# TRANSFORMER MODELS FOR MULTI-TEMPORAL LAND COVER CLASSIFICATION USING REMOTE SENSING IMAGES

M. Voelsen*, S. Lauble, F. Rottensteiner, C. Heipke

Institute of Photogrammetry and GeoInformation, Leibniz University Hannover, Germany
voelsen@ipi.uni-hannover.de

**Commission II, WG II/4**

**KEY WORDS:** land cover classification, remote sensing, Swin Transformer, FCN, multi-temporal images

**ABSTRACT:**

The pixel-wise classification of land cover, i.e. the task of identifying the physical material of the Earth's surface in an image, is one of the basic applications of satellite image time series (SITS) processing. With the availability of large amounts of SITS it is possible to use supervised deep learning techniques such as Transformer models to analyse the Earth's surface at global scale and with high spatial and temporal resolution. While most approaches for land cover classification focus on the generation of a mono-temporal output map, we extend established deep learning models to multi-temporal input and output: using images acquired at different epochs we generate one output map for each input timestep. This has the advantage that the temporal change of land cover can be monitored. In addition, features conflicting over time are not averaged. We extend the Swin Transformer for SITS and introduce a new spatio-temporal transformer block (ST-TB) that extracts spatial and temporal features. We combine the ST-TB with the swin transformer block (STB) that is used in parallel for the individual input timesteps to extract spatial features. Furthermore, we investigate the usage of a temporal position encoding and different patch sizes. The latter is used to merge neighbouring pixels in the input embedding. Using SITS from Sentinel-2, the classification of land cover is improved by +1.8% in the mean F1-Score when using the ST-TB in the first stage of the Swin Transformer compared to a Swin Transformer without the ST-TB layer and by +1,6% compared to fully convolutional approaches. This demonstrates the advantage of the introduced ST-TB layer for the classification of SITS.

## 1. INTRODUCTION

Pixel-wise classification, referred to as semantic segmentation in Computer Vision, is the task of assigning a class label to each pixel in an image. For land cover classification, a common application for remote sensing images, these classes correspond to different physical materials on the Earth's surface like *Water* or *Forest*. New satellite missions, such as Sentinel-2 from the Copernicus program of the European Union, provide satellite image time series (SITS) with high spatial and temporal resolution at global scale. These SITS enable the use of supervised deep learning methods such as Fully Convolutional Neural Networks (FCNs) or Transformer models, which are able to achieve excellent results on big datasets.

The main goal of this paper is the extension of deep learning models for multi-temporal land cover (LC) classification based on SITS. We will refer to the produced pixel-wise classification outputs as *maps* in the remainder of this paper. Whereas existing work mostly focuses on the generation of a mono-temporal output map even if multi-temporal input images are employed, our goal is the generation of a LC map for each input image timestep. This has the advantage that the class of a pixel can change over time, opening the possibility to model LC change, and that conflicting input features are not averaged. For training we use labels from a topographic database that is updated every three months. This inherently leads to some label noise, i.e. errors in the reference labels, as it takes some time until a change of LC is included into this database. On the other hand, this strategy provides a large amount of labelled data with respect to both, the area covered and the number of timesteps.

* Corresponding author

There are two promising deep learning models that have already been successfully adapted for semantic segmentation of remote sensing images: FCNs and Transformer networks. FCNs are a standard model for vision tasks which are used in remote sensing, e.g. for change detection, crop and LC classification. Several approaches have adapted these models for SITS, e.g. by using parallel encoders for different timesteps (Caye Daudt et al., 2019) or by computing temporal features using convolutions (Pelletier et al., 2019). Transformer models, initially proposed for machine translation by Vaswani et al. (2017), have also been adopted for image classification (Chen et al., 2022; Dosovitskiy et al., 2021) and segmentation (Strudel et al., 2021), achieving state-of-the-art performance. Transformers are based on attention modules considering all input tokens (e.g. words) of a sequence, and consider global dependencies directly. When it comes to semantic segmentation of images, the computational complexity of these attention modules increases quadratically with the image size, which results in a trade-off between computation time and classification performance. Therefore, most approaches merge several pixels into a patch to decrease the number of input tokens at the cost of losing spatial resolution, e.g. (Dosovitskiy et al., 2021). Approaches such as the Swin Transformer (Liu et al., 2021) mitigate this effect by computing the attentions only in local windows, resulting in a drastically reduced computational complexity.

Several approaches combine a Transformer backbone with a FCN decoder to obtain pixel-wise predictions and achieve promising results with mono-temporal remote sensing images, e.g. (Gao et al., 2021; Zhang et al., 2022). We extend these models for multi-temporal LC classification and investigate different adaptations of our baseline model that is composed of

a Swin Transformer backbone and a FCN decoder with multi-temporal output. For the multi-temporal Swin encoder we combine the original Swin transformer block, which is run in parallel for all timesteps, with a new spatio-temporal transformer block (ST-TB) to extract spatio-temporal features between the input patches of all timesteps. In our experiments we investigate the effectiveness of this ST-TB layer, the usage of a temporal position encoding based on the day-of-year of the image acquisition and the influence of the patch size that is used to reduce the spatial resolution. We show that it is possible to train such a model from scratch without enormous GPU resources if enough varying training data with sufficient variability is available, and we compare the performance of our Transformer-FCN model with a purely convolutional model that is based on a U-Net architecture. In this context, we also investigate different variants of the U-Net structure, e.g. parallel encoders for the different timesteps, to ensure a fair comparison of all methods. Our scientific contribution can be summarized as follows:

- Adaptation of transformer-based and convolution-based deep neural networks for generating multi-temporal output maps in LC classification,
- Investigation of different variants of the transformer-based model, including temporal position embedding, parallel blocks for the timesteps in the encoder and the used patch size for merging neighbouring pixels,
- Comparison with a FCN solution, also including adaptations such as parallel encoder blocks for the images of different timesteps.

## 2. RELATED WORK

In this section, first, we discuss related work that exploits fully convolutional neural networks for pixel-wise classification (semantic segmentation) with a focus on remote sensing images. Afterwards, we introduce transformer models and the way in which they are adapted for semantic segmentation before discussing several approaches that use transformer adaptations for remote sensing images with a focus on SITS.

FCNs (Long et al., 2015) have been used in various remote sensing applications e.g. for the classification of LC (Pelletier et al., 2019; Voelsen et al., 2022), change detection (Caye Daudt et al., 2019), or agricultural crop classification (Ji et al., 2018). The main component of a FCN is the convolution, in which a kernel with learnable weights is shifted across the input to extract features, thus integrating local context in the feature computation. By combining layers with convolutions and downsampling, more context is integrated and the computed features become more complex. While in most applications these features are computed in the spatial dimensions (Ronneberger et al., 2015; Caye Daudt et al., 2019), they can also be determined in other ones, e.g. across different time steps (Pelletier et al., 2019), spectral bands, or combinations thereof (Ji et al., 2018). Most FCN variants are based on an encoder-decoder structure, e.g. U-Net (Ronneberger et al., 2015), which also uses skip connections to combine feature maps from the encoder and the decoder in order to preserve fine spatial structures.

