_{P1}$, and even the forest areas surrounded by the highway access are classified, which is not the case for any other variant. Another example is the extended lake in circle 3. Variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{swin}$, $V\text{-}FE_{skip}$ and $V\text{-}FE_{2D}$ have problems classifying the whole region of the lake correctly; that is, for each of the variants some parts of the lake are missing. The prediction of variant $V\text{-}FE_{P1}$ is very close to the observed *Water* area in the corresponding satellite image. Furthermore, some

country roads with a width smaller than highways are only predicted with variant $V\text{-}FE_{P1}$, and the prediction of roads is not interrupted as often as for the other variants. These observed differences are all very small, which can be one reason why the quality indices do not indicate a better performance than the other variants. Of course, also some incorrect classification occurs for variant $V\text{-}FE_{P1}$, for instance between the classes *Vegetation* and *Agriculture* (circle 4) or the areas of *Barren land*, which is predicted to be larger than in the reference (circle 5).

In contrast to the obtained accuracy scores, the qualitative examples show the potential of using a smaller patch size. In several cases, the predicted LC maps of variant $V\text{-}FE_{P1}$ look more similar to the observed LC in the satellite imagery than the predictions of the other variants. The reason why the results are worse on average, especially compared to the test dataset $\mathcal{T}_{te}^{L}$ (-3% mF^1-score), are the even finer details in the prediction of $V\text{-}FE_{P1}$. These are sometimes not included in $\mathcal{T}_{te}^{L}$, caused by the automatic generation process of the labels. A further possible reason for slightly more wrong classifications can be the smaller receptive field in comparison to all other variants, as discussed in section 5.2.4.

Variant	*Stl*	*Sld*	*Agr*	*Veg*	*Dfor*	*CFor*	*Wat*	*Bar*
$V\text{-}FE_{hyb}$	87.9 ± 0.1	57.8 ± 0.7	93.8 ± 0.1	50.6 ± 0.7	78.8 ± 0.8	**74.4** ± 0.3	**89.0** ± 0.1	**50.9** ± 2.3
$V\text{-}FE_{swin}$	87.9 ± 0.1	57.7 ± 0.9	93.7 ± 0.1	49.5 ± 0.8	78.6 ± 0.2	73.7 ± 0.3	**89.0** ± 0.3	48.0 ± 2.1
$V\text{-}FE_{2d}$	87.7 ± 0.7	54.5 ± 0.4	93.7 ± 0.0	49.2 ± 0.7	77.8 ± 0.1	62.6 ± 3.5	**89.0** ± 0.7	43.1 ± 3.6
$V\text{-}FE_{skip}$	**88.1** ± 0.3	**58.5** ± 0.5	**93.9** ± 0.1	**51.3** ± 0.3	**80.0** ± 0.6	73.8 ± 0.2	**89.0** ± 1.0	50.2 ± 1.7
$V\text{-}FE_{P1}$	87.5 ± 0.7	52.2 ± 2.2	93.5 ± 0.2	50.7 ± 1.0	79.3 ± 0.2	70.7 ± 2.2	88.5 ± 0.5	42.9 ± 3.1

Table 6.13: F^1-scores for all classes in % on $\mathcal{T}_{cor}^{L}$ for different variants to analyse the 3D patch generation module that is used in the proposed model $V\text{-}FE_{hyb}$. All quality metrics are averages over three experiments, the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

Results on the Kleve dataset: The OA and mF^1-scores achieved by te compared model variants on the Kleve test dataset $\mathcal{T}_{kleve}^{L}$ are reported in table 6.14, and the individual F^1-scores are reported in table 6.15. The results on the Kleve dataset $\mathcal{T}_{kleve}^{L}$ are comparable to those on the two test datasets of Lower Saxony. Very similar performances are obtained for variants $V\text{-}FE_{hyb}$ and $V\text{-}FE_{skip}$. The slightly better OA for variant $V\text{-}FE_{skip}$ (+0.4%) is mainly caused by a higher F^1-score for the class *Agriculture* (+0.7%), which contains the highest number of pixels in the dataset. Variant $V\text{-}FE_{2D}$ achieves the lowest OA and mF^1-score again, and the difference of 2.2% in the mF^1-score compared to variant $V\text{-}FE_{hyb}$ is statistically significant. Variant $V\text{-}FE_{P1}$ performs better compared to variant $V\text{-}FE_{2D}$, but still 1.5% worse in the mF^1-score than variants $V\text{-}FE_{hyb}$ and $V\text{-}FE_{skip}$. However, this difference is not significant. The best F^1-scores are distributed between all variants (cf. table 6.15). Comparable F^1-scores are obtained for the classes *Water* (87-89%), *Settlement* (57-60%), and *Sealed area* (16-18%). A high drop in performance is obtained for the class *Coniferous forest* for variant $V\text{-}FE_{2D}$, which performs 10% worse for that class than all other variants. However, due to the low number of points (147 for the class *Coniferous forest*), some wrong classifications already have a drastic impact on the F^1-scores.

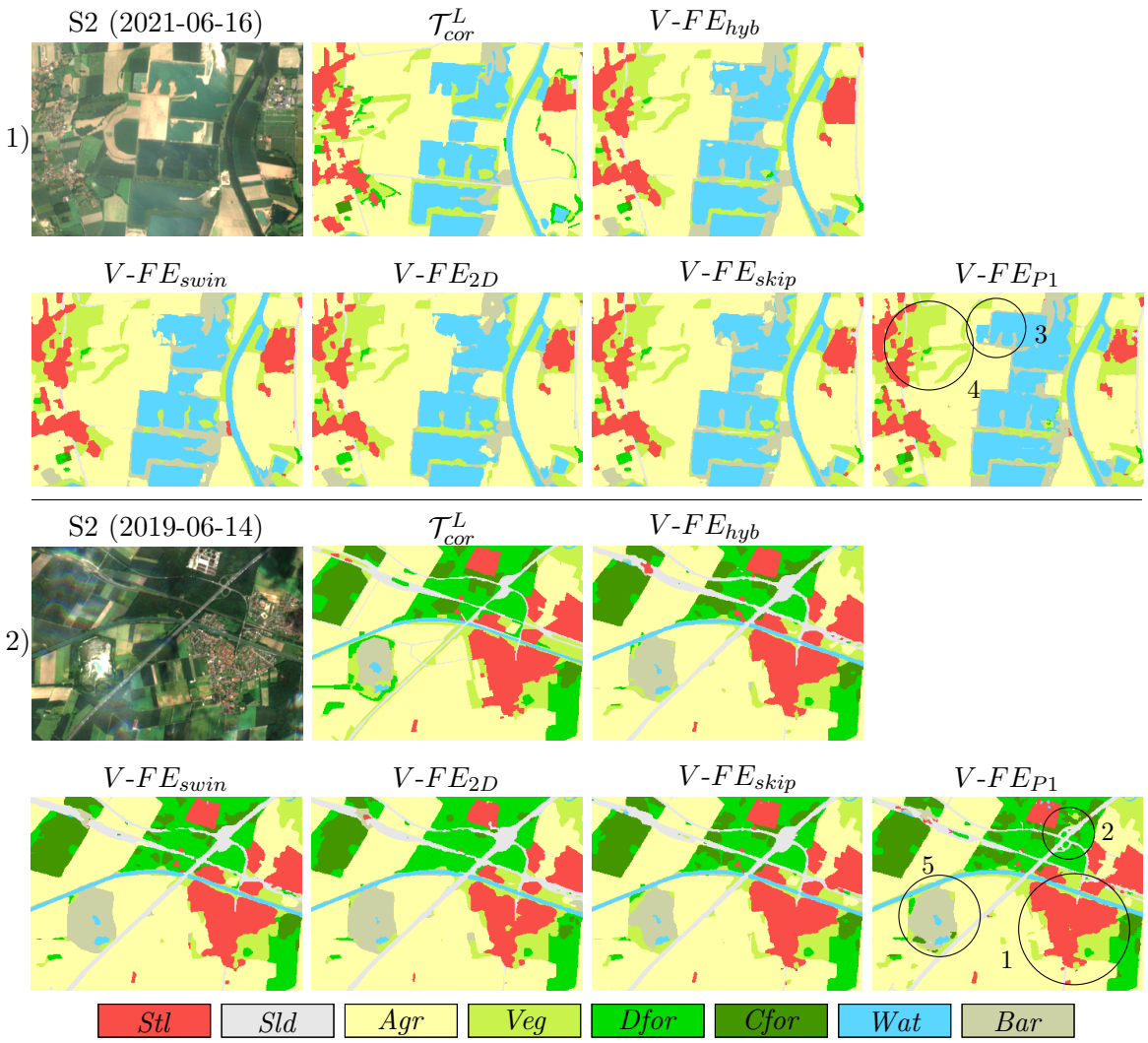

Figure 6.11: Qualitative results for variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{swin}$, $V\text{-}FE_{2D}$, $V\text{-}FE_{skip}$ and $V\text{-}FE_{P1}$ in comparison to the reference from the test dataset $\mathcal{T}^L_{cor}$.

Variant	$\mathcal{T}^L_{kleve}$	
	OA	mF^1
$V\text{-}FE_{hyb}$	72.3 ± 1.0	64.8 ± 0.8
$V\text{-}FE_{swin}$	71.6 ± 0.6	64.0 ± 0.5
$V\text{-}FE_{2d}$	70.0 ± 0.2	62.6 ± 0.1
$V\text{-}FE_{skip}$	**72.7** ± 0.6	**64.9** ± 0.4
$V\text{-}FE_{P1}$	71.3 ± 0.4	63.3 ± 0.7

Table 6.14: OA and mF^1-scores in % achieved on the Kleve test dataset $\mathcal{T}^L_{kleve}$ in the experiments to evaluate the 3D-PG module that is used in variant $V\text{-}FE_{hyb}$, . All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

Summary: To summarize the results on the Lower Saxony and Kleve datasets, the expectation of an improvement in classification performance by using 3D convolutions to extract spatial-temporal dependencies was not met , as the original patch generation module from the Swin

Variant	*Stl*	*Sld*	*Agr*	*Veg*	*Dfor*	*CFor*	*Wat*	*Bar*
$V\text{-}FE_{hyb}$	59.2 ± 0.6	**18.4** ± 0.5	86.1 ± 1.7	**69.6** ± 2.3	73.7 ± 1.8	81.2 ± 0.8	88.7 ± 0.3	41.3 ± 0.6
$V\text{-}FE_{swin}$	57.2 ± 0.1	16.2 ± 2.2	85.2 ± 2.0	66.8 ± 2.8	75.4 ± 0.9	**82.2** ± 0.9	88.5 ± 0.1	40.3 ± 3.2
$V\text{-}FE_{2d}$	**60.2** ± 0.9	**18.4** ± 0.4	82.6 ± 0.0	65.2 ± 1.0	73.5 ± 0.4	72.1 ± 2.6	86.7 ± 0.6	**41.9** ± 1.2
$V\text{-}FE_{skip}$	58.7 ± 1.1	17.9 ± 2.0	**86.8** ± 1.7	68.3 ± 2.3	**76.7** ± 1.8	82.1 ± 0.0	**89.2** ± 0.5	39.8 ± 1.3
$V\text{-}FE_{P1}$	57.0 ± 0.8	16.1 ± 2.0	85.7 ± 1.1	66.2 ± 0.8	74.3 ± 2.7	82.0 ± 1.3	88.4 ± 0.6	36.8 ± 4.5

Table 6.15: F^1-scores in % on $\mathcal{T}^L_{kleve}$ for different model variants differing by the patch generation. All quality metrics are averages over three experiments, the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

Transformer performs almost similarly well. However, compared to the variant that uses 2D convolutions for patch generation, the usage of 3D convolutions performs significantly better. Using the additional skip connection in variant $V\text{-}FE_{skip}$ resulted in minor improvements compared to $V\text{-}FE_{hyb}$ on $\mathcal{T}^D_{cor}$ and $\mathcal{T}^D_{kleve}$, but these differences are smaller than the corresponding standard deviations. Variant $V\text{-}FE_{P1}$, which uses a patch size of $P = 1$ and no patch generation module, achieves lower accuracies than the best performing variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{swin}$ and $V\text{-}FE_{skip}$. This can be caused by the smaller receptive field, which is reduced by a factor of 4 in the conducted experiments, resulting in less spatial context that can be used to classify a specific pixel. Furthermore, for variant $V\text{-}FE_{P1}$, the classified LC maps show more detailed results than all other variants and, in some cases, also than the test datasets. That the reference $\mathcal{T}^L_{te}$ does not include some fine details is caused by the rasterization process. This process results in the fact that pixels, covering fine structures such as streets by a small percentage, are assigned to the dominant class next to it. This can explain the differences in the accuracies to some extent.

6.3.2 Results on the Dynamic Earth dataset

Table 6.16 reports the accuracy metrics for the Dynamic Earth test dataset $\mathcal{T}^D_{te}$. The best OA and the best mF^1-score are achieved for variant $V\text{-}FE_{P1}$, with an OA of 68.4% and a mF^1-score of 43.8%. Regarding the mF^1-score, this variant is closely followed by variant $V\text{-}FE_{hyb}$ using the 3D patch generation module, with only -0.1% in the mF^1-score, whereas the OA is 2% better for variant $V\text{-}FE_{P1}$. $V\text{-}FE_{2D}$ achieves only slightly worse results than $V\text{-}FE_{hyb}$ and $V\text{-}FE_{P1}$, for instance the mF^1-score is 0.4% lower than for variant $V\text{-}FE_{P1}$. With an OA of 67.5%, variant $V\text{-}FE_{2D}$ achieves a comparably high OA, even higher than the one of $V\text{-}FE_{hyb}$. The larger OA is caused by a larger F^1-score of 65% for the class *Soil*, which is 2% higher than the one of variants $V\text{-}FE_{hyb}$ and $V\text{-}FE_{swin}$. The overall accuracy and mF^1-score of variant $V\text{-}FE_{skip}$ show similar results: While the mF^1 for $V\text{-}FE_{skip}$ is slightly lower than for $V\text{-}FE_{hyb}$ (-0.7%) or $V\text{-}FE_{P1}$ (-0.8%), variant $V\text{-}FE_{skip}$ achieves the second best OA. Again, this behaviour is caused by the larger F^1-score of the class *Soil*, which is 2.6% larger than the one for variant $V\text{-}FE_{hyb}$. As *Soil* covers more than 30% of the pixels in the test dataset, this has a relatively high impact on the OA, while the effect on the mF^1 is smaller. The worst accuracies are obtained for variant $V\text{-}FE_{swin}$, with a difference of 1.6% in mF^1 and 2.5% in OA compared to variant $V\text{-}FE_{P1}$. As the standard

deviations are also higher, especially for the mF^1-score of variants $V\text{-}FE_{swin}$ and $V\text{-}FE_{2D}$, none of these differences is statistically significant.

