INVESTIGATING 2D AND 3D CONVOLUTIONS FOR MULTITEMPORAL LAND COVER CLASSIFICATION USING REMOTE SENSING IMAGES

M. Voelsen[1]*, M. Teimouri[1,2], F. Rottensteiner[1], C. Heipke[1]

[1] Institute of Photogrammetry and GeoInformation, Leibniz Universität Hannover, Germany
(voelsen, teimouri, rottensteiner, heipke)@ipi.uni-hannover.de
[2] Department of Photogrammetry and Remote Sensing, K. N. Toosi University of Technology, Tehran, Iran
mteimouri@mail.kntu.ac.ir

Commission III, WG III/7

KEY WORDS: land cover classification, remote sensing, FCN, multi-temporal images, 3D-CNN

ABSTRACT:

With the availability of large amounts of satellite image time series (SITS), the identification of different materials of the Earth's surface is possible with a high temporal resolution. One of the basic tasks is the pixel-wise classification of land cover, i.e. the task of identifying the physical material of the Earth's surface in an image. Fully convolutional neural networks (FCN) are successfully used for this task. In this paper, we investigate different FCN variants, using different methods for the computation of spatial, spectral, and temporal features. We investigate the impact of 3D convolutions in the spatial-temporal as well as in the spatial-spectral dimensions in comparison to 2D convolutions in the spatial dimensions only. Additionally, we introduce a new method to generate multi-temporal input patches by using time intervals instead of fixed acquisition dates. We then choose the image that is closest in time to the middle of the corresponding time interval, which makes our approach more flexible with respect to the requirements for the acquisition of new data. Using these multi-temporal input patches, generated from Sentinel-2 images, we improve the classification of land cover by 4% in the mean F1-score and 1.3% in the overall accuracy compared to a classification using mono-temporal input patches. Furthermore, the usage of 3D convolutions instead of 2D convolutions improves the classification performance by a small amount of 0.4% in the mean F1-score and 1.2% in the overall accuracy.

1. INTRODUCTION

Pixel-wise classification (also known as semantic segmentation) is the task of assigning a class label to each pixel in an image. In remote sensing (RS), one common task is the classification of land cover, in which these classes correspond to different physical materials on the Earth's surface, e.g. *Water* or *Forest*. The current state-of-the-art methods to solve this task in a supervised manner are based on deep learning (DL), especially on fully convolutional neural networks (FCNs) (Long et al., 2015) such as U-Net (Ronneberger et al., 2015), which are special variants of convolutional neural networks (CNN).

New satellite constellations such as Sentinel-2, provided by the European Space Agency (ESA), acquire images with high temporal resolution at global scale. Such satellite image time series (SITS) enable a DL method to learn the spatial, spectral, and temporal development of each land cover class and thus further increase the classification accuracy (Pelletier et al., 2019; Ji et al., 2018). There are different possibilities to integrate the temporal and the spectral component into DL methods, for instance using recurrent neural networks (RNNs) (Ho Tong Minh et al., 2018; Mou et al., 2019) or convolutions in a CNN (Ge et al., 2020; Ji et al., 2018). The former is designed to explicitly model the temporal dependencies of an input sequence, e.g. time series data, and can provide one prediction for each part of the sequence. The latter shifts a kernel with learnable weights in defined dimensions to compute features that involve information from the local neighbourhood; such methods usually provide one output. In this work we will refer to a CNN in which the kernels are shifted in d dimensions as a d-dimensional CNN. A common 2D-CNN applies two-dimensional convolutions to extract spatial features. This can be extended, e.g. to 3D convolutions to extract spatial-spectral information. Three-dimensional CNNs are successfully used in video (Ji et al., 2013) and in crop classification (Ji et al., 2018). However, the performance of 3D convolutions in the spatial-spectral or the spatial-temporal dimensions for the application of land cover classification remains to be investigated.

Another common characteristic in the classification of SITS is that fixed acquisition dates are used during the training process. As a consequence, data acquired at those dates should ideally be available during inference. This requirement can be relaxed, e.g. by an additional interpolation step to fill missing time steps (Pelletier et al., 2019). In this paper, we introduce a more flexible way to generate multi-temporal input patches for a CNN. Instead of using individual acquisition dates we divide the year into several time intervals, similar to binning in the generation of a histogram, and use the image that is closest in time to the center of the corresponding interval to create multi-temporal patches. Using this type of multi-temporal input, we train different CNN architectures, including 2D-CNNs, 3D-CNNs, and a mixture of both. We apply the 3D convolutions in the spatial-temporal as well as in the spatial-spectral dimensions to investigate which dimensions are more suitable for land cover classification. As the baseline architecture we use a variant of U-Net (Ronneberger et al., 2015).

The scientific contribution of this paper can be summarized as follows:

- We investigate the impact of multi-temporal input data generated by choosing images from different time epochs

* Corresponding author

on the performance of a 2D-CNN for land cover classification compared to mono-temporal input data.

- We investigate the impact of 3D convolutions in the spatial-temporal as well as the spatial-spectral dimensions in comparison to 2D convolutions in the spatial dimensions in the application of land cover classification.
- We investigate the usage of 3D convolutions only in the first convolutional block to reduce the number of unknown network parameters and save training time and compare it to a CNN with 3D convolutions in all convolutional layers.