In contrast to convolutional layers, transformer models include global context directly by using self-attention layers. The transformer model, introduced for machine translation (Vaswani et al., 2017), is used for many vision tasks today. The self-attention mechanism is computed between all tokens (parts of the input, e.g. words) of the input sequence, and therefore the features are independent from the distance in the input sequence. Since their success in natural language processing, transformer models have been adapted to the field of computer vision for tasks such as image classification (Dosovitskiy et al., 2021), object detection (Chen et al., 2022) or semantic segmentation (Strudel et al., 2021). Dosovitskiy et al. (2021) introduced the Vision Transformer (ViT), which directly uses the transformer model from Vaswani et al. (2017) for image classification. In this work, the input image is divided into patches of a fixed size that are flattened and mapped to a latent vector of constant size using a linear projection before they are provided to the transformer. Strudel et al. (2021) adapt the Vision Transformer for semantic segmentation by adding a decoder after the ViT. The output embeddings for the individual patches serve as input to this decoder, which predicts patch-level class scores; pixel-wise predictions are obtained by upsampling. The performance of this model directly depends on the patch size, as the results show that an increasing patch size results in a coarser representation of the image but also in a faster training process as the computational complexity decreases. This drawback is solved by the Swin (**s**hifted **win**dow) Transformer (Liu et al., 2021). Here, the attentions are computed in a local window that combines a fixed number of patches (e.g. $7 \times 7$), which drastically decreases the computational complexity. To include global context, these windows are shifted between subsequent layers to allow an information flow between them. Furthermore, Swin computes hierarchical representations by gradually merging neighbouring patches in deeper layers, similar to downsampling steps in a FCN. Thanks to these adaptations the Swin transformer can be used as a backbone for different vision tasks, including semantic segmentation.

Several works combine the Swin-backbone with a FCN-decoder for semantic segmentation of remote sensing images (Zhang et al., 2022; Gao et al., 2021; He et al., 2022) and achieve promising results. Different modifications can further improve the results. For instance, He et al. (2022) use a feature compression module based on convolutions instead of merging neighbouring patches for downsampling. Zhang et al. (2022) add a boundary detection head as an additional output to further improve the results at class boundaries. Wang et al. (2022) use a convolutional path for detailed structures in parallel to a Transformer path that extracts global features for building detection and achieve better results than with pure transformer or convolutional approaches. Tarasiou et al. (2023) use an adaptations of the Vision Transformer for crop classification based on SITS. After computing attentions between all timesteps of the same patch, they reshape the outputs and the attentions are computed between all patches of the same timestep. This model is shown to achieve better accuracies than other state-of-the-art techniques. Garnot and Landrieu (2021) do not use the Swin transformer directly, instead they integrate temporal attention into the skip connection modules of their multitemporal U-Net adaptation for panoptic segmentation of crop parcels.

Very few approaches focus on the generation of multi-temporal output maps with SITS. Zhu et al. (2021) extend a U-Net model with LSTM layers in the decoder and are able to predict output maps for each input timestep. The hybrid Conv-LSTM approach outperforms purely convolutional or recurrent models in their experiments. Relatively close to our approach is the one from (Yuan et al., 2022): The authors propose the so called

*SITS-former*, which serves as a pre-trained model for Sentinel-2 time series classification. Training is done in a self-supervised way and can be fine tuned afterwards for downstream tasks. In their model the authors first use 3D convolutions to extract spatio-spectral features in parallel for each timestep and use the results as patch embeddings for the transformer encoder. In contrast to our approach, the input patches have a spatial size of $5 \times 5$ pixels, which reduces the spatial context to a small local neighbourhood. Whereas the output of the pre-trained model is multi-temporal, for the test application of crop classification, the outputs for all epochs are combined by pooling in the temporal dimension, so that the downstream task only provides a monotemporal output. A remaining challenge is the availability of training datasets with multitemporal labels for the classification of SITS. There are several datasets generated for specific tasks, e.g. for building extraction (Van Etten et al., 2021; Caye Daudt et al., 2019) or crop classification (Rußwurm et al., 2020). For the task of LC classification, Toker et al. (2022) give an overview about some existing satellite datasets and conclude that most have single time labels or long revisiting times (e.g. yearly) for the label data. They propose a new dataset with monthly labels for LC classification at pixel level for a period of two years and 75 selected areas all over the world, which can serve as a new benchmark for LC change detection.

Transformers use a positional encoding to allow the model to make use of the order of the input sequence (Vaswani et al., 2017). This encoding can be fixed or include learnable parameters and is normally added to the input embeddings. Most approaches for image classification adapt this encoding, e.g. (Dosovitskiy et al., 2021; Liu et al., 2021; Strudel et al., 2021), and all of them agree that the usage of a positional encoding increases the performance, while the type of encoding is less critical. When it comes to SITS not only the spatial order of the patches needs to be encoded, but also the temporal one. For this purpose, 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 used observation and integrate this in their temporal auto-encoder module to classify crop types with SITS. Similar, Tarasiou et al. (2023) use an acquisition-time-specific temporal encoding to also accommodate for irregular distributions of the images in time that is learned during training and improves the mean Intersection over Union (mIoU) metric by 2%. Yuan et al. (2022) use a fixed positional encoding vector that is assigned to the day of the year of the input image. In their experiments the performance slightly decreases when the temporal encoding is used. This motivates the investigation of a temporal encoding for our application, because the acquisition dates of the images may be at irregular intervals and may vary between different years.

To the best of our knowledge none of the existing approaches investigates the use of transformer-based models for multi-temporal semantic segmentation for LC classification. There are several approaches that combine the Transformer models with FCN, especially the Swin-Transformer with a FCN decoder, and achieve promising results with mono-temporal remote sensing images. We extend this architecture to provide multi-temporal output, investigate different model adaptations and compare the results to purely convolutional architectures.

## 3. METHODOLOGY

In this section, we describe the models for multi-temporal classification that are compared in our experiments. In all cases, the input consists of a time series of co-registered remote sensing images. For each of the $T$ timesteps, an image of size $B \times H \times W$ is given, with $H$, $W$ and $B$ indicating the image height, width and the number of spectral bands, respectively. The output consists of a LC map for each of the timesteps according to a pre-defined class structure. In section 3.1 we describe FCN variants used as baselines. Afterwards, the new transformer-based model is introduced (section 3.2).

### 3.1 Models based on FCN

The FCN used in this paper is an adaptation of U-Net (Ronneberger et al., 2015), which we already used in (Voelsen et al., 2022) for the comparison of mono- and multitemporal input timeseries. The main extension is the adaptation to a multi-temporal input and output. The encoder is composed of four convolutional blocks (CB), each consisting of two convolutional layers with a kernel size of $k = 3$, followed by batch normalization (Ioffe and Szegedy, 2015) and a rectified linear unit (ReLu) activation. To reduce the spatial dimension by a factor of two, a max-pooling layer is added at the end of the first three CBs, with a window size of 2 x 2 and stride 2. The number of output feature channels is set to a fixed size $D_{FCN}$ for the first encoder block and doubled every time the spatial resolution is reduced. The decoder consists of three upsampling layers using bilinear interpolation, each followed by another CB. Finally, a 1x1 convolution maps the feature vectors to raw class scores, which are normalized by a softmax layer. Between the encoder and decoder blocks with identical spatial resolution there are skip connections, which concatenate the corresponding feature maps before they are processed in the next decoder block.