Similarly to the results obtained in section 6.2.2, the accuracies for the classes *Agriculture* and *Wetland* are very low, with F^1-scores between 0 and 3%. A class with higher F^1-scores and stable results over all variants is *Forest & vegetation*. Except for variant $V\text{-}FE_{skip}$, which achieves an F^1-score which is 0.6% lower than the second-best score, the F^1-score differs by a maximum of 0.2%. Slightly higher differences are achieved for the classes *Soil* and *Water* (maximum difference of 3.5% and 5% in the F^1-scores, respectively). For both classes, variant $V\text{-}FE_{P1}$ achieves the best F^1-scores. The largest differences and the largest standard deviations are obtained for the class *Impervious surface*, with a difference of -6% for variant $V\text{-}FE_{swin}$ compared to variant $V\text{-}FE_{hyb}$ or $V\text{-}FE_{skip}$. The standard deviations for class *Impervious surface* for variants $V\text{-}FE_{swin}$, $V\text{-}FE_{2D}$ and $V\text{-}FE_{skip}$ are very high, with ±12%, ±11.6% and ±12.1%, respectively. These results indicate a higher instability for these two variants when trained on the Dynamic Earth dataset.

Variant	OA	mF^1	*Imp*	*Agr*	*Veg*	*Wet*	*Sol*	*Wat*
$V\text{-}FE_{hyb}$	66.4 ± 2.3	43.8 ± 0.5	30.4 ± 1.5	1.8 ± 0.7	**75.2** ± 0.2	**2.0** ± 2.0	62.7 ± 4.7	90.9 ± 2.5
$V\text{-}FE_{swin}$	65.9 ± 0.9	42.3 ± 1.7	24.3 ± 12.0	2.4 ± 2.8	75.0 ± 0.5	0.4 ± 0.4	62.9 ± 2.9	89.0 ± 1.7
$V\text{-}FE_{2D}$	67.5 ± 1.2	43.5 ± 1.5	28.1 ± 11.6	1.9 ± 0.9	**75.2** ± 0.9	0.7 ± 0.4	65.0 ± 4.6	90.0 ± 1.9
$V\text{-}FE_{skip}$	67.6 ± 0.6	43.1 ± 2.0	**30.9** ± 12.1	0.2 ± 0.3	74.4 ± 0.5	0.3 ± 0.4	65.3 ± 0.9	87.4 ± 1.9
$V\text{-}FE_{P1}$	**68.4** ± 0.6	**43.9** ± 1.0	26.6 ± 7.4	**2.9** ± 1.2	75.0 ± 0.2	0.0 ± 0.0	**66.4** ± 1.2	**92.4** ± 0.6

Table 6.16: OA, mF^1 and individual F^1-scores for all classes in % on the Dynamic earth dataset $\mathcal{T}_{te}^D$ achieved by several variants that use different patch generation modules. All quality metrics are averages over three experiments, the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

Qualitative results: Figure 6.12 shows qualitative results for three examples (indicated by the numbers) of all variants compared in this section on the Dynamic Earth dataset, along with the input and the reference from $\mathcal{T}_{te}^D$. Note that example 2 shows a more detailed view of the central urban area from example 1. The predictions show a larger variation than those on the Lower Saxony dataset. For instance, the prediction of single buildings is much better for variant $V\text{-}FE_{P1}$, but it is still not as detailed as in the reference (example 2). In the same area, variant $V\text{-}FE_{2D}$ predicts only a few pixels as *Impervious surface* and in contrast, variants $V\text{-}FE_{swin}$ and $V\text{-}FE_{skip}$ predicts almost the whole area as *Impervious surface*. For the proposed variant $V\text{-}FE_{hyb}$, some individual buildings are separated, but most of the buildings are connected. Another example in which variants $V\text{-}FE_{P1}$ and $V\text{-}FE_{hyb}$ achieve better results than the two other ones is the river shown in example 1. Variants $V\text{-}FE_{2D}$, $V\text{-}FE_{swin}$ and $V\text{-}FE_{skip}$ only partly predict the river as a *Water* area, and the areas that are labeled as *Wetland* in the reference are assigned to the class *Forest & vegetation* or *Soil*. Variants $V\text{-}FE_{hyb}$ and $V\text{-}FE_{P1}$ are able to predict the river completely as *Water*, but these variants also fail to correctly predict the areas labeled as *Wetland*. The area shown in example 3 is difficult to classify correctly for all model variants. All models predict a much larger area as *Forest & vegetation* in the right part of the image and the separation between the classes *Impervious surfaces* and *Soil* is also difficult for all variants. Comparably good results are obtained by variant $V\text{-}FE_{skip}$, which predicts several of the single buildings in circle 1 correctly,

which is not the case for the other variants. However, also for $V\text{-}FE_{skip}$, the buildings are much larger than those in the reference.

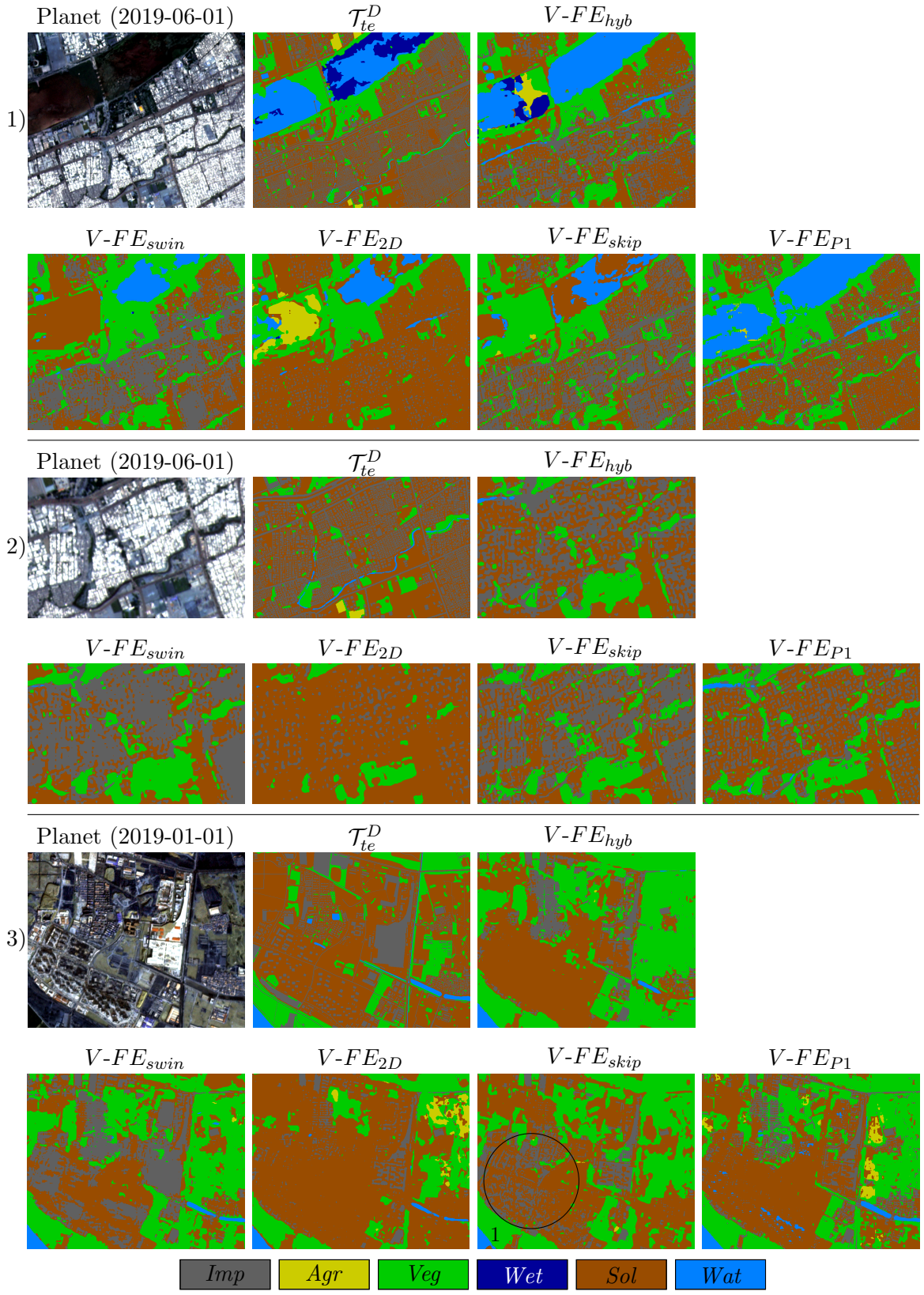

Figure 6.12: Qualitative results for the image patches for variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{swin}$, $V\text{-}FE_{2D}$, $V\text{-}FE_{skip}$ and $V\text{-}FE_{P1}$ in comparison to the reference from the test dataset $\mathcal{T}_{te}^{D}$.

Summary: To conclude, the proposed 3D patch generation module, used in variant $V\text{-}FE_{hyb}$, performs better than the standard patch generation module (variant $V\text{-}FE_{swin}$) on the Dynamic Earth dataset. It also slightly outperforms variant $V\text{-}FE_{2D}$, which uses the 2D patch generation module, and variant $V\text{-}FE_{skip}$, which uses the additional skip connection, in terms of the mF^1-score. For both variants, the achieved overall accuracy is about 1% better than the one of variant $V\text{-}FE_{hyb}$. However, due to higher standard deviations of variants $V\text{-}FE_{swin}$, $V\text{-}FE_{2D}$ and $V\text{-}FE_{skip}$, none of the differences in the mF^1-scores are statistically significant. The variant without any patch generation ($V\text{-}FE_{P1}$) achieves the best performance in the mF^1-score and in the overall accuracy, but the difference compared to the second-best variant ($V\text{-}FE_{hyb}$) is minor. As the input images had to be reduced in size by a factor of four in height and width to be able to train variant $V\text{-}FE_{P1}$ compared to all other variants, the receptive field is also smaller by a factor of four in height and width. This might have a negative impact on the model's performance and, therefore, the improvements of $V\text{-}FE_{P1}$ might be even larger if the same input size can be used. The qualitative results show that the smaller patch size results in much finer details in the predictions, which shows that the patch generation process has an impact on the classification results, independently of the strategy to combine pixels into patches.

6.3.3 Discussion

The results on the different test datasets do not show a significant tendency towards one of the tested model variants. In all cases, the proposed model $V\text{-}FE_{hyb}$, which uses the new 3D patch generation module, achieves the second-best results, however the best performing variant varies between $V\text{-}FE_{swin}$, $V\text{-}FE_{skip}$ on the Lower Saxony and Kleve dataset and $V\text{-}FE_{P1}$ on the Dynamic Earth dataset. Based on the discussed results, research questions 3 and 4 from chapter 1 can be discussed:

3. Does the proposed patch embedding module, utilizing 3D convolutions to encode spatio-temporal dependencies, achieve better performance compared to standard patch embedding strategies such as the one in the Vision Transformer (Dosovitskiy et al., 2021) or when compared to using 2D convolutions to consider spatial context only?

4. Does an additional skip connection at a higher spatial resolution than the one of the feature maps after patch embedding lead to improved classification performance?

Regarding the first part of research question 3, the expectation that 3D convolutions outperform the standard patch generation module from the Swin Transformer was only partly confirmed as the improvements on all datasets for variant $V\text{-}FE_{hyb}$ compared to variant $V\text{-}FE_{swin}$ are not significant. However, on three of the four test datasets, the proposed 3D patch generation module achieves better results than the standard patch generation. The qualitative results underline the slightly better performance of the 3D patch generation module, e.g. some finer details such as buildings are classified correctly, compared to the predictions of variant $V\text{-}FE_{swin}$. However, these details have a minor impact on the performance metrics, because they all only affect a small number of pixels, e.g. at object borders.

The second part of research question 3 cannot be answered clearly, as the results achieved on the different datasets are contradicting. While variant $V\text{-}FE_{hyb}$ performs significantly better than variant $V\text{-}FE_{2D}$ on the Lower Saxony dataset, these two variants perform similarly well on the Dynamic Earth dataset. However, when summarising the results on all datasets, variant $V\text{-}FE_{hyb}$ performs better than or similarly well as variant $V\text{-}FE_{2D}$, and therefore, it would be the method of choice for further application.

Using the additional skip connection in variant $V\text{-}FE_{skip}$ resulted in similar but slightly better classification results on the Lower Saxony and Kleve datasets, while the performance of variant $V\text{-}FE_{skip}$ was slightly worse on the Dynamic Earth dataset. None of the reported differences is statistically significant. Therefore, and to answer research question 4, the usage of the additional skip connection at a spatial resolution of $(H_0/2, W_0/2)$ did not result in significantly improved performance for any of the tested variants.