For a part of the network, the images corresponding to the individual timesteps can be processed in parallel. These *parallel blocks* have shared weights and are used to extract spatial features first; their number is adaptable, e.g. parallel processing could just occur in the first CB. Subsequently, the feature maps of all timesteps are concatenated, increasing the number of input feature maps for the next convolution by a factor of $T$. To avoid losing spatial or temporal features in the first joined block, we do not change the number of feature maps in the first fused CB. For the later layers in the decoder the feature maps for all timesteps are separated again. In order to be able to use the skip connections from the encoder, this is done at the block having the resolution at which the feature maps were fused in the encoder; for instance, if parallel processing only occurred in the first encoder block, the feature maps would be split just before the last CB in the decoder. To do so, the number of output feature maps is increased by a factor of $T$ for the convolution before the separation into $T$ parallel decoder branches, which leads to a number of trainable parameters that increases with $T$ (cf. Section 4.3). The resultant feature maps are then separated to $T$ stacks of feature maps, which are concatenated with the corresponding feature maps from the end of the parallel encoder blocks (skip connections). Afterwards, the remaining CBs are executed in parallel, resulting in $T$ output maps of size $C \times H \times W$, with $C$ as the number of classes.

**3.1.1 FCN variants:** The parallel encoder and decoder blocks constrain the model to first extract spatial features before both, spatial and temporal features can be extracted. To investigate the effectiveness of the parallel blocks we fuse the timesteps at different stages of the model. In variant $FCN_{B2}$, the first two CBs in the encoder and, consequently, the last two CBs in the decoder are executed in parallel. This is the model

with the smallest number of blocks of joint feature extraction from all timesteps. In variant $FCN_{B1}$, the features are fused after the first CB and the separation is done before the last CB in the decoder. Finally, in variant $FCN_{B0}$, there is no parallel processing of the individual timesteps at all. The fusion is performed by stacking all spectral channels from all timesteps, resulting in a input image of size $(T \cdot B) \times H \times W$. The separation is done just before the final 1x1 convolution to obtain a classification map for each timestep.

### 3.2 Model based on Swin Transformer

For the hybrid Transformer-FCN model we combine the Swin Transformer (Liu et al., 2021) with the UPerNet (Xiao et al., 2018) as decoder and adapt this model to multi-temporal images. We start with a summary of the basic Swin encoder for mono-temporal images before describing our adaptations of that model for multi-temporal input and output images.

For the Swin backbone we use the original architecture of Liu et al. (2021). Like previous Transformer models it splits the input image into non-overlapping patches of size $P \times P$ that are flattened and linearly projected to vectors of dimension $D_{Swin}$. Several Swin-Transformer blocks (STB) are applied; the patch-based feature vectors serve as the input to the first STB. After applying several STB, a patch merging layer is applied to the feature maps to reduce the number of patches and produce a hierarchical representation, similar to convolutional backbones. In a patch merging layer the features from $2 \times 2$ patches are concatenated and a fully connected layer is applied to reduce the number of feature maps again. All STB that share the same number of patches are referred to as a *Stage* $i$ in combination with the preceding patch merging layer to obtain this number of patches. Patch merging is applied three times, resulting in a total number of four *Stages* ($i \in [1, ..., 4]$). In each Stage a total number of $L_i$ STB are applied consecutively, with $l_i \in [0, ..., L_i]$ as the $l$-th block in *Stage* $i$. Each time the number of patches is reduced, the number of feature maps is doubled, i.e. in *Stage* $i$ the number of layers is $D_i = D_{Swin} \cdot 2^{i-1}$, with $D_{Swin}$ being the number of feature maps in the first *Stage* ($D_1 = D_{Swin}$). For a more detailed description of this backbone model we refer to (Liu et al., 2021).

Each of the swin-transformer blocks consists of a window based multi-head self-attention (W-MSA) module, which is followed by a Multilayer Perceptron (MLP) with two layers, $d_{MLP} = 4 \cdot D_i$ dimensions and GELU non-linearity between them (one green rectangle in figure 1). Layer Normalization (LN) is applied before each W-MSA and MLP module, and a residual connection is applied after each module. In a W-MSA module the attentions are computed in local windows of size $M$, i.e. considering $M \times M$ patches, each of size $P \times P$. To connect patches of neighbouring windows in the attention computation, the windows are shifted by $\frac{M}{2}$, for the following block, resulting in the following computations for two consecutive STB (for simplicity we omit the *Stage* index $i$) (Liu et al., 2021):

$$\begin{aligned} \hat{z}^l &= W\text{-}MSA(LN(z^{l-1})) + z^{l-1} \\ z^l &= MLP(LN(\hat{z}^l)) + \hat{z}^l_i \\ \hat{z}^{l+1} &= SW\text{-}MSA(LN(z^l)) + z^l \\ z^{l+1} &= MLP(LN(\hat{z}^{l+1})) + \hat{z}^{l+1}. \end{aligned} \quad (1)$$

In equation 1, $\hat{z}^l$ refers to the output of the (S)W-MSA module and $z^l$ refers to the output of the MLP for block $l$. SW-MSA refers to a W-MSA block that is applied to a window partitioning that is shifted compared to the windows in block $l-1$. For more details of the shifted window approach we refer the reader to (Liu et al., 2021).

Similar to (Vaswani et al., 2017) the attentions in a W-MSA layer are computed in a number of parallel heads that differ for each *Stage* $i$ and are denoted by $h_i$. In each head the self-attention is computed based on equation 2:

$$Attention(Q, K, V) = SoftMax(QK^T/\sqrt{d} + R)V, \quad (2)$$

with $R \in \mathbb{R}^{M^2 \times M^2}$ as a relative position bias, $d = D_i/h_i$ as the query/key dimension and $Q, K, V \in \mathbb{R}^{M^2 \times d}$ as query, key and value matrices. $Q, K$ and $V$ originate from the input matrix $(LN(z^{l-1}))$ that is transformed by three linear transformations. The outputs of all heads are concatenated in the end to form the output $\hat{z}^l \in \mathbb{R}^{M^2 \times D_i}$.

Similar to Liu et al. (2021) we combine the Swin backbone with a FCN decoder to obtain per-pixel class labels. This decoder is based on UPerNet (Xiao et al., 2018), because in previous experiments we found this combination to slightly outperform the combination of Swin encoder with our FCN decoder introduced in section 3.1. Basically, UPerNet is a U-Net architecture with a Pyramid Pooling Module before the first decoder layer to extract features at different scales. Due to the dimensions of the Swin encoder the features maps with identical spatial resolutions can again be fused via skip-connections. We refer the reader to (Xiao et al., 2018) for more details on UPerNet.

**3.2.1 Swin-encoder for multi-temporal images:** The original Swin encoder is used for images of size $B \times H \times W$ and produces an output map of size $C \times H \times W$. We extend this to be able to process an input of size $T \times B \times H \times W$ and produce an output of size $T \times C \times H \times W$ and apply parallel feature extraction for all timesteps for a certain (configurable) numbers of *Stages* in the transformer. To do so we introduce spatio-temporal transformer blocks (ST-TB). After describing these ST-TB and the temporal encoding, we introduce different variants of our method that use the ST-TB module in different *Stages* of the model.

**Spatio-temporal transformer block:** We introduce a new spatio-temporal transformer block (ST-TB) that extends the STB to multi-temporal images and is used in combination with the normal STB that can be run in parallel for the individual timesteps, as shown in figure 1. Similar to STB, the ST-TB consists of an adapted window based multi-head self-attention block ($W\text{-}MSA_{Time}$), followed by a MLP and GELU non-linearity as described in section 3.2. In $W\text{-}MSA_{Time}$ the attentions are computed in local windows of size $M \times M \times T$, which means that $M \times M$ patches of the local neighbourhood for all timesteps are included. This also increases the computational complexity by a factor of $T$. Again, the attentions are computed in $h_i$ parallel heads, by computing the self-attention in each head using equation 3:

$$Attention_{Time}(Q, K, V) = SoftMax(QK^T/\sqrt{d})V, \quad (3)$$

with $Q, K, V \in \mathbb{R}^{TM^2 \times d}$ as query, key and value matrices and $d = D_i/h_i$ as the query and key dimension. As the ST-TB extracts spatio-temporal features, we use it in combination with $T$ STB that are executed in parallel to extract spatial features for each timestep $t \in [1, ..., T]$:

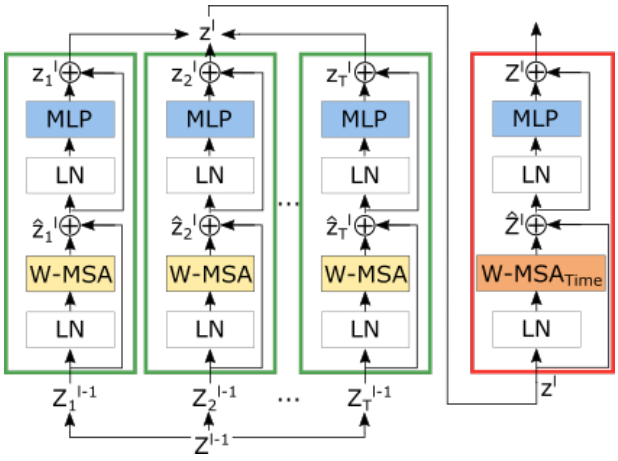



Figure 1. Parallel Swin transformer blocks (STB, green rectangles) for all timesteps followed by the spatio-temporal transformer block (ST-TB, red rectangle).

$$\begin{aligned}\hat{z}_t^l &= W\text{-}MSA(LN(Z_t^{l-1})) + Z_t^{l-1}\\ z_t^l &= MLP(LN(\hat{z}_t^l)) + \hat{z}_t^l\\ \hat{Z}^l &= W\text{-}MSA_{Time}(LN(z^l)) + z^l\\ Z^l &= MLP(LN(\hat{Z}^l)) + \hat{Z}^l. \end{aligned} \quad (4)$$

In equation 4, $\hat{z}_t^l$ and $z_t^l$ refer to the outputs of the $W\text{-}MSA$ and $MLP$ modules, respectively of the parallel STB $t$ in block $l$. Afterwards the outputs of all timesteps are fused to obtain $z^l$ with an additional dimension of size $T$, which serves as input to the ST-TB. The ST-TB consists of the $W\text{-}MSA_{Time}$ to obtain the intermediate output $\hat{Z}^l$ and the $MLP$ to obtain the final output $Z^l$. Figure 1 shows how the STB and ST-TB are combined to form one block $l$. In this way the parallel STB compute attentions only in the spatial dimension, whereas the ST-TB extracts spatio-temporal features. Note that in contrast to the STB computations in eq. 1 we use the same windows in $W\text{-}MSA_{Time}$ as in $W\text{-}MSA$ and shift these windows by $\frac{M}{2}$ for the next block $(l+1)$.

**Temporal position encoding:** Similar to (Garnot et al., 2020) we adapt the positional encoding from (Vaswani et al., 2017) to a temporal position encoding based on the acquisition date of the used satellite images:

$$te_{DOY,f} = sin\left(\frac{DOY}{10000^{\frac{2f}{D_{Swin}}}} + \frac{\pi}{2} mod(f,2)\right), \quad (5)$$

with $DOY \in [1,...,365]$ as the day of the year and $f \in [1,...,D_{Swin}]$ as the feature index. The temporal position encoding $TE(DOY) = [te_1,...,te_{D_{Swin}}]$ is computed for each input timestep and added to the input embedding just before the first STB is applied.

**3.2.2 Swin-FCN variants:** We want to investigate the effectiveness of the spatio-temporal transformer block (ST-TB). To do so, we experiment with different Swin-FCN models, that use the ST-TB in different $Stages$ of the model. In variant $Swin_{S2}$ the fusion of all timesteps is done after *Stage 2* and the parallel STB and combined ST-TB layers are used in *Stages 1* and *2*. In variant $Swin_{S1}$ the parallel STB and ST-TB are used in *Stage 1*. In variant $Swin_{S0}$ there is no parallel processing of the timesteps at all. Similar to $FCN_{B0}$ the fusion is done by stacking the images of all timesteps, resulting in a input image of size $T \cdot B \times H \times W$. To obtain a classification map for each timestep the feature maps are separated before the final $1 \times 1$ convolution in the UPer-Net. If not specified differently, all variants use the same number of blocks $L = [2, 2, 6, 2]$ and heads $h = [3, 6, 12, 24]$ for *Stages 1 - 4*, an input feature dimension of $D_{Swin} = 96$, a window size of $M = 7$ and a patch size of $P = 4$, which corresponds to SWIN-T in (Liu et al., 2021).

### 3.3 Training

During the training process, the parameters of the network are iteratively updated using the ADAM optimizer (Kingma and Ba, 2015), which minimizes a loss function that measures the discrepancy between the reference and the predictions of the network using the current parameters. To counteract any imbalance of the class distribution of the training samples, we minimize the weighted cross entropy loss, considering class weights based on the degree of difficulty of the current classifier to predict the class labels correctly (Wittich and Rottensteiner, 2021). The weighted cross-entropy loss $L_{CrEn}$ is based on the softmax predictions $y_n^c$ for a sample $n$ to belong to class $c$:

$$L_{CrEn} = -\frac{1}{N}\sum_n\sum_c C_n^c \cdot ln(y_n^c) \cdot cw_c. \quad (6)$$

In equation 6, $C_n^c = 1$ if the $n^{th}$ sample (i.e., the $n^{th}$ pixel in a minibatch) belongs to class $c$, otherwise $C_n^c = 0$. $N$ is the total number of pixels in the minibatch for which the loss is computed. The class weights $cw_c$ are set to 1 for all classes during the first epoch, which corresponds to using an unweighted loss. After the first training epoch, the last training minibatch is classified using the current network parameters and the result is used to compute the intersection over union ($IoU_c$) for every class $c$, which is then used to adjust the class weights:

$$IoU_c = \frac{TP_c}{TP_c + FP_c + FN_c}. \quad (7)$$

In equation 7, $TP_c$, $FP_c$ and $FN_c$ refer to the number of pixels that are true positives, false positives and false negatives, respectively, with respect to class $c$. As these results highly depend on the minibatch used for the calculation (it may even happen that a class is not present in that minibatch), we average the IoUs from the last 10 epochs (or from all available ones before epoch 11). Following (Wittich and Rottensteiner, 2021), these IoU scores are then used to determine the class weights $cw_c$ for the next epoch:

$$cw_c = (1 - \Delta IoU_c)^\kappa = [1 - (IoU_c - \frac{1}{N_c}\sum_{h=0}^{N_c} IoU_h)]^\kappa, \quad (8)$$

where $\Delta IoU_c$ is the difference between the mean $IoU$ of all classes and the $IoU$ of class $c$, $N_c$ denotes the number of classes, and the hyperparameter $\kappa$ is used to scale the influence of classes with a lower $IoU$ on the results. These class weights are used in the loss (equation 6) during the following epoch.