Variant $V\text{-}FE_{P1}$ was used to compare the results that can be achieved by omitting the patch generation completely. In general, this model was expected to outperform the variants that use a patch generation module with a patch size of $P = 4$, but, the receptive field had to be reduced by the factor of the patch size, which might have an impact on the classification performance of this variant. The expectation of better accuracy scores was only confirmed by the accuracies on the Dynamic Earth dataset, for which $V\text{-}FE_{P1}$ performs best. On the Lower Saxony dataset, variant $V\text{-}FE_{P1}$ achieves lower scores than variants $V\text{-}FE_{hyb}$, $V\text{-}FE_{swin}$ and $V\text{-}FE_{skip}$. However, the qualitative results for both datasets show the potential of using the input pixels as patches directly. In several predicted label maps, the object boundaries are sharper and small details like single buildings or streets can be predicted with much more detail. Some of these details are not included in the automatically generated dataset of Lower Saxony $\mathcal{T}_{te}^{L}$, which can be one reason for the slight decrease in performance on that dataset. In the Dynamic Earth dataset, many such fine details are included in the labels, which explains the better performance for $V\text{-}FE_{P1}$ on this dataset.

The main reason for using a patch size larger than one in a Transformer model is the increase in training and inference time when more input patches serve as inputs to self-attention layers. Table 6.17 shows an overview of the inference time needed for an area of 256×256 pixels and the corresponding number of model parameters for all variants compared in this section. The number of model parameters is only slightly affected by the different patch generation variants, which is reasonable as only several layers in the beginning or end of the models are adapted and all other stages stay similar as for the proposed variant $V\text{-}FE_{hyb}$. As variant $V\text{-}FE_{P1}$ classifies input patches of size 64×64 pixels, the inference time of this variant is multiplied by a factor of 16 for a fair comparison. The inference times show comparable values for variant $V\text{-}FE_{swin}$ and $V\text{-}FE_{2D}$, while variant $V\text{-}FE_{hyb}$, which uses the new 3D PG module, needs slightly more time on both datasets. This is expected because the 3D convolutions compute features based on shifting the kernel matrix in three dimensions before any downsampling layer. Variant $V\text{-}FE_{skip}$, which uses the additional skip connection, slightly increases the inference time in comparison to variant $V\text{-}FE_{hyb}$. $V\text{-}FE_{skip}$ is also the variant with the highest number of model parameters, increasing the number of trainable parameters by 1.5 million on both datasets compared to variant $V\text{-}FE_{hyb}$. This slight increase is caused by the additional convolutional layers that are needed to integrate

the encoder features into the last decoder layers. A drastic increase of the inference time can be observed for variant $V\text{-}FE_{P1}$, which needs a time that is larger by a factor of nine on $\mathcal{T}_{te}^{L}$ and larger by a factor of twelve on $\mathcal{T}_{te}^{D}$, in comparison to variant $V\text{-}FE_{hyb}$. The increase by such a large factor is expected because the models $V\text{-}FE_{P1}$ and $V\text{-}FE_{hyb}$ are equal except for the patch generation module. This means that, while variant $V\text{-}FE_{hyb}$ classifies a region of 256×256 pixels by classifying one input timeseries X, variant $V\text{-}FE_{P1}$ needs to classify 16 input timeseries X with the reduced spatial dimension of 64×64 to classify the same region on the ground.

To summarise, variants $V\text{-}FE_{P1}$, $V\text{-}FE_{skip}$ and $V\text{-}FE_{hyb}$ are all based on the new FE_{hyb} module and achieve the overall best results when comparing them to the standard patch generation or the patch generation based on 2D convolutions. Furthermore, some qualitative results indicate a slightly better performance for variant $V\text{-}FE_{P1}$ to classify fine structures such as single buildings or streets, which is not possible at this level of detail with all other variants. However, this comes at the cost of a drastically increased time for inference.

			$\mathcal{T}_{te}^{L}$		$\mathcal{T}_{te}^{D}$	
Variant	PE	P	t_i	$\#p$	t_i	$\#p$
$V\text{-}FE_{hyb}$	3D	4	0.042	26.2	0.044	39.4
$V\text{-}FE_{swin}$	Swin	4	0.029	25.8	0.036	39.4
$V\text{-}FE_{2D}$	2D	4	0.031	25.8	0.037	39.4
$V\text{-}FE_{skip}$	3D	4	0.044	27.6	0.045	40.7
$V\text{-}FE_{P1}$	-	1	0.384	26.0	0.528	39.3

Table 6.17: Overview of inference times t_i [ms] and number of parameters $\#p$ [mio.] for the different model variants in the experiment set analysing the patch generation module. The inference times are averages over all input timeseries of the test dataset during evaluation on the same graphic card (Nvidia GeForce RTX 3090), i.e. the time t_i is the time needed to classify one input $X \in \mathbb{R}^{T\times B\times H_0\times W_0}$. One exception is variant $V\text{-}FE_{P1}$, for which the inference time is multiplied by a factor of 16 to compare the inference time needed to classify the same area on the ground in comparison to the other variants. A comparison of training times is not possible as different GPUs are used.

6.4 Evaluation of the temporal weighting module (Exp. 4)

In this section, the results of the experiments regarding the temporal weighting module are evaluated. This is done to answer research question 5, whether the proposed temporal weighting module that is used in all model stages improves the performance compared to a model that does not use this module, as stated in chapter 1. The comparison includes the variant $V\text{-}FE_{hyb}$, which uses the temporal weighting module in the skip connections of all model stages, and the variant $V\text{-}FE_{noTW}$ that does not use any temporal weighting module. First, the results on the Lower Saxony and Kleve dataset are analysed, which is followed by the evaluation of the results on the Dynamic Earth dataset. In section 6.4.3, the results on all datasets are summarised and discussed in general with respect to research question 5.

6.4.1 Results on the Lower Saxony and Kleve datasets

The overall accuracies as well as the mF^1-scores for the test datasets of Lower Saxony and Kleve achieved in the experiments analysed in this section are reported in table 6.18. Additionally, table 6.19 reports the individual F^1-scores of all classes for the corrected test dataset $\mathcal{T}^L_{cor}$, while the F^1-scores on the test dataset $\mathcal{T}^L_{te}$ can be found in appendix C. Note that the accuracy scores of variant *V-FE*$_{hyb}$ are the same as in the previous sections.

Variant	$\mathcal{T}^L_{cor}$		$\mathcal{T}^L_{te}$		$\mathcal{T}^L_{kleve}$	
	OA	mF^1	OA	mF^1	OA	mF^1
V-FE$_{hyb}$	**86.3** ± 0.1	72.9 ± 0.4	**85.7** ± 0.1	**78.7** ± 0.2	72.3 ± 1.0	64.8 ± 0.8
V-FE$_{noTW}$	86.1 ± 0.1	**73.5** ± 0.5	85.4 ± 0.1	78.5 ± 0.3	**72.9** ± 0.0	**65.1** ± 0.4

Table 6.18: OA and mF^1-scores for all classes in % for the experiments regarding the temporal weighting module achieved on the Lower Saxony and Kleve datasets. All quality metrics are averages over three experiments, the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

The proposed model *V-FE*$_{hyb}$ achieves a slightly better mF^1 and OA on the test dataset $\mathcal{T}^L_{te}$ (0.2%), while variant *V-FE*$_{noTW}$, which does not use the temporal weighting module, achieves slightly better mF^1-scores on the corrected test dataset $\mathcal{T}^L_{cor}$ (0.6%) and on the Kleve dataset $\mathcal{T}^L_{kleve}$ (0.3%). As the standard deviations are also in the range of these improvements, none of these differences is statistically significant. The class-specific F^1-scores on the corrected test dataset $\mathcal{T}^L_{cor}$, as reported in table 6.19, are all very similar for both variants. The largest difference in the F^1-score is obtained for the class *Barren land* with a decrease in the performance of 2.5% for model *V-FE*$_{hyb}$, which is the reason why the mF^1-score is higher for variant *V-FE*$_{noTW}$. However, for this class, the standard deviations are also in the range of the improvement, with 2.3% and 2.8% for variants *V-FE*$_{hyb}$ and *V-FE*$_{noTW}$, respectively. This indicates that the general performance for this class is more unstable than it is for the other classes, and consequently, this difference is not significant. For all other classes, the differences are below 1% with no clear tendency towards one of the two compared variants.

To summarise, the temporal weighting module does not have a strong positive or a negative impact on the model's performance when it is evaluated on the Lower Saxony dataset. All performance metrics are relatively similar and the differences are in the range of the corresponding standard deviations.

Variant	*Stl*	*Sld*	*Agr*	*Veg*	*Dfor*	*CFor*	*Wat*	*Bar*
V-FE$_{hyb}$	**87.9** ± 0.1	57.8 ± 0.7	**93.8** ± 0.1	50.6 ± 0.7	78.8 ± 0.8	**74.4** ± 0.3	89.0 ± 0.1	50.9 ± 2.3
V-FE$_{noTW}$	87.8 ± 0.1	**58.4** ± 0.2	93.6 ± 0.0	**51.3** ± 0.9	**79.7** ± 0.0	74.1 ± 0.3	**89.5** ± 0.0	**53.4** ± 2.8

Table 6.19: F^1-scores for all classes in % on the corrected test dataset of Lower Saxony $\mathcal{T}^L_{cor}$, achieved by the two model variants with and without the temporal weighting module. All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

6.4.2 Results on the Dynamic Earth dataset

The overall accuracies, mF^1-scores, as well as the individual F^1-scores schieved on the Dynamic Earth dataset in the experiments to analyse the temporal weighting module are reported in table 6.20. The proposed model $V\text{-}FE_{hyb}$ obtains a mF^1-score that is 1.9% higher than the one for model variant $V\text{-}FE_{noTW}$, which is a significant improvement. The F^1-score of the classes *Forest & vegetation* and *Water* are similar. The classes *Agriculture* and *Wetland*, which are the classes with very low F^1-scores in general, are not differentiated at all by variant $V\text{-}FE_{noTW}$. The F^1-score of the class *Impervious Surfaces* is 11% higher for the proposed model $V\text{-}FE_{hyb}$ compared to variant $V\text{-}FE_{noTW}$. The F^1-scores achieved for this class are stable for both variants, with a standard deviation of ±1.5% for $V\text{-}FE_{hyb}$ and ±2% for $V\text{-}FE_{noTW}$, which underlines the better performance of variant $V\text{-}FE_{hyb}$ for the class *Impervious surfaces*. The F^1-score of the class *Soil* is 4% higher for $V\text{-}FE_{noTW}$ in combination with a much smaller standard deviation. In contrast, the standard deviation for this class reaches almost ±5% for variant $V\text{-}FE_{hyb}$, indicating that these results are more instable in comparison. The higher F^1-score causes the higher OA for variant $V\text{-}FE_{noTW}$, as this class covers over 30% of the test dataset $\mathcal{T}_{te}^{D}$.

In summary, the obtained mF^1-score for the proposed variant $V\text{-}FE_{hyb}$ is significantly better than the mF^1-score for the compared variant $V\text{-}FE_{noTW}$. Furthermore, for five of the six LC classes for the Dynamic Earth dataset, variant $V\text{-}FE_{hyb}$ achieves higher accuracy scores compared to variant $V\text{-}FE_{noTW}$. Thus, the temporal weighting module has a positive impact on the classification performance of the Dynamic Earth dataset, especially for the class *Impervious surfaces*, while for the class *Soil* no temporal weighting results in a 4% higher F^1-score.

Variant	OA	mF^1	*Imp*	*Agr*	*Veg*	*Wet*	*Sol*	*Wat*
$V\text{-}FE_{hyb}$	66.4 ± 2.3	**43.8** ± 0.5	**30.4** ± 1.5	**1.8** ± 0.7	**75.2** ± 0.2	**2.0** ± 2.0	62.7 ± 4.7	**90.9** ± 2.5
$V\text{-}FE_{noTW}$	**68.2** ± 0.7	41.9 ± 0.3	19.5 ± 2.0	0.2 ± 0.1	74.8 ± 0.6	0.0 ± 0.0	**67.0** ± 1.2	90.2 ± 0.8

Table 6.20: OA, mF^1 and individual F^1-scores for all classes in % on the Dynamic earth dataset $\mathcal{T}_{te}^{D}$ achieved by the two with and without the temporal weighting module. All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

6.4.3 Discussion

In this section, the introduced temporal weighting module TW was analysed by comparing the proposed model $V\text{-}FE_{hyb}$ in which the TW module is used, to variant $V\text{-}FE_{noTW}$ that does not use the TW module in the skip connections. Analysing the results on the different datasets, research question 5 can be answered:

5. Does integrating the proposed temporal weighting module into the skip connections throughout all model stages enhance the classification performance compared to not using any temporal weighting?

While the usage of the temporal weighting module has no significant impact on the classification performance of the Lower Saxony dataset, there is a significant improvement of 1.9% on the Dy-

namic Earth dataset with respect to the mF^1-score. The results show that the additional weighting of the features extracted in the spatial stream of the corresponding encoder module by using the features extracted in the temporal stream as weights in all model stages can increase the classification performance, but not necessarily for all datasets. A possible reason might be the higher percentage of class changes during the observed time period included in the Dynamic Earth dataset. For timesteps in which changes in LC occur, information encoded in the temporal stream might be more important, which can be a reason that the temporal weighting module (TW) module results in better performance on this dataset.

For variant $V\text{-}FE_{noTW}$ the inference times and number of parameters are approximately the same as those for the proposed model $V\text{-}FE_{hyb}$ (cf. table 6.11) as the matrix multiplication replaces other convolutional layers that are otherwise used to map the encoder features to the needed dimension of the decoder. For these reasons, the temporal weighting module can additionally be used, but the classification performance does not improve in all cases. It could be the case that the impact of this module is more important for applications that are even more dependent on temporal information, like change detection or crop mapping, but this would require further experiments that are beyond the scope of this thesis.