## 4. EXPERIMENTS

### 4.1 Dataset

Our test site covers the whole area of the German federal state of Lower Saxony (47600 $km^2$). The dataset comprises

Sentinel-2 images acquired between January 2019 and December 2022. We use Sentinel-2 Level-2A data, which contain georeferenced bottom-of-atmosphere reflectance and cloud masks from the top-of-atmosphere reflectance of every pixel (Bertini et al., 2012). We use the four spectral bands with a ground sampling distance (GSD) of 10 m (red, green, blue, near infrared). All bands are normalized to zero-mean and unit standard deviation by using $v'_{i,b} = (v_{i,b} - \mu_b)/\sigma_b$, where $v'_{i,b}$ and $v_{i,b}$ correspond to the corrected and the original grey value of pixel $i$ in band $b$ of an image, respectively, and $\mu_b$ and $\sigma_b$ denote the mean and standard deviation of band $b$, respectively; $\sigma_b$ and $\mu_b$ are computed based on a part of the dataset that covers the whole area and images acquired from 2019 and 2020. The provided cloud mask is used to exclude parts of the images that contain more than 5% cloud coverage. This results in a different number of available images for different regions, varying between 7 and 50 for a time period of one year.

To obtain the class labels to be used in training, information from the official German landscape model ATKIS is used (AdV, 2008). This database contains information about 113 different land use classes, which is too detailed for automatic classification. To define a suitable class structure for LC, several land use classes from the database are merged, so that in the end, nine classes are differentiated: *Settlement (stl.), Sealed area (sld.), Agriculture (agr.), Greenland (grl.), Forest (for.), Flowing water (fwt.), Standing water (swt.), Sea (sea)* and *Barren land (bar.)*. In addition, the class *others* is used for areas without label information that occur due to errors in the database or in areas outside the state borders. This information is used to disregard samples of this class in training and evaluation. The database is continuously updated, based on in-situ surveys and aerial flights that take place every three years for the same region. The updates are provided every three month (ends of March, June, September and December), resulting in four label maps per year. For the experiments in this paper, these reference label images are rasterized at the GSD of the satellite imagery, and each Sentinel-2 image is combined with the label image closest in time to its acquisition date. This procedure leads to some label noise, as some more recent changes visible in the images are not yet contained in the database.

For computational reasons, the available data are split into tiles of 8×8 $km^2$ (800 × 800 pixels, referred to as BE8 tiles in the following), which leads to a total number of 885 tiles covering Lower Saxony (cf. figure 2). For three tiles (shown in red in figure 2), the corresponding reference label image was corrected manually for different image acquisition dates, resulting in 13 corrected BE8 label images in total. This is done to obtain a reference for the evaluation that is not affected by label noise. In this process, about 18% of the pixels were changed, which gives an indication for the amount of label noise to be expected in the remaining data. Most of these changes occur between the classes *Greenland* and *Agriculture*.

To generate one multi-temporal input patch for training or inference we decided to use a total time period of one year (first image from January, last image from December) to be close to the vegetation cycle, as many approaches show improvements especially for classes containing vegetation or crops when multi-temporal data are used (Ji et al., 2018; Rußwurm and Körner, 2020). We split the year into $T$ time intervals, e.g. for $T = 4$ there are four intervals, each covering three months. During evaluation for each interval, the Sentinel-2 image acquired most closely in time to the middle of the current interval is selected. In this way the time periods between the used images are as similar as possible and testing is done on the same images for all experiments. For training, we found it to be beneficial to choose one Sentinel-2 image that is acquired in the current time interval randomly from all available images in that time period. In that way, even if the same area is chosen multiple times, the used images can vary, which increases the variability of the whole training dataset.

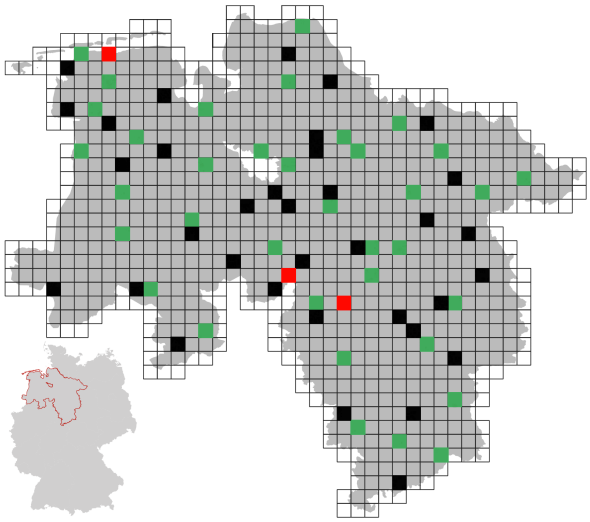


Figure 2. Overview of the available BE-8 tiles of 8×8 $km^2$ each. Grey / green: potential training / validation tiles. Red: test tiles with manually corrected reference (dataset $R_1$). Black: test tiles without corrected reference (dataset $R_2$).

### 4.2 Experimental protocol

**4.2.1 Experimental setup:** For all experiments, we split our dataset into a set of 810 BE-8 tiles for training, 36 BE8-tiles for validation (green tiles in figure 2) and 39 BE-8 tiles for testing (black and red tiles in figure 2). Training is based on the method described in section 3.3. To create the input patches, we randomly crop windows of $(H, W) = (256, 256)$ pixels from the available training tiles. We apply random data augmentation, including rotations by 90°, 180°, 270° and horizontal and vertical flipping, which results in a large variety of available training patches. Training is carried out in epochs, where one epoch consists of a series of iterations, each considering a small minibatch of input patches. The number of iterations per epoch is set so that in each epoch, 10.000 patches are used to update the parameters. Training continues for a maximum number of 100 epochs, but is stopped earlier if the validation accuracy does not increase for 10 epochs. The minibatch size is set to 4 and reduced to 2 for experiments with 12 timesteps. During training the ADAM optimizer (Kingma and Ba, 2015) is used with the parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate is set to 0.001 for the FCN approaches and to 6e-5 for the Swin transformer. These values were found to perform best on the validation dataset and are also those used by Liu et al. (2021) for the Swin transformer. For both architectures the learning rate is decreased by a factor of 0.7 every 10 epochs. The parameter $\kappa$ for weight computation is set to 1 for all experiments, as this value resulted in a good trade-off between the accuracies of the over- and underrepresented classes. For each experiment, three models are trained, each time starting from a different random initialization of the layers and using different random batches for training to assess the influence of these random components on the results.