6.5 Comparison to other methods (Exp. 5)

In the final section of the experiment results, the proposed method is compared to two baseline approaches from the literature, as introduced in section 5.2.6. The models selected for this comparison are those that are most similar to the proposed method according to the literature review of chapter 3. The first baseline is a multi-temporal FCN that is purely based on convolutional layers and was already used for multi-temporal LC classification in previous work (Voelsen et al., 2023). The second one is the Utilise model, adapted from Stucker et al. (2023) to the task of multi-temporal LC classification by modifying the last layer of the model. In the following, the results of these two variants are reported and discussed. First, the results on the Lower Saxony and Kleve datasets are described in section 6.5.1, before the results on the Dynamic Earth dataset are presented in section 6.5.2. For the Dynamic Earth dataset, the results from a U-Net used by Toker et al. (2022) to publish first results on their new dataset is also used for the comparison.

6.5.1 Results on the Lower Saxony and Kleve datasets

The overall accuracies as well as the mF^1-scores for the proposed method $V\text{-}FE_{hyb}$ in comparison to the two baseline variants on the Lower Saxons and Kleve test datasets are reported in table 6.21. Additionally, the F^1-scores for the individual classes on the corrected test dataset $\mathcal{T}^L_{cor}$ are reported in table 6.22. The class-specific F^1-scores on the test datasets $\mathcal{T}^L_{cor}$ and $\mathcal{T}^L_{kleve}$ are reported in appendix D. In the following, the obtained results of the FCN baseline are discussed first, before the results from the *Utilise* model are analysed.

Comparison to FCN: The multi-temporal FCN achieves lower mF^1-scores and overall accuracies on all three test datasets than the proposed model $V\text{-}FE_{hyb}$. The differences in the

Variant	$\mathcal{T}^L_{cor}$		$\mathcal{T}^L_{te}$		$\mathcal{T}^L_{kleve}$	
	OA	mF^1	OA	mF^1	OA	mF^1
V-FE$_{hyb}$	**86.3** ± 0.1	**72.9** ± 0.4	**85.7** ± 0.1	78.7 ± 0.2	**72.3** ± 1.0	**64.8** ± 0.8
FCN	86.2 ± 0.0	72.7 ± 0.5	85.1 ± 0.4	78.2 ± 0.4	71.1 ± 0.4	63.6 ± 0.4
Utilise	86.0 ± 0.1	72.4 ± 0.1	**85.7** ± 0.3	**78.9** ± 0.6	71.9 ± 0.3	64.2 ± 0.1

Table 6.21: OA and mF^1-scores achieved on the Lower Saxony and Kleve datasets by the proposed method and two baseline models. All quality metrics are averages over three experiments, the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

mF^1-scores are small compared to the standard deviations (-0.2% on $\mathcal{T}^L_{cor}$, -0.5% on $\mathcal{T}^L_{te}$). When applied to the Kleve dataset, the difference of the *FCN* from *V-FE$_{hyb}$* is a bit higher in terms of the mF^1-score (1.2%). Nevertheless, this difference is not statistically significant because the standard deviations for *V-FE$_{hyb}$* are also in this range. The individual F^1-scores only vary slightly for the classes *Settlement*, *Sealed area*, *Deciduous forest* and *Water*, with differences of about 1% on all test datasets. Higher differences are obtained for the classes *Agriculture* and *Vegetation* on the Kleve dataset $\mathcal{T}^L_{kleve}$ (cf. appendix D), with -2.1% and -5.8%, respectively, which indicates a limited generalisation ability of the *FCN* for these classes.

Overall, the *FCN* achieves worse accuracies over all test datasets, but the differences are small and not statistically significant. In the FCN the temporal dimension is combined with the spectral dimension in the first layer of the model, but still the model can extract temporal dependencies from the input data by using different weights in the kernel matrices of the different spectral-temporal input channels. The results indicate that the purely convolutional model is able to extract most of the features that are important for multi-temporal LC classification.

Variant	*Stl*	*Sld*	*Agr*	*Veg*	*Dfor*	*CFor*	*Wat*	*Bar*
V-FE$_{hyb}$	87.9 ± 0.1	57.8 ± 0.7	**93.8** ± 0.1	**50.6** ± 0.7	78.8 ± 0.8	**74.4** ± 0.3	89.0 ± 0.1	**50.9** ± 2.3
FCN	**88.0** ± 0.1	**59.4** ± 0.4	93.7 ± 0.1	50.1 ± 0.4	**79.9** ± 0.3	72.7 ± 0.3	88.8 ± 0.5	49.5 ± 2.3
Utilise	87.9 ± 0.0	58.6 ± 2.1	93.6 ± 0.0	49.8 ± 0.5	78.7 ± 0.4	73.9 ± 0.9	**89.3** ± 0.1	47.1 ± 1.5

Table 6.22: F^1-scores for all classes in % on the Lower Saxony dataset $\mathcal{T}^L_{cor}$ achieved by variant *V-FE$_{hyb}$* in comparison to the two baseline models *FCN* (Voelsen et al., 2023) and *Utilise* (Stucker et al., 2023). All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

Comparison to *Utilise*: The *Utilise* model achieves slightly lower mF^1-scores compared to the proposed model *V-FE$_{hyb}$* on the test datasets $\mathcal{T}^L_{cor}$ (-0.5%) and $\mathcal{T}^L_{kleve}$ (-0.6%), while it achieves a slightly better mF^1-score on $\mathcal{T}^L_{te}$ (+0.2%). However, none of these differences is statistically significant. Similar to the results of the *FCN*, for most classes the F^1-scores only vary slightly between all three models. This applies to the classes *Settlement*, *Sealed area*, *Agriculture*, *Deciduous forest*, *Coniferous forest* and *Water*. Similar to the *FCN*, the performance for *Vegetation* is worse on the Kleve dataset $\mathcal{T}^L_{kleve}$ (-2.9% in the F^1-score, cf. appendix D), compared to *V-FE$_{hyb}$*. In

contrast to the results of the FCN, the performance for the class *Agriculture* is very close to the achieved results of $V\text{-}FE_{hyb}$ on all three test datasets, including the Kleve dataset. Higher differences are obtained for the class *Barren land*, with a drop in performance of -3.8% on $\mathcal{T}_{cor}^{L}$ for the $Utilise$ model compared to $V\text{-}FE_{hyb}$. On the test dataset $\mathcal{T}_{te}^{L}$ it is the opposite as the $Utilise$ model achieves a F^1-score that is 1.9% higher than the F^1-score achieved by $V\text{-}FE_{hyb}$.

6.5.2 Results on the Dynamic Earth dataset

The overall accuracies, mF^1-scores as well as the class-specific F^1-scores on the Dynamic Earth test dataset $\mathcal{T}_{te}^{D}$ for the proposed method $V\text{-}FE_{hyb}$ in comparison to the two baseline variants, FCN and $Utlise$, are shown in table 6.23. After the comparison of the proposed method to the two baselines, the achieved accuracies are also compared to those published by Toker et al. (2022). This comparison is based on the IoU scores on the validation dataset $\mathcal{T}_{val}^{D}$, as Toker et al. (2022) only published these scores.

Variant	OA	mF^1	*Imp*	*Agr*	*Veg*	*Wet*	*Sol*	*Wat*
$V\text{-}FE_{hyb}$	**66.4** ± 2.3	43.8 ± 0.5	30.4 ± 1.5	1.8 ± 0.7	75.2 ± 0.2	2.0 ± 2.0	**62.7** ± 4.7	**90.9** ± 2.5
FCN	65.0 ± 3.9	41.8 ± 1.6	24.9 ± 3.9	1.9 ± 2.1	74.6 ± 0.6	1.1 ± 1.2	58.2 ± 11.2	90.4 ± 0.5
$Utilise$	65.4 ± 1.3	**46.5** ± 0.5	**44.4** ± 1.6	**5.1** ± 0.5	**75.4** ± 0.2	**3.0** ± 1.4	**62.7** ± 2.7	88.6 ± 1.7

Table 6.23: OA, mF^1 and individual F^1-scores for all classes in % on the Dynamic Earth dataset $\mathcal{T}_{te}^{D}$ achieved by variant $V\text{-}FE_{hyb}$ in comparison to the two baseline models FCN (Voelsen et al., 2023) and $Utilise$ (Stucker et al., 2023). All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold.

Comparison to FCN: The FCN achieves the lowest mF^1-scores on the test dataset. The mF^1-scores is 2.0% worse than the mF^1-score achieved by variant $V\text{-}FE_{hyb}$. Due to the higher standard deviation of the FCN, this difference is not statistically significant. In general, the standard deviations of the FCN are higher than those of all other variants, indicating that the achieved results are more unstable than those of the other models.

Table 6.23 shows that the highest differences and standard deviations are obtained for the classes *Soil* and *Impervious surface*. The F^1-score for the class *Soil* is 58% for the FCN, which is 4% lower than the one for variant $V\text{-}FE_{hyb}$. For the class *Impervious surface*, the F^1-score for the FCN is 5.5% lower than the one for variant $V\text{-}FE_{hyb}$. Both classes have a comparably high standard deviation of 11% for *Soil* and almost 4% for *Impervious surface* for the FCN. These values show that the FCN can achieve a similar performance as the proposed method for these classes, but a stable, high accuracy is not obtained, as the performance drops drastically for other experiment runs with the same setting. For the other classes, the performance for the FCN is only slightly lower, with differences not exceeding 1-2%.

Comparison to *Utilise*: The $Utilise$ model achieves the highest mF^1-score on the test dataset $\mathcal{T}_{te}^{D}$ in comparison to the FCN and the proposed method $V\text{-}FE_{hyb}$, while the OA is lower than the OA of both $V\text{-}FE_{hyb}$ variants. The mF^1-score of the Utilise model is 2.7% higher compared

to the one achieved by model $V\text{-}FE_{hyb}$. This improvement is mainly caused by the F^1-score of the class *Impervious surface*, which reaches 44.4%, a value that is 14% higher than the F^1-score of the proposed model. For most other classes, the performance is on a similar level, with better results for the $Utilise$ for *Agriculture* (+3.3%), *Vegetation* (+0.2%) and *Wetland* (+1%), compared to variant $V\text{-}FE_{hyb}$. For the class *Water* the proposed variant $V\text{-}FE_{hyb}$ performs 2.3% better in terms of the F^1-score.

The F^1-scores on the validation dataset are also interesting. The mF^1-score is 45.1% for variant $V\text{-}FE_{hyb}$ and only 41.3% for the $Utilise$ model, which is a decrease of 3.9% for $Utilise$ compared to $V\text{-}FE_{hyb}$. The mF^1-score achieved on the validation dataset for Utilise is, therefore, 5% worse than its mF^1-score on the test dataset. No such difference occurs for the experiments with the proposed model $V\text{-}FE_{hyb}$, for which the mF^1-scores between the validation and test dataset differ only of about 1%. These results indicate some domain shift between the validation and test datasets, which is reasonable as this dataset covers different areas distributed over the whole Earth. Furthermore, the $Utilise$ model seems to generalise better to the test data, while model $V\text{-}FE_{hyb}$ performs better on the validation dataset.

Comparison to (Toker et al., 2022): The $mIoU$ as well as the individual IoU scores for the U-Net baseline from Toker et al. (2022) are compared to the proposed method $V\text{-}FE_{hyb}$ in table 6.24. Note that U-Net is used in this section as a name for the specific model used in Toker et al. (2022). Furthermore, the shown scores for $V\text{-}FE_{hyb}$ differ from all previous accuracy metrics, as the predictions from all timesteps are combined to a mono-temporal output score. Furthermore, the individual IoU scores are those from the validation dataset, as only these values are published in (Toker et al., 2022). Note that the comparison of the validation data is not independent of the training, as the validation data was used to select the hyperparameters and to terminate the training process based on the early stopping criteria.

	$\mathcal{T}^D_{te}$	$\mathcal{T}^D_{val}$						
Variant	$mIoU$	$mIoU$	Imp	Agr	Veg	Wet	Sol	Wat
$V\text{-}FE_{hyb}$	39.0 ± 0.9	39.7 ± 1.3	25.0 ± 0.2	24.2 ± 9.5	81.1 ± 1.7	0.1 ± 0.2	39.7 ± 0.7	67.8 ± 3.5
U-Net	37.6	33.5	28.6	6.9	76.4	0.0	38.4	50.5

Table 6.24: $mIoU$ and class specific IoU scores in % on $\mathcal{T}^D_{te}$ and $\mathcal{T}^D_{val}$ for the comparison between the proposed method $V\text{-}FE_{hyb}$ and the U-Net baseline from Toker et al. (2022).

The $mIoU$ scores reported for the proposed method are better than those from the U-Net on the test and validation datasets by 1.4% and 6.2%, respectively. As Toker et al. (2022) do not publish standard deviations of their quality metrics, it cannot be stated explicitly whether any of these differences are statistically significant. However, the improvement of 6.2% is much higher than the standard deviations of the proposed model $V\text{-}FE_{hyb}$, which indicates that this difference can be significant. The individual IoU scores on the validation dataset show clear improvements for model $V\text{-}FE_{hyb}$. For instance, for the class *Agriculture*, an IoU score of 24% is achieved, which is 17% higher than the one for the U-Net model. A similar behaviour is observed for the class *Water*, for which the scores improve by 17%. Smaller improvements are achieved for *Forest & vegetation* (+5%) and *Soil* (+1.3%), while the class *Wetland* is not differentiated correctly by any of the

models. The only class for which the U-Net achieves better results than $V\text{-}FE_{hyb}$ is *Impervious surface*, with an IoU score that is 3.6% larger than the one achieved by $V\text{-}FE_{hyb}$.