**4.2.2 Evaluation protocol:** For evaluation, the classification results on the test tiles are compared to the reference. As these tiles are larger than the input size of the models, the evaluation is done with a sliding window approach using a horizontal and vertical shift of 128 pixels. This results in four predictions per pixels (except at the edges of a BE8 tile) and the resulting softmax scores for each class are averaged to obtain the final predictions. Quality indicators are determined based on a per-pixel comparison between the predicted labels and the reference. We report the overall accuracy (OA), i.e. the percentage of pixels with correctly predicted class labels, the F1-Scores per class and the mean F1-Score (mF1). The OA is somewhat biased by the imbalanced class distribution (see table 1) of the dataset. The F1-Score, which is the harmonic mean of precision and recall that is computed based on the predictions for each class separately, is not influenced by the class imbalance at all. Consequently, the impact on the mF1 is equal for all classes, as the number of pixels is not taken into account. These indicators are determined based on the three manually corrected test tiles (referred to as set $R_1$) as well as on the 39 non-corrected test tiles (referred to as $R_2$). $R_1$ is not affected by the errors in the reference (label noise), but small changes in the classification result have a larger impact on the evaluation metrics. $R_2$ forms a larger set of samples, but it is affected by label noise. The distribution of the class labels in the training and test datasets is shown in table 1.

| Set | Percentage of samples for each class [%] | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | *stl.* | *sld.* | *agr.* | *grl.* | *for.* | *fwt.* | *swt.* | *sea* | *bar.* |
| Tr. | 9.2 | 0.7 | 38.3 | 23.1 | 21.5 | 1.5 | 0.7 | 3.5 | 1.4 |
| $R_2$ | 9.5 | 0.7 | 45.1 | 22.3 | 19.1 | 0.3 | 0.8 | 1.2 | 1.0 |
| $R_1$ | 8.1 | 2.3 | 54.7 | 12.0 | 9.5 | 1.1 | 1.2 | 10.6 | 0.6 |

Table 1. Class label distribution for the training and test datasets.

**4.2.3 Test setup:** The evaluation is split into two main parts. In a first set of experiments, we investigate the FCN variants described in section 3.1. The main goal of these experiments is the investigation of the influence of the parallel convolutional blocks for the different timesteps, resulting in purely convolutional models that can be used for a comparison to the Swin-FCN model. These experiments are shown and discussed in section 4.3. In the second set of experiments (section 4.4), we evaluate the hybrid Swin-FCN architecture described in section 3.2. Again, we compare different variants of the architecture, but also the influence of the additional temporal encoding and a varying patch size. In the end we compare the most promising setups of both model variants and discuss the detection of LC changes based on visual inspection.

### 4.3 Evaluation of FCN-variants

To evaluate and compare the FCN variants introduced in section 3.1, we run experiments for the variants $FCN_{B0}$, $FCN_{B1}$ and $FCN_{B2}$, each for $T = 4$ and $T = 12$ timesteps, respectively. The number of filters $D_{FCN}$ used in the first convolutional block is set to 64. As shown in table 2, this results in a varying number of trainable parameters, depending on $T$. To investigate the influence of the number of trainable parameters, the experiment for variant $FCN_{B0}$ with $T = 4$ is also trained with $D_{FCN} = 128$. The results are summarized in table 2.

The results show that an earlier merging of all timesteps leads to an increase of the overall model performance. While most results for the mF1 on $R_1$ and $R_2$ with $T = 4$ are not statistically significant with a confidence level of 0.05, for $T = 12$ variant $FCN_{B0}$ is significant better than the others. These results indicate that the extraction of temporal features in early layers with high spatial resolution is more important than the temporal features extracted in layers with coarser spatial resolution.

The results for variant $FCN_{B0}$ with $D_{FCN} = 64$ and $D_{FCN} = 128$ do not differ significantly. For the variant with $D_{FCN} = 64$ the results on $R1$ are slightly better, while the opposite is true for $R2$, even if the model with $D_{FCN} = 128$ has four times as many parameters. These results are consistent with observations from previous experiments: for the FCN model, a higher number of trainable parameters does not result in an increase of the performance. Whereas the F1-Scores for most classes are relatively stable for all variants, this is not the case for the different classes of water (*Standing Water*, *Flowing Water* and *Sea*). A possible explanation for this finding are the tides, which lead to shapes similar to rivers or lakes when the tide is out and can just be classified correctly if more timesteps are combined. A larger number of input timesteps improves the results on $R_2$ by almost 2% in mF1 (variant $FCN_{B0}$) and slightly on $R_1$; this improvement occurs consistently for the F1-Scores of all classes.

In summary, the parallel extraction of spatial features does not lead to a better performance of the FCN models, which is consistent for different numbers of timesteps. For the comparison to the Swin-FCN variants we use variant $FCN_{B0}$ with $D_{FCN}$ = 128 for $T = 4$ and $D_{FCN} = 64$ for $T = 12$.

### 4.4 Evaluation of the Swin Transformer variants

Similar to the experiments with the FCN-variants we investigate the influence of the spatio-temporal transformer block (ST-TB) in combination with the Swin transformer block (STB) that separates the images of different timesteps in *Stage 1* ($Swin_{S1}$) or *Stages 1* and *2* ($Swin_{S2}$). In addition we investigate the effects of the temporal position encoding as introduced in section 3.2.1 and a smaller patch size of $P = 2$ using variant $Swin_{S1}$, because this variant turned out to be the most promising one. We conduct the experiments for $T = 4$ and repeat the experiments with $T = 12$ for the variants with the best performances. The results are summarized in table 3.

It can be observed that the performance on the corrected dataset $R_1$ is quite similar for all variants, both in terms of mF1 and OA. This is different for $R_2$: On this dataset $Swin_{S1}$ significantly outperforms $Swin_{S0}$ (+1.8%) and $Swin_{S2}$ (+1.6%) considering the mF1 for $T = 4$ and a significance level of 0.05. Again, the results for the water classes improve most (e.g. +5.9% in mF1 for *Flowing Water* compared to $Swin_{S0}$), but also those for *Sealead area* (+2.6%) or *Barren land* (+2.3%) increase. These results show the advantage of the parallel STB for all timesteps combined with the ST-TB to extract spatio-temporal features. In our approach the second layer of parallel STB already takes joint spatio-temporal features as input, which could be the reason why adding another layer of parallel STB in *Stage 2* does not increase the performance anymore. The temporal position encoding has no significant effect on the performance, as it leads to a slight decrease in performance for $T = 4$ and to a slight increase for $T = 12$. A reason for this might be the consistent definition (per variant) of the time periods in which the images are sampled. Our assumption is that the temporal position is more important if the acquisition times differ much more within the same experiment.

The best results on $R_2$ are obtained with a reduced patch size of $P = 2$ ($T = 4$). In this case the mF1 increases by 1.6% and

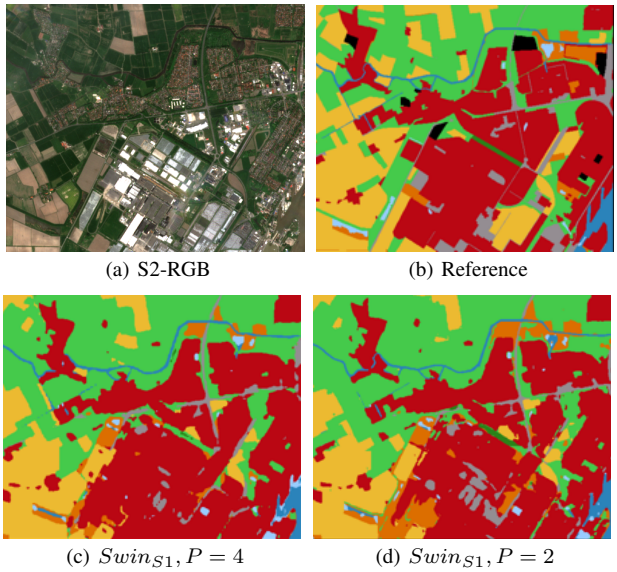
| variant | $D_{FCN}$ | $T$ | $f$ | F1-scores on $R_2$ [%] | | | | | | | | | $R_2$ [%] | | $R_1$ [%] | | $\#p$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | *stl.* | *sld.* | *agr.* | *grl.* | *for.* | *fwt.* | *swt.* | *sea* | *bar.* | mF1 | OA | mF1 | OA | |
| $FCN_{B0}$ | 64 | 4 | 0 | 86.5 | 41.7 | 90.7 | 76.2 | 94.2 | 69.4 | **82.3** | 95.7 | 34.8 | 74.6 ± 0.0 | 86.9 ± 0.1 | **70.6 ± 0.6** | **81.4 ± 0.3** | 4M |
| $FCN_{B0}$ | 128 | 4 | 0 | 86.7 | 43.0 | 90.5 | **76.6** | **94.3** | **72.2** | 80.4 | 94.3 | 35.9 | **74.9 ± 0.2** | **87.0 ± 0.2** | 70.3 ± 1.3 | 80.9 ± 1.0 | 16M |
| $FCN_{B1}$ | 64 | 4 | 1 | **86.9** | 43.9 | **90.8** | 75.7 | 92.6 | 54.7 | 77.4 | 86.8 | 34.3 | 71.5 ±3.2 | 86.1 ±0.6 | 68.7 ±1.3 | 80.5 ±0.5 | 10M |
| $FCN_{B2}$ | 64 | 4 | 2 | 86.5 | **44.4** | 90.7 | **76.6** | 93.9 | 64.6 | 77.0 | **96.5** | **37.8** | 74.2 ±0.9 | 86.8 ±0.1 | 68.0 ±0.2 | 80.3 ± 0.4 | 17M |
| $FCN_{B0}$ | 64 | 12 | 0 | **86.9** | 42.5 | **91.0** | **77.2** | **94.5** | **75.4** | **85.9** | **98.5** | **36.2** | **76.5 ±0.1** | **87.5 ±0.1** | **70.8 ±0.5** | **81.7 ±0.4** | 9M |
| $FCN_{B1}$ | 64 | 12 | 1 | **86.9** | 43.8 | 90.5 | 74.7 | 91.6 | 71.2 | 74.8 | 93.0 | 35.4 | 73.5 ±1.6 | 85.8 ±0.0 | 69.3 ±0.1 | 81.2 ±0.4 | 31M |
| $FCN_{B2}$ | 64 | 12 | 2 | 86.2 | **44.3** | **91.0** | 76.0 | 92.7 | 61.1 | 84.0 | 84.8 | 35.6 | 72.9 ±5.3 | 86.3 ±0.5 | 69.1 ±0.3 | 81.1 ±0.2 | 50M |

Table 2. Results for LC classification with U-Net architectures. $D_{FCN}$: number of filters in first convolutional block, $T$: number of used timesteps, $f$: conv. block after which feature maps are fused. $\#p$: Number of parameters. Best results for the same value of $T$ are indicated in bold.

| $T$ | $f$ | $TE$ | $P$ | F1-scores on $R_2$ [%] | | | | | | | | | $R_2$ [%] | | $R_1$ [%] | | $\#p$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | *stl.* | *sld.* | *agr.* | *grl.* | *for.* | *fwt.* | *swt.* | *sea* | *bar.* | mF1 | OA | mF1 | OA | |
| 4 | 0 | - | 4 | 85.0 | 38.4 | 90.4 | 75.7 | 94.0 | 65.7 | 83.0 | 96.9 | 31.3 | 73.4 ±0.8 | 86.6 ±0.0 | 70.1 ±0.3 | 81.6 ±0.1 | 87M |
| 4 | 1 | - | 4 | 85.6 | 41.0 | 90.6 | 76.3 | 94.3 | 71.6 | 84.7 | **98.9** | 33.6 | 75.2 ±0.0 | 87.0 ±0.1 | 70.0 ±0.4 | **81.7 ±0.2** | 60M |
| 4 | 2 | - | 4 | 85.1 | 40.2 | 90.5 | 76.5 | 94.1 | 63.4 | 84.2 | 96.0 | 32.5 | 73.6 ±0.5 | 86.8 ±0.1 | 69.4 ±0.5 | 81.4 ±0.1 | 60M |
| 4 | 1 | yes | 4 | 85.5 | 41.8 | 90.7 | 76.7 | 94.3 | 67.4 | 84.6 | 97.1 | 34.4 | 74.7 ±0.7 | 87.0 ±0.1 | 69.6 ±0.3 | 81.5 ±0.1 | 60M |
| 4 | 1 | - | 2 | **87.1** | **45.6** | **91.1** | **77.9** | **94.9** | **72.1** | **86.5** | 97.4 | **36.2** | **76.5 ±1.1** | **87.8 ±0.0** | **70.3 ±0.2** | 81.6 ±0.1 | 60M |
| 12 | 1 | - | 4 | 85.7 | 40.4 | 90.9 | 77.5 | 94.4 | 71.2 | 84.2 | **99.4** | 33.8 | 75.3 ±0.9 | 87.4 ±0.0 | 69.2 ±0.2 | 81.5 ±0.0 | 60M |
| 12 | 1 | yes | 4 | **86.3** | **42.1** | **91.1** | **78.1** | **94.5** | **73.9** | **85.1** | 99.3 | **34.9** | **76.1 ±0.1** | **87.7 ±0.2** | **69.6 ±0.4** | **81.9 ±0.1** | 60M |

Table 3. Results for LC classification with Swin Transformer. $T$: number of used timesteps, $f$: *Stage* after which the timesteps are fused, $TE$: use of temporal encoding, $P$: patch size. Best results for the same value of $T$ are indicated in bold.

the OA by 0.8% compared to variant $Swin_{S1}$ with $P = 4$. The F1-Scores of all classes (except *Sea*) are also best for this experiment. These results are confirmed by visual inspection. In the example shown in figure 3, many objects have clearer outlines, e.g. the *Sealed area* inside the *Settlement* area, the river passing through the smaller settlement area on the left or the water area in the bottom right corner. These results clearly show the advantage of a smaller patch size and confirm the results achieved by Swin Transformer in other applications. However, reducing $P = 4$ to $P = 2$ increases the computational complexity and thus training time by a factor of four. A comparison of the FCN-variants with the swin-variants does not result in a clear conclusion. Especially on the corrected dataset $R_1$ the results do not differ significantly. On $R_2$, $Swin_{S1}$ with $P = 2$ outperforms $FCN_{B0}$ for $T = 4$ by 1.6% in mF1 and 0.8% in OA.