To summarise, the proposed method outperforms the U-Net baseline used in Toker et al. (2022), while the improvement on the test dataset is only slightly higher than the standard deviation of the $mIoU$-scores of model $V\text{-}FE_{hyb}$. It has to be mentioned that the comparison might include some more small differences in the training and evaluation processes, as discussed in section 5.2.6. Nevertheless, these results are in line with the observations made in the comparison to the multi-temporal FCN that was conducted earlier in this section.

6.5.3 Discussion

The multi-temporal FCN (Voelsen et al., 2023), the *Utilise* model (Stucker et al., 2023) and the U-Net used in (Toker et al., 2022), were employed for a comparison of their performance to the one of the proposed model $V\text{-}FE_{hyb}$. In addition to the results reported in this section, table 6.25 gives an overview of the inference times and numbers of model parameters for the compared models. With the achieved results, the last research question, stated in chapter 1 can be answered:

6. How do the results obtained with the proposed model compare to approaches from literature, specifically those using a multi-temporal FCN (Voelsen et al., 2023), the Utilise model (Stucker et al., 2023) and U-Net (Toker et al., 2022)?

Over all conducted experiments, the FCN variants achieve the worst results. This includes the multi-temporal FCN of Voelsen et al. (2023) trained on the Lower Saxony and Dynamic Earth datasets, as well as the U-Net model used in Toker et al. (2022). Additionally, the achieved standard deviations are higher for the FCN, indicating more unstable results with this method. Not all the differences between the convolutional models and the proposed method of this thesis are significant, indicating that a purely convolutional model is able to extract meaningful temporal information from the input timeseries to some extent. The comparison to the U-Net model, which predicts mono-temporal output maps, shows significantly worse results for the U-Net model, but the training and data generation processes differ in some details between the one of the U-Net and $V\text{-}FE_{hyb}$. The main advantage of the FCN baseline is the fast inference and training time, as shown in table 6.25. Whereas the number of parameters is in a comparable range to the one of the $V\text{-}FE_{hyb}$ model on the Dynamic Earth dataset, the inference times are lower by a factor of four on the Dynamic Earth dataset. For the Lower Saxony dataset, a model with less stages is used, as a smaller receptive field is suitable for that dataset. This is also the reason why the inference time is reduced by an even larger factor of seven for the FCN trained with the Lower Saxony dataset in comparison to the proposed method and an even larger factor in comparison to the *Utilise* model. To conclude and to answer the first part of research question 6, the proposed model outperforms all tested multi-temporal fully convolutional models on all datasets, including the multi-temporal FNC from Voelsen et al. (2023) and the U-Net used in Toker et al. (2022).

The Utilise model achieves a similar performance as the proposed model. While the results are slightly worse on the Lower Saxony dataset for *Utilise*, the results obtained on the Dynamic

Earth dataset are better. The lower performance of the *Utilise* model on the validation dataset is interesting, indicating that the model has a good generalisation performance, because the mean test accuracy is significant better than that on the validation dataset. Comparing the inference times and the number of parameters, shown in table 6.25, the Utilise model has very few parameters but requires a longer time for inference compared to $V\text{-}FE_{hyb}$. To answer the second part of research question 6, the proposed model achieves similar performance in comparison to the *Utilise* model from Stucker et al. (2023), as the achieved performance of the proposed model is better on one of the used datasets and worse on the second dataset. The results also show that models with a smaller number of parameters do not necessarily lead to faster inference times (cf. table 6.25).

Variant	$\mathcal{T}_{te}^{L}$		$\mathcal{T}_{te}^{D}$	
	t_i	$\#p$	t_i	$\#p$
$V\text{-}FE_{hyb}$	0.042	26.2	0.044	39.4
FCN	0.006	9.5	0.01	33.0
Utilise	0.066	0.9	0.066	0.9

Table 6.25: Overview of inference times t_i [ms] and number of parameters $\#p$ [mio,] for the different baseline models in comparison to the model proposed in this thesis. The inference times are averages over all inference times for all input timeseries of the test dataset during evaluation on the same graphic card (Nvidia GeForce RTX 3090).

7 Conclusions and Outlook

7.1 Conclusion

In this thesis, the task of multi-temporal land cover classification was addressed by proposing a new method to predict multi-temporal LC maps from SITS. The new model is based on a hybrid approach, combining self-attention and convolutions to extract meaningful features with suitable methods from input data. The generated output are multi-temporal LC maps, showing the land cover of the individual timesteps for every pixel in the input images. To capture spatial and temporal dependencies with suitable approaches, a new hybrid feature extraction module was introduced. This module uses convolutional layers in one of its two streams to capture spatial context for each input timestep in parallel. In the second stream, self-attention layers are used to capture temporal dependencies over all timesteps in the input images. This is done separately for each temporal sequence of patches. To further improve the classification performance, a new patch generation module based on 3D convolutions was proposed. This module extracts spatial and temporal context at the original spatial resolution of the input images, instead of merging them directly like in the standard patch generation module of the Swin Transformer. Furthermore, a temporal weighting module is used within the skip connections of the proposed model. This module was adapted from (Stucker et al., 2023), who use such a module only in the bottleneck layer of their model.

The new method was evaluated by conducting several sets of experiments on three datasets for multi-temporal LC classification. First, the general classification performance of the proposed model and its ability to correctly predict LC changes over time was analysed. The results allow to answer the first research question. In this context, the following statements can be made: The results of the proposed method have shown good classification performance, i.e. F^1-scores above 75%, for several LC classes for which a high number of training samples is available or for classes whose appearance in the satellite images is clearly distinguishable from others. Difficulties arise for classes with few training samples or classes with a similar appearance, the latter leading to several misclassifications between the similar classes. The prediction of LC changes is possible due to the generation of multi-temporal output maps, but many false positives decrease the IoU score for binary change.

In the following experiments, it could be shown that the proposed model achieves better performance than a model that is purely based on self-attention in the spatial and temporal streams. The results show that the usage of convolutions is better suited for capturing spatial context for the task of LC classification. The evaluation additionally included a third variant that computes joint spatio-temporal features by using self-attention, but this variant also achieved worse results

than the proposed model. These results are related to research question 2 as they show that the hybrid feature extraction in the proposed model leads to better results than modules that solely rely on self-attention. Besides, the computational costs are significantly reduced by factors between 1.6 and 3 by separating the computations in a temporal and a spatial stream.

The evaluation of the experiments regarding the new patch generation module shows that it performs equally well or better than the standard patch generation or a patch generation based on 2D convolutions. The analysis of the qualitative results further underlines its superior performance. For instance, on the Lower Saxony dataset, fewer streets are interrupted by gaps, and on the Dynamic Earth dataset, the prediction of river and settlement areas is more precise than for the other variants. To conclude, and to answer the third research question, the new 3D patch generation module could not significantly improve the results in comparison to the standard patch generation or the 2D patch generation. However, for all tested scenarios, the 3D patch generation achieves similar or better results than the compared variants. By using the new patch generation module, the model is able to extract spatial and temporal features at finer spatial resolutions compared to models employing the standard patch generation, where features are first extracted at a resolution of $(H/P, W/P)$. To incorporate this information into the later layers of the model, an additional skip connection has been introduced. This modification was made to analyse whether the integration of these features can further improve the model's performance, addressing the fourth research question. The conducted experiments show that using the additional skip connection resulted in no significant improvement of the results, since the achieved accuracies were comparable to those obtained by the proposed method without the skip connection. However, due to computational limitations, the skip connections could only be integrated at a resolution of $(H/2, W/2)$, and the accuracy might improve if another skip connection at the original spatial resolution (H, W) is integrated. The analysis of the patch generation included the investigation of the usage of patches in general. For this purpose, another variant without any patch generation was used. Instead, the image pixels are directly used as inputs, which corresponds to using a patch size of one. Whereas only the obtained accuracy scores on the Dynamic Earth dataset show a slight improvement, the qualitative results on both datasets show good classification results. For instance, finer details, such as outlines of object borders, were better preserved and are therefore closer in similarity to the way in which they appear in the corresponding label maps. Even more minute details like vegetation between individual streets were predicted correctly. It should be noted that the spatial size of the input had to be reduced by a factor of four in height and width to be able to train the model without the PG module. Therefore, the receptive field of this model in corresponding stages only covers an area which is sixteen times smaller than for the proposed model that uses PG with a larger input size, which can negatively impact the extraction of spatial features and the model performance.

The evaluation of the experiments with respect to the temporal weighting module used in all skip connections of the new model shows a significant improvement for one of the used datasets, whereas the classification performance on the other datasets stays at a similar level. The impact on the computation time and number of parameters, however, is minor because the features from the encoder need to be transformed to the feature dimension of the decoder regardless, and the temporal weighting module only introduces another matrix multiplication into this process. For

this reason, the new module is considered to be better compared to the standard strategy of fusing all features from the encoder stage and mapping them to the decoder dimension, even if the impact on the classification performance is only minor for some datasets. In conclusion, and with respect to the fifth research question, the integration of the temporal weighting module can significantly improve results, albeit for only one of the used datasets in this thesis.

To further analyse the proposed method, it was compared to two different models from literature that could easily be adapted to or were already introduced for the task of multi-temporal LC classification. The corresponding set of experiments has shown that the proposed method performs better than purely convolutional-based models. However, a FCN is also able to capture spatial and temporal dependencies to some extent as for most conducted experiments, the performance of the FCN did not decrease significantly in comparison to the other tested models. The proposed method performs similarly well as the Utilise model, which also combines convolutional and self-attention layers, but the inference times are 30% faster when using the new model. To summarise these findings and to answer the sixth research question, the comparison to models from the literature showed that the proposed model outperforms purely convolutional-based approaches while achieving similar results compared to another hybrid, i.e. the Utilise model.

Besides the answers to the stated research question, the experiments presented in this thesis showed some more interesting details. One challenge is the classification of fine objects like single buildings or narrow streets. An improvement in the qualitative results was achieved by removing the patch generation module at the beginning of the model. This improvement comes at the cost of drastically increasing the computation time. However, with more GPU resources, this limitation can be overcome. Another topic, which was not the primary focus of this thesis, is the detection of LC changes over time. Nevertheless, this topic is inherently integrated when predicting multi-temporal LC maps, which can be used for follow-up applications such as the update of topographic databases. Despite the low binary change detection score, which was mainly caused by a large number of false positives for *Change* in the change maps created by comparing two classification maps, the qualitative results also showed some promise. Good predictions were particularly obtained for changes affecting the classes *Settlement* and *Water*. However, accurately predicting the extent of changes or changes between classes with more similar appearances was more challenging. Considering the fact that for follow-up tasks like map updating, some classes (e.g. *Settlement*) are of higher importance than others, and that the exact spatial extent is often refined by human supervision, the results show the promising potential of deriving change explicitly from the model.

To conclude, the proposed contributions of this thesis made it possible to investigate the usage of self-attention, convolution, and patch generation in detail for the task of multi-temporal LC classification. With the obtained results, the research questions stated in chapter 1 were answered:

- First, the proposed method was analysed regarding its general classification performance and with respect to the prediction of occurring LC changes over time, which is related to research question 1. Strong classification performance is achieved for classes with many training samples or clear visual distinctions, but difficulties arise for classes with few samples

or similar appearances. The score for binary changes is low, mainly caused by many false positives for changes, but larger regions of LC changes are already predicted correctly.

- The second research question is related to the proposed hybrid features extraction module. The analysis of this module, which combines convolutions in the spatial dimensions with self-attention in the temporal one, showed that it achieves better performance compared to variants purely relying on self-attention.

- The experiments regarding the introduced 3D patch generation module that are related to research questions 3 and 4 showed that the introduced 3D patch generation module could slightly improve the performance compared to the standard patch generation from the Swin Transformer and a patch generation based on 2D convolutions. The additional integration of another skip connection, however, did not further improve the results.

- Research question 5 is related to the temporal weighting module that is used in the skip connections of the proposed method. The integration of this module could significantly improve the performance for one of the datasets, whereas the performance stayed similar on the other datasets.

- In the last set of experiments, the proposed method was compared to other models from literature. The new method outperforms other methods that are purely based on convolutions, whereas it performs similarly well compared to another hybrid model, which answers research question 6.

These findings show the potential of the proposed model for multi-temporal LC classification. In particular, the usage of hybrid techniques that combine suitable methods for different characteristics of the input data, which is done in the proposed model by combining the convolutional patch generation with feature extraction modules that are based on self-attention and convolution, significantly improved the results. Possible next steps to further improve the classification performance, e.g. by integrating changes explicitly into the model, are discussed in the following section.

7.2 Outlook

With the obtained results the research questions stated at the beginning of this thesis have been answered. However, the analysis also showed that some open questions remain. In the following, additional aspects that can be analysed in future work will be discussed.

Predicting land cover and land cover changes by a multi-task model: The new method introduced in this thesis predicts one LC map for each timestep. However, the model is not explicitly trained to identify changes over time. By modifying the model, for instance, by adding a head to predict land cover changes over time, it is expected that the performance for the task of LC classification improves correspondingly. An additional classification head could predict a binary classification map, predicting pixel by pixel whether a change between the first and last image of the timeseries occurred or not. A further extension could include predicting a binary change map for every epoch within an input timeseries. To force the model to focus on classifying

these changes in the training process, an additional loss term, e.g., the binary cross-entropy loss, computed based on the predicted change maps in comparison to those from the training data, can be added to the existing cross-entropy loss. The prerequisites for such a multi-modal model are training labels for land cover and land cover class changes, which both need to be available for multiple timesteps, such as in the case of the Dynamic Earth dataset (Toker et al., 2022). The prediction of changes in combination with the LC class is especially useful for applications like map updating, in which it is important to know the classes before and after a change. Challenges during the training process of such a multi-task model can arise when land cover change is a rare event , and land cover labels are available for the whole training region. Therefore, selection or weighting strategies might be useful to select training samples that include changes more frequently or give higher weights to those pixels assigned to change in the training data. Having an explicitly integrated classification of changes in the model is expected to obtain an improved performance for changes, which can then be used as a basis for updating topographic databases.

Integrating methods to mitigate the effect of label noise: For the training process of larger DL models, such as the applied Transformer model, larger training datasets help to reduce overfitting and improve the generalisation performance. However, large datasets with manual annotations are often not available. The usage of automatically generated training labels, as in the case of the Lower Saxony dataset that is used in this thesis usually results in a much larger variety of training samples, but also in *label noise*. Developing strategies to mitigate the effect of *label noise* is a separate research field (Song et al., 2023). Several methods can be integrated into the supervised training process of DL models, for instance, by estimating confidence scores of each label in the training dataset and using them to weight each sample differently (Song et al., 2023). Alternatives are, for instance, the modification of the model's architecture in a way that it is more robust against *label noise*. This can be done by adding a noise adaptation layer on top of the model, which is supposed to mimic the label transition for any pixels affected by *label noise* (Song et al., 2023). The difficulty of this strategy is the accurate determination of the noise adaptation layer, as it is usually difficult to differentiate between samples being hard to assign to one class and samples having wrong labels. The adaptation layer has to be determined for every dataset with its own specification again, making it difficult to estimate the potential of this method for the datasets used in this thesis. However, integrating such an approach in the applied training process of this thesis is expected to reduce the impact of the *label noise* in the case of the dataset of Lower Saxony. However, it remains an open question which method is best for the task of LC classification.

Self-supervised training: Self-supervision is another strategy to pre-train a model on the data that is used for downstream tasks. In this case, no labelled data is necessary, as a task is defined for which the output of the network is generated from the input data itself. Common strategies include masking parts of the input image in space or time and letting the model anticipate the missing values. This strategy has the advantage that the data to be used for the following application is already utilized in the pre-training step, and so no adjustments regarding the data type are necessary afterwards. A review about self-supervised learning in remote sensing is given by Wang et al. (2023). One approach that is closely related to the method in this thesis is the one proposed by Yuan et al. (2022). The authors mask random patches in time in the self-supervised pre-training stage. This strategy could be extended to masking random patches in space and time when SITS are

used as input data. A similar strategy is followed by Cong et al. (2022), who pre-train a Transformer model on SITS data by applying a self-supervised training approach that masks patches in space and time.

Using Foundation models: Another strategy to improve the generalisation performance for comparably small datasets is the usage of Foundation Models (FM). FMs are trained on large amounts of unlabelled or weakly-labelled data using self-supervised or semi-supervised training procedures for which commonly data from different modalities is integrated. The usage of such large and diverse training data results in large models (up to billions of parameters). After a FM is trained, it can be fine-tuned to various downstream tasks, e.g. by modifying and training the last layers of the model (Jiao et al., 2023). Recently, FMs have also been introduced into the field of remote sensing; Jiao et al. (2023) provide a survey about research in this field. There are also FM trained on SITS as input data. For instance, An et al. (2024) train a Foundation model in a self-supervised way through augmenting the input images and enhancing intra-instance similarity by using cosine similarity in the loss function.

Extending the method to input time series with varying length: The input image timeseries required for the method presented in this thesis is always based on a time period of a year and includes one image for each month. This restricts the applicability of the method to SITS data which also cover the required period. A strategy to overcome this limitation can be a model that is trained on varying time periods, including different lengths and intervals of the used inputs. Such a method could integrate all available images in a certain period or an automatic selection process to choose several images within a period. Models that are based on self-attention, as used in the temporal stream of the proposed model, can handle varying input length without modifications and are, therefore, appropriate for this task. Probably, a well-designed temporal encoding is more important in such a training scenario in which the temporal intervals might differ more than in the image selection strategy that is used in this thesis. This could mean that the applied temporal encoding, which is based on the DOY, has to be extended, e.g. by integrating the days since the last used image (useful when multiple years are taken into account). Furthermore, the application of larger models might enhance the ability to learn more complex and diverse temporal dependencies.

Usage of multi-modal data: For the applied task of LC classification, only the spectral information of one satellite constellation is used to generate the input timeseries. However, approaches that use multi-modal data commonly achieve improved classification performance due to the use of additional input data. Additional data can include height information, images from different optical sensors or additional information about the environment, e.g. about the cloud coverage or other climate factors which might impact the appearance of the observed objects. In this regard, it always depends on the application which additional data is useful. For the classification of LC, information about the surface height could, for instance, help to improve the classification performance for buildings in comparison to streets or other sealed areas with lower heights that can have a similar appearance in the satellite imagery (Wittich and Rottensteiner, 2021). In the case of cloud coverage, additional information from radar satellites can help to fill these data gaps in time.

Another common approach is the combination of SITS, which usually have a medium spatial resolution, e.g. 10 m for Sentinel-2, but a high temporal resolution, with aerial imagery that is usually mono-temporal but has a much higher spatial resolution (Heidarianbaei et al., 2024; Bergamasco et al., 2023; Sandmann et al., 2022). A common strategy is the extraction of features from both image data in two separate streams, resulting in output features that are combined for the final classification afterwards (Heidarianbaei et al., 2024; Sandmann et al., 2022). This strategy has the advantage that in the multi-temporal stream, suitable methods to extract temporal context can be used, e.g. LSTM (Sandmann et al., 2022) or self-attention (Heidarianbaei et al., 2024), while in the mono-temporal stream, spatial context can be captured, e.g. using convolutions (Sandmann et al., 2022; Heidarianbaei et al., 2024). One dataset which provides such data is the French Land cover from the Aerospace ImageRy (Flair) dataset (Garioud et al., 2023), consisting of multi-temporal Sentinel-2 images covering one year in combination with mono-temporal aerial images with a GSD of 20 cm and annotations for 13 LC classes. Such a dataset can be used to investigate this aspect and analyse if the usage of aerial imagery can improve the performance of classification tasks with SITS.

Integrating other Transformer-based approaches: The experiments conducted in this thesis are all based on the Swin Transformer, which achieves strong results for pixel-wise classification. However, research in the field of Transformer models is developing fast, and several approaches overcome some of the shortcomings of the Swin Transformer architecture. The main limitation of the Swin model is the restricted receptive field in the early layers of the model caused by the window partitioning. Several approaches integrate more global context again, for instance with a selection stratefy that extends the computation of self-attention in local windows by additionally computing self-attention between random connections of the input features (Zaheer et al., 2020). Other approaches fuse features from neighbouring patches to be able to capture dependencies between these fused regions (Wang et al., 2021) or reduce the dimension of the input matrices to be able to use a larger number of input patches (Zhang and Yan, 2023). On the other hand, computational resources are continuously improving, making it possible to use an increasing number of features as hardware advances.

Transition to other applications: The application in this thesis was achieving LC classification by utilizing SITS. However, the method is not restricted to this application and can be transfered to other tasks for which pixel-wise predictions are required. The method can also easily be adjusted to mono-temporal output predictions or regression tasks by modifying the last layer of the model. In the field of remote sensing, possible applications are building detection, crop classification, deforestation mapping, and flood detection. However, for different tasks and datasets an adjustment of the models' receptive field might be necessary, especially when significantly different GSDs are used. This could include more stages of the model or larger kernels when convolutional layers are used to capture spatial context.

To conclude, there are several topics that can be the focus for future research which are relevant for tasks like automatic map updating, urban planning or environmental analysis. Promising aspects that have the potential to significantly improve results are the application of additional data sources, e.g. aerial imagery, or the explicit integration of detecting land cover changes.

Acknowledgements

I would like to thank the Land Survey Office of Lower Saxony (Landesamt für Geoinformation und Landesvermessung Niedersachsen - LGLN) for providing the Lower Saxony dataset and for their financial support of this project. Furthermore, I would like to thank the EFTAS Fernerkundung GmbH for providing the Kleve dataset and the NVIDIA Corporation for providing GPU resources to this project.

Appendix

Appendix A: Class specific F^1-scores achieved in the experiments related to the hybrid feature extraction module

Variant	*Stl*	*Sld*	*Agr*	*Veg*	*Dfor*	*CFor*	*Wat*	*Bar*
$V\text{-}FE_{hyb}$	87.9 ± 0.1	**61.0** ± 0.2	**88.7** ± 0.0	**78.5** ± 0.4	**83.7** ± 0.3	**92.7** ± 0.1	**95.7** ± 0.1	**41.9** ± 0.3
$V\text{-}FE_{att}$	86.4 ± 1.0	57.6 ± 0.1	87.8 ± 0.5	77.4 ± 0.4	82.0 ± 0.6	91.8 ± 0.3	95.4 ± 0.1	35.7 ± 3.8
$V\text{-}FE_{full}$	**88.0** ± 0.1	60.8 ± 0.4	**88.7** ± 0.0	**78.5** ± 0.3	83.6 ± 0.3	**92.7** ± 0.1	95.6 ± 0.1	41.5 ± 0.7

Table A1: F^1-scores in % for all classes achieved on $\mathcal{T}_{te}^L$ by the compared model variants in the experiments of the hybrid feature extraction module. All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations. Best scores are indicated in bold font.

Appendix B: Class-specific F^1-scores achieved in the experiments related to the convolutional patch embedding module

Variant	*Stl*	*Sld*	*Agr*	*Veg*	*Dfor*	*CFor*	*Wat*	*Bar*
$V\text{-}FE_{hyb}$	**88.0** ± 0.1	60.8 ± 0.4	88.7 ± 0.0	78.5 ± 0.3	83.6 ± 0.3	**92.7** ± 0.1	95.6 ± 0.1	41.5 ± 0.7
$V\text{-}FE_{swin}$	87.6 ± 0.3	60.4 ± 0.4	88.8 ± 0.1	**78.6** ± 0.1	**83.9** ± 0.1	92.6 ± 0.1	95.5 ± 0.2	43.5 ± 0.5
$V\text{-}FE_{2d}$	85.9 ± 1.1	55.8 ± 1.2	88.4 ± 0.1	76.6 ± 0.7	81.3 ± 0.6	90.0 ± 1.0	95.5 ± 0.0	40.8 ± 0.8
$V\text{-}FE_{skip}$	87.8 ± 0.1	**61.6** ± 0.6	**89.0** ± 0.0	78.4 ± 0.2	83.1 ± 0.7	92.3 ± 0.2	**95.7** ± 0.2	**41.7** ± 0.6
$V\text{-}FE_{P1}$	86.5 ± 0.3	53.7 ± 2.2	88.3 ± 0.2	76.7 ± 0.0	81.4 ± 0.4	91.1 ± 0.5	95.6 ± 0.1	30.7 ± 6.7

Table A2: F^1-scores in % for all classes on $\mathcal{T}_{te}^L$ for different variants to analyse the 3D patch embedding module that is used in the proposed model $V\text{-}FE_{hyb}$. All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold font.

Appendix C: Class-specific F^1-scores achieved in the experiments related to the temporal weighting module

Variant	*Stl*	*Sld*	*Agr*	*Veg*	*Dfor*	*CFor*	*Wat*	*Bar*
V-FE$_{hyb}$	**88.0** ± 0.1	**60.8** ± 0.4	**88.7** ± 0.0	**78.5** ± 0.3	**83.6** ± 0.3	**92.7** ± 0.1	**95.6** ± 0.1	41.5 ± 0.7
V-FE$_{noTW}$	87.6 ± 0.1	60.7 ± 0.4	88.5 ± 0.2	78.4 ± 0.1	83.3 ± 0.2	92.4 ± 0.3	**95.6** ± 0.0	**41.6** ± 1.9

Table A3: F^1-scores in % for all classes achieved on $\mathcal{T}_{te}^{L}$ by the two model variants differing by the temporal weighting module used in the proposed model *V-FE$_{hyb}$*. All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold font.

Appendix D: Class-specific F^1-scores achieved in the comparison to other baseline methods

Variant	*Stl*	*Sld*	*Agr*	*Veg*	*Dfor*	*CFor*	*Wat*	*Bar*
V-FE$_{hyb}$	**88.0** ± 0.1	60.8 ± 0.4	88.7 ± 0.0	**78.5** ± 0.3	**83.6** ± 0.3	**92.7** ± 0.1	**95.6** ± 0.1	41.5 ± 0.7
FCN	87.6 ± 0.3	**61.5** ± 0.8	88.5 ± 0.2	77.6 ± 0.1	82.3 ± 0.2	91.9 ± 0.2	**95.6** ± 0.3	40.3 ± 1.9
Utilise	87.8 ± 0.4	60.6 ± 2.1	**88.8** ± 0.1	78.4 ± 0.5	**83.6** ± 0.4	92.6 ± 0.3	96.0 ± 0.2	**43.4** ± 0.9

Table A4: F^1-scores for all classes in % on the Lower Saxony dataset $\mathcal{T}_{te}^{L}$ achieved by variant *V-FE$_{hyb}$* in comparison to the two baseline models *FCN* (Voelsen et al., 2023) and *Utilise* (Stucker et al., 2023). All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold font.

Variant	*mF^1*	*Stl*	*Sld*	*Agr*	*Veg*	*Dfor*	*CFor*	*Wat*	*Bar*
V-FE$_{hyb}$	**64.8** ± 0.8	59.2 ± 0.6	**18.4** ± 0.5	86.1 ± 1.7	**69.6** ± 2.3	73.7 ± 1.8	81.2 ± 0.8	88.7 ± 0.3	**41.3** ± 0.6
FCN	63.6 ± 0.4	**59.7** ± 0.7	17.7 ± 1.1	84.7 ± 0.6	64.6 ± 0.6	**75.2** ± 2.3	80.5 ± 1.2	88.0 ± 0.1	39.7 ± 0.6
Utilise	64.2 ± 0.1	58.9 ± 0.0	16.8 ± 0.2	**6.2** ± 0.8	66.7 ± 1.7	74.3 ± 0.2	**81.5** ± 0.7	**89.4** ± 0.1	39.5 ± 0.5

Table A5: F^1-scores for all classes in % on the Kleve dataset $\mathcal{T}_{kleve}^{L}$ achieved by variant *V-FE$_{hyb}$* in comparison to the two baseline models *FCN* (Voelsen et al., 2023) and *Utilise* (Stucker et al., 2023). All quality metrics are averages over three experiments; the numbers behind the accuracy scores indicate the corresponding standard deviations. Best accuracy scores are indicated in bold font.