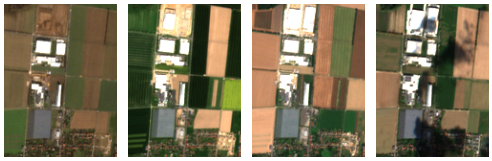


(a) S2-RGB (b) Reference

(c) $Swin_{S1}, P = 4$ (d) $Swin_{S1}, P = 2$

Figure 3. Exemplary prediction results on a tile corresponding to $R_2$ for variant $Swin_{S1}$ for $P = 4$ and $P = 2$. Colours: red - *bld.*, grey - *sld.*, yellow - *agr.*, light green - *grl.*, dark green - *for.*, dark blue - *fwt.*, light blue - *swt*, turquoise - *sea*, brown - *bar.*

All Swin-variants were trained from scratch and early stopping usually ended training between epochs 60 and 80. This is a big advantage in contrast to the original Transformer models, which were almost impossible to use without pre-trained models or immense GPU resources and large datasets (Steiner et al., 2022; Dosovitskiy et al., 2021).



(a) 20210223 (b) 20210509 (c) 20210812 (d) 20221006

(e) Ref. (f) 20210223 (g) 20210509 (h) 20210812 (i) 20221006

Figure 4. Exemplary prediction results on a tile corresponding to $R_2$ for variant $Swin_{S1}$ with $P = 2$. (a) - (d) RGB composites of the S-2 images corresponding to the predictions (f) - (i). (e) shows the reference Note that the predictions (f) - (h) correspond to the same input time series in 2021 while (i) is an example for one of the predictions from the time series in 2022. Colours: cf. figure 3

**Visual analysis of the multi-temporal output:** The motivation for generating multi-temporal output maps is the hope to detect LC changes very early by predicting these changes when they appear in the satellite image. This includes the detection of changes before they are updated to the database. An example is shown in figure 3, where areas with wrong labels of the class *agr.* in the bottom part of the figure are correctly classified as *stl.* in both results. Another example is shown in figure 4. While figures 4(f) to 4(h) correspond to the same input time series in 2021, figure 4(i) is an example of an output map from the time

series in 2022. The results for timesteps from the same input all look quite similar, even if there is a LC change, e.g. the new building that is constructed in 2021 (figures 4(a) - 4(c)). In this case the corresponding training labels are affected by label noise (fig 4(e)) and the model is not able to predict the change that occurs during the input time period. For the temporally following time series for the next year (2022, see an example map in figure 4(i)) the new building is detected correctly. This observation leads to the conclusion that some adjustments of the classifier are still necessary in order to detect changes already within the input batch in which the change occurs.

## 5. CONCLUSION

In this paper, we investigated different extensions of Transformer models for LC classification with satellite image time series by adapting the Swin Transformer model from (Liu et al., 2021) to deal with multi-temporal input and output. We introduced a new spatio-temporal transformer block (ST-TB) for the extraction of spatio-temporal features which we use in combination with the Swin transformer block (STB) to extract spatial features for all timesteps in parallel, which outperforms the Swin variant without the ST-TB when it is used in the first *Stage* of the model. The comparison to the purely convolutional models show quite similar results on the corrected test dataset and only a small increase in performance on the test dataset that was not corrected manually for the Swin transformer. The largest impact on the model performance has the reduction of the input patch size to $P = 2$, which is consistent with observations for other Swin- or Vision Transformer approaches.

Future research will investigate the modelling and prediction of land cover changes within the time series that serves as input to the model. In the obtained results, all predictions of the same input time series were quite similar even if the actual land cover was changing, which means that the exact date of a change is not predicted correctly. One main reason for this may be the lack of correct labels (often called label noise in literature), especially for the exact date of a change. Future research will focus on this challenge by investigating approaches to mitigate the effect of label noise and also by evaluating the models performance on a cleaner dataset with multi-temporal training data, e.g. from (Toker et al., 2022). Another aspect is the possibility to use input time series with varying length, which could help to exploit the full temporal resolution of the data. Lastly, as the comparison between FCN and Swin was on a quite similar level, future research will also investigate these two and other hybrid variants regarding classification performance, runtime in training and inference, and memory consumption.

## ACKNOWLEDGEMENTS

We thank the German Land Survey Office of Lower Saxony (Landesamt für Geoinformation und Landesvermessung Niedersachsen - LGLN) for providing the data of the geospatial database and for their support of this project.
We thank NVIDIA Corporation for providing GPU resources to this project.

## REFERENCES

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV), 2008. ATKIS®-Objektartenkatalog für das Digitale Basis-Landschaftsmodell 6.0. Available online (accessed 03/07/2023): http://www.adv-online.de/GeoInfoDok/GeoInfoDok-6.0/Dokumente/.

Bertini, F., Brand, O., Carlier, S., Del Bello, U., Drusch, M., Duca, R., Fernandez, V., Ferrario, C., Ferreira, M., Isola, C., Kirschner, V., Laberinti, P., Lambert, M., Mandorlo, G., Marcos, P., Martimort, P., Moon, S., Oldeman, P., Palomba, M., Pineiro, J., 2012. Sentinel-2 ESA's optical high-resolution mission for GMES operational services. *ESA bulletin. Bulletin ASE. European Space Agency*, SP-1322.

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

Chen, Z., Duan, Y., Wang, W., He, J., Lu, T., Dai, J., Qiao, Y., 2022. Vision transformer adapter for dense predictions. *arXiv preprint arXiv:2205.08534*.

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

Gao, L., Liu, H., Yang, M., Chen, L., Wan, Y., Xiao, Z., 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, 10990–11003.

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

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

He, X., Zhou, Y., Zhao, J., Zhang, Di, Yao, R., Xue, Y., 2022. Swin transformer embedding UNet for remote sensing image semantic segmentation. *IEEE Transactions on Geoscience and Remote Sensing*, 60, Paper 4408715.

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

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

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

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B., 2021. Swin transformer: Hierarchical vision transformer using shifted windows. *IEEE International Conference on Computer Vision (ICCV)*, 9992–10002.

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

Pelletier, C., Webb, G. I., 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., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 234–241.

Rußwurm, M., Körner, M., 2020. Self-attention for raw optical Satellite Time Series Classification. *ISPRS Journal of Photogrammetry and Remote Sensing*, 169, 421–435.

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

Steiner, A., Kolesnikov, A., Zhai, X., Wightman, R., Uszkoreit, J., Beyer, L., 2022. How to train your ViT? Data, augmentation, and regularization in vision transformers. *Transactions on Machine Learning Research*.

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

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

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., Leal-Taixé, L., 2022. Dynamicearthnet: Daily multi-spectral satellite dataset for semantic change segmentation. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 21158–21167.

Van Etten, A., Hogan, D., Manso, J. M., Shermeyer, J., Weir, N., Lewis, R., 2021. The multi-temporal urban development spacenet dataset. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6398–6407.

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

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

Wang, L., Fang, S., Meng, X., Li, R., 2022. Building Extraction With Vision Transformer. *IEEE Transactions on Geoscience and Remote Sensing*, 60, Paper 5625711.

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

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

Yuan, Y., Lin, L., Liu, Q., Hang, R., 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.

Zhang, C., Jiang, W., Zhang, Y., Wang, W., Zhao, Q., Wang, C., 2022. 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, Paper 4408820, 1–20.

Zhu, Y., Geiß, C., So, E., Jin, Y., 2021. Multitemporal relearning with convolutional LSTM models for land use classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14, 3251-3265.