Bibliography

Aleissaee, A. A., Kumar, A., Anwer, R. M., Khan, S., Cholakkal, H., Xia, G.-S. and Khan, F. S., 2023. Transformers in remote sensing: A survey. *Remote Sensing.* 15(7), Paper 75.

An, X., He, W., Zou, J., Yang, G. and Zhang, H., 2024. Pretrain a remote sensing foundation model by promoting intra-instance similarity. *IEEE Transactions on Geoscience and Remote Sensing* 62, pp. 1–15.

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV), 2008. ATKIS®-Objektartenkatalog für das Digitale Basis-Landschaftsmodell 6.0. Available online (accessed 11/04/2025): `http://www.adv-online.de/GeoInfoDok/GeoInfoDok-6.0/Dokumente/`.

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV), 2024. Erläuterungen zum Anwendungsschema Landbedeckung 1.4 Available online (accessed 11/04/2025): `https://www.adv-online.de/GeoInfoDok/Aktuelle-Anwendungsschemata/Landbedeckung-1.0.1/`.

Arnab, A., Dehghani, M., Heigold, G., Sun, C., Lučić, M. and Schmid, C., 2021. Vivit: A video vision transformer. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 6836–6846.

Ayala, C., Aranda, C. and Galar, M., 2021. Towards fine-grained road maps extraction using Sentinel-2 imagery. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Science* V-3-2021, pp. 9–14.

Ba, J. L., Kiros, J. R. and Hinton, G. E., 2016. Layer normalization.

Badrinarayanan, V., Kendall, A. and Cipolla, R., 2017. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39(12), pp. 2481–2495.

Bergamasco, L., Bovolo, F. and Bruzzone, L., 2023. A dual-branch deep learning architecture for multisensor and multitemporal remote sensing semantic segmentation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 16, pp. 2147–2162.

Bishop, C. M., 2006. *Pattern recognition and machine learning.* 1st edn, Springer, New York (NY), USA.

Breiman, L., 2001. Random forests. *Machine learning* 45(1), pp. 5–32.

Caye Daudt, R., Le Saux, B., Boulch, A. and Gousseau, Y., 2019. Multitask learning for large-scale semantic change detection. *Computer Vision and Image Understanding.* 187, Paper nr. 102783.

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F. and Adam, H., 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In: *European Conference on Computer Vision (ECCV)*, p. 833–851.

Cong, Y., Khanna, S., Meng, C., Liu, P., Rozi, E., He, Y., Burke, M., Lobell, D. and Ermon, S., 2022. SatMAE: Pre-training transformers for temporal and multi-spectral satellite imagery. In: *Advances in Neural Information Processing Systems*, Vol. 35, pp. 197–211.

Cortes, C. and Vapnik, V., 1995. Support-vector networks. *Machine Learning* 20, pp. 273–297.

Dau, H. A., Keogh, E., Kamgar, K., Yeh, C.-C. M., Zhu, Y., Gharghabi, S., Ratanamahatana, C. A., Yanping, Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G. and Hexagon-ML, 2018. The UCR time series classification archive. available online (accessed 11/04/2025): `https://www.cs.ucr.edu/~eamonn/time_series_data_2018/`.

Di Mauro, N., Vergari, A., Basile, T. M. A., Ventola, F. G., Esposito, F. et al., 2017. End-to-end learning of deep spatio-temporal representations for satellite image time series classification. In: *Proceedings of the ECML/PKDD Discovery Challenges*. Skopje, Macedonia, 2017.

Ding, H., Xia, B., Liu, W., Zhang, Z., Zhang, J., Wang, X. and Xu, S., 2024. A novel mamba architecture with a semantic transformer for efficient real-time remote sensing semantic segmentation. *Remote Sensing*. 16(14), Paper 2620.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. and Houlsby, N., 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In: *International Conference on Learning Representations (ICLR)*.

Drusch, M., Del Bello, U., Carlier, S., Colin, O., Fernandez, V., Gascon, F., Hoersch, B., Isola, C., Laberinti, P., Martimort, P., Meygret, A., Spoto, F., Sy, O., Marchese, F. and Bargellini, P., 2012. Sentinel-2: ESA's optical high-resolution mission for GMES operational services. *Remote Sensing of Environment* 120, pp. 25–36.

Du, C., Lin, C., Jin, R., Chai, B., Yao, Y. and Su, S., 2024. Exploring the state-of-the-art in multi-object tracking: A comprehensive survey, evaluation, challenges, and future directions. *Multimedia Tools and Applications* 83(29), pp. 73151–73189.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L. and Muller, P.-A., 2019. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery* 33, pp. 917 – 963.

Fernandez-Beltran, R., Baidar, T., Kang, J. and Pla, F., 2021. Rice-yield prediction with multi-temporal Sentinel-2 data and 3D CNN: A case study in Nepal. *Remote Sensing*. 13(7), Paper 1391.

Gao, L., Liu, H., Yang, M., Chen, L., Wan, Y., Xiao, Z. and Qian, Y., 2021. STransFuse: Fusing Swin transformer and convolutional neural network for remote sensing image semantic segmentation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 14, pp. 10990–11003.

Gao, M., Zheng, F., Yu, J. J., Shan, C., Ding, G. and Han, J., 2023. Deep learning for video object segmentation: a review. *Artificial Intelligence Review* 56(1), pp. 457–531.

Garioud, A., Gonthier, N., Landrieu, L., De Wit, A., Valette, M., Poupée, M., Giordano, S. and Wattrelos, b., 2023. Flair : A country-scale land cover semantic segmentation dataset from multi-source optical imagery. In: *Advances in Neural Information Processing Systems*, Vol. 36, pp. 16456–16482.

Garnot, V. S. F. and Landrieu, L., 2020. Lightweight temporal self-attention for classifying satellite images time series. In: *Advanced Analytics and Learning on Temporal Data*, Vol. 12588, Springer Nature, pp. 171–181.

Garnot, V. S. F. and Landrieu, L., 2021. Panoptic segmentation of satellite image time series with convolutional temporal attention networks. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 4872–4881.

Garnot, V. S. F., Landrieu, L., Giordano, S. and Chehata, N., 2020. Satellite image time series classification with pixel-set encoders and temporal self-attention. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12325–12334.

Geng, Y. and Luo, X., 2019. Cost-sensitive convolutional neural networks for imbalanced time series classification. *Intelligent Data Analysis* 23(2), pp. 357–370.

Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep Learning*. 1st edn, MIT Press Cambridge. Available online (accessed 11/04/2025): `http://www.deeplearningbook.org`.

Hanyu, T., Yamazaki, K., Tran, M., McCann, R. A., Liao, H., Rainwater, C., Adkins, M., Cothren, J. and Le, N., 2024. AerialFormer: Multi-resolution transformer for aerial image segmentation. *Remote Sensing*. 16(16), Paper 2930.

He, K., Zhang, X., Ren, S. and Sun, J., 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 1026–1034.

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.

He, X., Zhou, Y., Zhao, J., Zhang, Di, Yao, R. and Xue, Y., 2022. Swin transformer embedding UNet for remote sensing image semantic segmentation. *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–15.

Heidarianbaei, M., Kanyamahanga, H. and Dorozynski, M., 2024. Temporal ViT-U-Net tandem model: Enhancing multi-sensor land cover classification through transformer-based utilization of satellite image time series. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Science* X-3-2024, pp. 169–177.

Hendrycks, D. and Gimpel, K., 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Hochreiter, S. and Schmidhuber, J., 1997. Long short-term memory. *Neural Computation* 9(8), pp. 1735–1780.

Hüsken, M. and Stagge, P., 2003. Recurrent neural networks for time series classification. *Neurocomputing* 50, pp. 223–235.

Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariant shift. In: *International Conference on Machine Learning (ICML)*, Vol. 37, pp. 448–456.

Ismail Fawaz, H., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., Webb, G. I., Idoumghar, L., Muller, P.-A. and Petitjean, F., 2020. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery* 34(6), pp. 1936–1962.

Ji, S., Zhang, C., Xu, A., Shi, Y. and Duan, Y., 2018. 3D convolutional neural networks for crop classification with multi-temporal remote sensing images. *Remote Sensing*. 10(1), Paper 75.

Jiao, L., Huang, Z., Lu, X., Liu, X., Yang, Y., Zhao, J., Zhang, J., Hou, B., Yang, S., Liu, F., Ma, W., Li, L., Zhang, X., Chen, P., Feng, Z., Tang, X., Guo, Y., Quan, D., Wang, S., Li, W., Bai, J., Li, Y., Shang, R. and Feng, J., 2023. Brain-inspired remote sensing foundation models and open problems: A comprehensive survey. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 16, pp. 10084–10120.

Kaiser, P., Wegner, J. D., Lucchi, A., Jaggi, M., Hofmann, T. and Schindler, K., 2017. Learning aerial image segmentation from online maps. *IEEE Transactions on Geoscience and Remote Sensing* 55(11), pp. 6054–6068.

Karim, F., Majumdar, S., Darabi, H. and Harford, S., 2019. Multivariate LSTM-FCNs for time series classification. *Neural Networks* 116, pp. 237–245.

Kingma, D. P. and Ba, J., 2015. Adam: A method for stochastic optimization. *International Conference on Learning Representations (ICLR)*.

Krizhevsky, A., Sutskever, I. and Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*, Vol. 25, pp. 1097–1105.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. and Jackel, L. D., 1989. Backpropagation applied to handwritten zip code recognition. *Neural Computation* 1(4), pp. 541–551.

Li, K., Zhao, W., Peng, R. and Ye, T., 2022a. Multi-branch self-learning vision transformer (MSViT) for crop type mapping with optical-SAR time-series. *Computers and Electronics in Agriculture*. 203, Paper nr. 107497.

Li, Y., Mao, H., Girshick, R. and He, K., 2022b. Exploring plain vision transformer backbones for object detection. In: *European Conference on Computer Vision (ECCV)*, pp. 280–296.

Lillesand, T., Kiefer, R. W. and Chipman, J., 2015. *Remote sensing and image interpretation*. 3rd edn, John Wiley & Sons, New York.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S. and Guo, B., 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 10012–10022.

Lobo Torres, D., Queiroz Feitosa, R., Nigri Happ, P., Elena Cué La Rosa, L., Marcato Junior, J., Martins, J., Olã Bressan, P., Gonçalves, W. N. and Liesenberg, V., 2020. Applying fully convolutional architectures for semantic segmentation of a single tree species in urban environment on high resolution UAV optical imagery. *Sensors*. 20(2), Paper 563.

Long, J., Shelhamer, E. and Darrell, T., 2015. Fully convolutional networks for semantic segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431 – 3440.

Maas, A. E., Rasti, B. and Ulfarsson, M. O., 2018. Label noise robust classification of hyperspectral data. In: *2018 9th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pp. 1–5.

Maas, A. L., Hannun, A. Y., Ng, A. Y. et al., 2013. Rectifier nonlinearities improve neural network acoustic models. In: *International Conference on Machine Learning (ICML)*, Vol. 30(1), pp. 3–8.

Maggiori, E., Tarabalka, Y., Charpiat, G. and Alliez, P., 2017. High-resolution aerial image labeling with convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing* 55(12), pp. 7092–7103.

Miller, L., Pelletier, C. and Webb, G. I., 2024. Deep learning for satellite image time-series analysis: A review. *IEEE Geoscience and Remote Sensing Magazine* 12(3), pp. 81–124.

Mohammadi Foumani, N., Miller, L., Tan, C. W., Webb, G. I., Forestier, G. and Salehi, M., 2024. Deep learning for time series classification and extrinsic regression: A current survey. *ACM Computing Surveys* 56(9), pp. 1–45.

Muhammad, K., Hussain, T., Ullah, H., Del Ser, J., Rezaei, M., Kumar, N., Hijji, M., Bellavista, P. and de Albuquerque, V. H. C., 2022. Vision-based semantic segmentation in scene understanding for autonomous driving: Recent achievements, challenges, and outlooks. *IEEE Transactions on Intelligent Transportation Systems* 23(12), pp. 22694–22715.

Mutegeki, R. and Han, D. S., 2020. A CNN-LSTM approach to human activity recognition. In: *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, pp. 362–366.

Nair, V. and Hinton, G. E., 2010. Rectified linear units improve restricted boltzmann machines. In: *International Conference on Machine Learning (ICML)*, Vol. 27, pp. 807–814.

Noh, H., Hong, S. and Han, B., 2015. Learning deconvolution network for semantic segmentation. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 1520–1528.

Otto, P., Fusta Moro, A., Rodeschini, J., Shaboviq, Q., Ignaccolo, R., Golini, N., Cameletti, M., Maranzano, P., Finazzi, F. and Fassò, A., 2024. Spatiotemporal modelling of $PM_{2.5}$ concentrations in Lombardy (Italy): a comparative study. *Environmental and Ecological Statistics* 31(2), pp. 245–272.

Panboonyuen, T., Jitkajornwanich, K., Lawawirojwong, S., Srestasathiern, P. and Vateekul, P., 2021. Transformer-based decoder designs for semantic segmentation on remotely sensed images. *Remote Sensing.* 13(24), Paper 5100.

Pelletier, C., Webb, G. I. and Petitjean, F., 2019. Temporal convolutional neural network for the classification of satellite image time series. *Remote Sensing.* 11(5), Paper 523.

Ronneberger, O., Fischer, P. and Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention (MICCAI) Conference, part III 18* pp. 234–241.

Rumelhart, D. E., Hinton, G. E. and Williams, R. J., 1985. Learning internal representations by error propagation. *Technical report, Institute for Cognitive Science, University of California, San Diego.*

Rußwurm, M. and Körner, M., 2020. Self-attention for raw optical satellite time series classification. *ISPRS Journal of Photogrammetry and Remote Sensing* 169, pp. 421–435.

Rußwurm, M., Pelletier, C., Zollner, M., Lefèvre, S. and Körner, M., 2020. Breizhcrops: A time series dataset for crop type mapping. In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science*, Vol. XLIII-B2-2020, pp. 1545–1551.

Sandmann, S., Hochgürtel, G., Piroska, R. and Steffens, C., 2022. Cop4ALL NRW–Ableitung der Landbedeckung in Nordrhein-Westfalen mit Fernerkundung und künstlicher Intelligenz. *ZfV-Zeitschrift für Geodäsie, Geoinformation und Landmanagement* 5/2022, pp. 299–310.

Song, H., Kim, M., Park, D., Shin, Y. and Lee, J.-G., 2023. Learning from noisy labels with deep neural networks: A survey. *IEEE Transactions on Neural Networks and Learning Systems* 34(11), pp. 8135–8153.

Song, H., Rajan, D., Thiagarajan, J. and Spanias, A., 2018. Attend and diagnose: Clinical time series analysis using attention models. *AAAI Conference on Artificial Intelligence* 32(1), pp. 4091–4098.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I. and Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(56), pp. 1929–1958.

Strudel, R., Garcia, R., Laptev, I. and Schmid, C., 2021. Segmenter: Transformer for semantic segmentation. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 7262–7272.

Stucker, C., Garnot, V. S. F. and Schindler, K., 2023. U-tilise: A sequence-to-sequence model for cloud removal in optical satellite time series. *IEEE Transactions on Geoscience and Remote Sensing* 61, pp. 1–16.

Tarasiou, M., Chavez, E. and Zafeiriou, S., 2023. ViTs for SITS: Vision transformers for satellite image time series. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10418–10428.

Teimouri, M., Mokhtarzade, M., Baghdadi, N. and Heipke, C., 2022. Fusion of time-series optical and SAR images using 3D convolutional neural networks for crop classification. *Geocarto International* 37(27), pp. 15143–15160.

Toker, A., Kondmann, L., Weber, M., Eisenberger, M., Camero, A., Hu, J., Hoderlein, A. P., Şenaras, C., Davis, T., Cremers, D., Marchisio, G., Zhu, X. X. and Leal-Taixé, L., 2022. DynamicEarthNet: Daily multi-spectral satellite dataset for semantic change segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21158–21167.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. and Polosukhin, I., 2017. Attention is all you need. In: *Advances in Neural Information Processing Systems*, Vol. 30, pp. 5998–6008.

Voelsen, M., Lauble, S., Rottensteiner, F. and Heipke, C., 2023. Transformer models for multi-temporal land cover classification using remote sensing images. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Science*, Vol. V-3-2022, pp. 271–279.

Voelsen, M., Teimouri, M., Rottensteiner, F. and Heipke, C., 2022. Investigating 2D and 3D convolutions for multitemporal land cover classification using remote sensing images. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Science*, Vol. V-3-2022, pp. 271–279.

Voelsen, M., Rottensteiner, F. and Heipke, C., 2024. Transformer models for land cover classification with satellite image time series. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science* 92, pp. 547 – 568.

Wang, L., Fang, S., Meng, X. and Li, R., 2022a. Building extraction with vision transformer. *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–11.

Wang, L., Li, R., Duan, C., Zhang, C., Meng, X. and Fang, S., 2022b. A novel transformer based semantic segmentation scheme for fine-resolution remote sensing images. *IEEE Geoscience and Remote Sensing Letters* 19, pp. 1–5.

Wang, W., Xie, E., Li, X., Fan, D.-P., Song, K., Liang, D., Lu, T., Luo, P. and Shao, L., 2021. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: *IEEE International Conference on Computer Vision (ICCV)*, pp. 568–578.

Wang, Y., Braham, N. A. A., Xiong, Z., Liu, C., Albrecht, C. M. and Zhu, X. X., 2023. SSL4EO-S12: A large-scale multi-modal, multi-temporal dataset for self-supervised learning in earth observation. *IEEE Geoscience and Remote Sensing Magazine* 11(3), pp. 98–106.

Wang, Z., Yan, W. and Oates, T., 2017. Time series classification from scratch with deep neural networks: A strong baseline. In: *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 1578–1585.

Wittich, D. and Rottensteiner, F., 2021. Appearance based deep domain adaptation for the classification of aerial images. *ISPRS Journal of Photogrammetry and Remote Sensing* 180, pp. 82–102.

Xiao, T., Liu, Y., Zhou, B., Jiang, Y. and Sun, J., 2018. Unified perceptual parsing for scene understanding. In: *European Conference on Computer Vision (ECCV)*, pp. 418–434.

Xiao, X., Guo, W., Chen, R., Hui, Y., Wang, J. and Zhao, H., 2022. A swin transformer-based encoding booster integrated in u-shaped network for building extraction. *Remote Sensing.* 14(11), Paper 2611.

Xu, Z., Zhang, W., Zhang, T., Yang, Z. and Li, J., 2021. Efficient transformer for remote sensing image segmentation. *Remote Sensing.* 13(18), Paper 3585.

Yan, J., Liu, J., Wang, L., Liang, D., Cao, Q., Zhang, W. and Peng, J., 2022. Land-cover classification with time-series remote sensing images by complete extraction of multiscale timing dependence. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 15, pp. 1953–1967.

Yang, C., Rottensteiner, F. and Heipke, C., 2021. CNN-based multi-scale hierarchical land use classification for the verification of geospatial databases. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science* XLIII-B2-2021, pp. 495–502.

Yao, M., Zhang, Y., Liu, G. and Pang, D., 2024. SSNet: A novel transformer and CNN hybrid network for remote sensing semantic segmentation. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 17, pp. 3023–3037.

Yuan, Y., Lin, L., Liu, Q., Hang, R. and Zhou, Z.-G., 2022. SITS-former: A pre-trained spatio-spectral-temporal representation model for Sentinel-2 time series classification. *International Journal of Applied Earth Observation and Geoinformation.* 106, Paper 102651.

Zaheer, M., Guruganesh, G., Dubey, K. A., Ainslie, J., Alberti, C., Ontanon, S., Pham, P., Ravula, A., Wang, Q., Yang, L. and Ahmed, A., 2020. Big Bird: Transformers for longer sequences. In: *Advances in Neural Information Processing Systems*, Vol. 33, pp. 17283–17297.

Zhang, C., Jiang, W., Zhang, Y., Wang, W., Zhao, Q. and Wang, C., 2022a. Transformer and CNN hybrid deep neural network for semantic segmentation of very-high-resolution remote sensing imagery. *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–20.

Zhang, C., Wang, L., Cheng, S. and Li, Y., 2022b. SwinSUNet: Pure transformer network for remote sensing image change detection. *IEEE Transactions on Geoscience and Remote Sensing* 60, pp. 1–13.

Zhang, W., Zhang, H., Zhao, Z., Tang, P. and Zhang, Z., 2023a. Attention to both global and local features: A novel temporal encoder for satellite image time series classification. *Remote Sensing.* 15(3), Paper 618.

Zhang, Y. and Yan, J., 2023. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In: *International Conference on Learning Representations (ICLR)*, Vol. 11.

Zhang, Z., Liu, Q. and Wang, Y., 2018. Road extraction by deep residual U-Net. *IEEE Geoscience and Remote Sensing Letters* 15(5), pp. 749–753.

Zhao, B., Lu, H., Chen, S., Liu, J. and Wu, D., 2017a. Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics* 28(1), pp. 162–169.

Zhao, B., Xing, H., Wang, X., Song, F. and Xiao, Z., 2023a. Rethinking attention mechanism in time series classification. *Information Sciences* 627, pp. 97–114.

Zhao, H., Shi, J., Qi, X., Wang, X. and Jia, J., 2017b. Pyramid scene parsing network. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2881–2890.

Zhao, Y., Ban, Y. and Sullivan, J., 2023b. Tokenized time-series in satellite image segmentation with transformer network for active fire detection. *IEEE Transactions on Geoscience and Remote Sensing* 61, pp. 1–13.

Zheng, Y., Liu, Q., Chen, E., Ge, Y. and Zhao, J. L., 2014. Time series classification using multi-channels deep convolutional neural networks. In: *International conference on web-age information management*, Springer International Publishing Switzerland, pp. 298–310.

Curriculum Vitae

Personal Information

Name	Mirjana Voelsen
Date of birth	14.02.1994 in Langenhagen, Germany

Work Experience

since June 2019	Leibniz University Hannover (Germany) Institute of Photogrammetry and GeoInformation *Research Scientist*

Education

2016 - 2019	Leibniz University Hannover (Germany) Studies of Geodesy and Geoinformatics *Master of Science*
2013 - 2016	Leibniz University Hannover (Germany) Studies of Geodesy and Geoinformatics *Bachelor of Science*
2012 - 2013	Voluntary social year Kinderladen-Initiative Hannover e.V.
2000 - 2012	High School *A-levels*

Wissenschaftliche Arbeiten der Fachrichtung Geodäsie und Geoinformatik der Leibniz Universität Hannover

(Eine vollständige Liste der Wiss. Arb. ist beim Geodätischen Institut, Nienburger Str. 1, 30167 Hannover erhältlich.)

Nr. 391	KNABE, Annike:	New Concepts for Gravity Field Recovery using Satellites (Diss. 2023)
Nr. 392	KALIA, Andre:	Landslide activity detection based on nationwide Sentinel-1 PSI datasets (Diss. 2023)
Nr. 393	BROCKMEYER, Marco:	Modellierung von Bodenbewegungen anhand heterogener Messverfahren am Beispiel der niedersächsischen Landesfläche (Diss. 2023)
Nr. 394	ZHANG, Mingyue:	Characteristics and Benefits of Differential Lunar Laser Ranging (Diss. 2023)
Nr. 395	DENNIG, Dirk:	Entwicklung eines kinematischen Profilvermessungssystems am Beispiel Kranbahnvermessung (Diss. 2024)
Nr. 396	FUEST, Stefan:	Nudging travelers to societally favorable routes by means of cartographic symbolization (Diss. 2024)
Nr. 397	MOFTIZADEH, Rozhin:	Advanced Particle Filtering for Vehicle Navigation based on Collaborative Information (Diss. 2024)
Nr. 398	VASSILEVA, Magdalena Stefanova:	Satellite Radar Interferometry for Geohazards: from ground deformation to processes understanding (Diss. 2024)
Nr. 399	MALINOVSKAYA, Anna:	Statistical Process Monitoring of Networks (Diss. 2024)
Nr. 400	BANNERT, Jörn:	Der Einfluss von Straßenverkehrslärm und Umgehungsstraßen auf Grundstückswerte in Ortslagen - Bestimmung mittels Expertenbefragung nach der Delphi-Methode (Diss. 2024)
Nr. 401	AXMANN: Jeldrik:	Maximum consensus localization using LiDAR (Diss. 2024)
Nr. 402	TENNSTEDT, Benjamin:	Concept and Evaluation of a Hybridization Scheme for Atom Interferometers and Inertial Measurement Units (Diss. 2024)
Nr. 403	HAKE, Frederic:	Schadenserkennung an Bauwerken mittels maschinellem Lernens (Diss. 2025)
Nr. 404	KARIMIDOONA, Ali:	On Integrity Prediction for Network-RTK Positioning in Urban Environments (Diss. 2025)
Nr. 405	ORTEGA, Mabel:	Domain Adaptation for Deforestation Detection in Remote Sensing: Addressing Class Imbalance and Performance Estimation (Diss. 2025)
Nr. 406	KUPRIYANOV, Alexey:	Investigation of Optical Accelerometry and Novel Satellite Formations for Future Gravimetry Missions (Diss. 2025)
Nr. 407	BREVA, Yannick:	On the Observation Quality of Robot-based GNSS Antenna Calibration for Determining Codephase Corrections (Diss. 2025)
Nr. 408	GUO, Zelong:	Co- and Post-seismic Slip Models Inferred from InSAR Geodesy (Diss. 2025)
Nr. 409:	YUAN, Yunshuang:	Collective Perception: A Fully-Sparse Deep Learning Framework for Multi-Agent Data Fusion (Diss. 2025)
Nr. 410	SU, Jingyao:	Towards interval-based autonomous integrity monitoring: Error bounding and uncertainty propagation (Diss. 2025)
Nr. 411	VINCENT, Asha:	Clock Networks for Geodetic Applications (Diss. 2025)
Nr. 412	KAMALASANAN, Vinu:	Control of Walking Behavior in Shared Spaces using Augmented Reality (Diss. 2025)
Nr. 413	TRUSHEIM, Philipp:	Kooperative Positionierung mittels Bündelausgleichung mit dynamischen Objekten (Diss. 2025)
Nr. 414	KRÖGER, Johannes:	Estimation and Validation of multi-GNSS multi-Frequency Phase Center Corrections (Diss. 2025)
Nr. 415	RUWISCH, Fabian:	GNSS Feature Maps – Robust Lane-level Accurate GNSS Navigation In Urban Trenches (Diss. 2025)
Nr. 416	WANG, Wandi:	Multi-Sensor Remote Sensing Time Series Analysis of Landslide Processes (Diss. 2025)
Nr. 417	ZOU, Qianqian:	3D mapping with probabilistic uncertainty measures (Diss. 2025)
Nr. 418	VOELSEN, Mirjana:	Combining Convolutions and Attention for Land Cover Classification of Satellite Image Time Series (Diss. 2025)