The dataset used for evaluation consists of Sentinel-2 images covering the entire German state of Lower Saxony (47600 km^2) for 2019 and 2020. The training labels are derived from a topographic database and differentiate nine land cover classes.

2. RELATED WORK

In recent years, deep learning methods have been used in various remote sensing applications, such as land cover classification (Kussul et al., 2017; Stoian et al., 2019), change detection (Ma et al., 2019; Mou et al., 2019), road extraction (Oehmcke et al., 2019), wildfire monitoring (Zhang et al., 2021a), and agricultural crop classification (Ji et al., 2018). One of the main advantages of these networks is their superior performance in comparison to more traditional methods using hand-crafted features. With the advent of satellite sensors and the availability of large amounts of SITS with a medium spatial and high temporal resolution, e.g. from Sentinel-2, deep neural networks (DNNs) can be used to extract spatial, spectral as well as temporal features automatically.

Convolutional neural networks (CNNs) are a type of DNNs that is successfully used for this task in remote sensing applications. These networks are capable of extracting spectral, spatial, spatial-spectral, and spatial-temporal features. For pixel-wise classification, fully convolutional neural networks (FCNs) are commonly used as a special type of CNN, consisting of an encoder-decoder structure in which the output resolution is identical to the one of the input. While the most common models for semantic segmentation, such as SegNet (Badrinarayanan et al., 2017) or U-Net (Ronneberger et al., 2015) use 2D kernels and shift them in the two spatial dimensions, it is also possible to use the kernel in the spectral or temporal dimension, e.g. (Pelletier et al., 2019), or in a combination of both, e.g. (Ji et al., 2018; Debella-Gilo and Gjertsen, 2021). Please refer to section 3.1 for a detailed definition of the convolutional operation. In the following, we focus on approaches to use the convolutional operation in different dimensions.

In order to extract spectral or temporal features, Li et al. (2017a) used a 1D-CNN to extract features from hyperspectral images. Although their method improved the classification accuracy, they only considered features in the temporal or spectral dimensions. Pelletier et al. (2019) proposed a CNN to classify SITS. They compared 1D convolutions in the spectral or temporal dimension, 2D convolutions in the temporal-spectral dimensions, and RNNs and concluded that the use of 2D convolutions outperforms the other approaches. Kussul et al. (2017) proposed a multi-level CNN, including a 1D-CNN with convolutions in the spectral dimension and a 2D-CNN with convolutions in the spatial dimensions, for classifying agriculture crops and land cover from optical and radar time series. Their results show that the 2D-CNN outperforms the other methods. In order to jointly extract features from more than two dimensions, Zhang et al. (2017) proposed a dual-channel CNN to extract spectral features with 1D convolutions in one branch and spatial features with 2D convolutions in the other one. Afterwards, the outputs of the two branches are combined to predict land cover or agriculture crops, achieving promising results. What is not covered in the introduced papers is a shift of the kernel in more than two dimensions. In this regard, Li et al. (2017b) proposed a 3D-CNN framework to extract deep spectral-spatial features from hyperspectral images. They showed that the performance of the 3D-CNN is better than the one of other methods, including a 2D-CNN with the convolution computed in the spatial dimensions. Similarly, Han et al. (2020) as well as Sellami et al. (2020) successfully used 3D-CNNs to extract spectral-spatial features from hyperspectral images. While Han et al. (2020) used a 3D-CNN with channel-wise attention for sea-ice detection, Sellami et al. (2020) proposed an unsupervised method for spectral band clustering and used the 3D convolutions in each group to classify different crop types. Spatial-temporal features were extracted by Ji et al. (2018), who applied a 3D-CNN for crop classification using SITS. The authors concluded that the 3D-CNN is especially suitable for modelling the dynamics of crop growth. Fernandez-Beltran et al. (2021) showed that 3D-CNNs can also be used in regression tasks. In order to estimate rice yield, the authors used 3D convolutions to extract spatial-temporal features from large-scale SITS and additional soil and climate data and achieved better results than other traditional and DL approaches.

One challenge when using convolutions with kernels having more dimensions is an increase in the number of parameters and, consequently, training time. To mitigate this effect, several strategies have been proposed that use 3D convolutions only in a part of a network architecture. Ge et al. (2020) proposed a 2D-3D-CNN that uses multiple branches, each consisting of a combination of 3D and 2D convolutions. The authors applied it to hyperspectral images and achieved promising results. Zhang et al. (2020) used a 3D-2D-CNN to also extract spatial-spectral features from hyperspectral images. They showed that the proposed method can already learn the most discriminative features when only a small number of training samples is available. Oehmcke et al. (2019) only used 3D convolutions in the first convolutional block of a U-Net. They used the 3D convolutions to extract features from SITS that are partly covered with clouds and achieved slightly better results than a 2D-CNN model trained using single-epoch images without any clouds. All the mentioned approaches indicate that the use of additional dimensions in the convolutional operation improves the accuracy of a CNN. However, to the best of our knowledge none of the existing approaches investigates the usage of 3D convolutions in the spatial-temporal as well as in the spatial-spectral dimensions for the application of land cover classification. We analyse this question for multi-temporal remote sensing images by comparing two variants of 3D-CNNs and a commonly used 2D-CNN.

One common characteristic of methods for classifying crops or land cover from SITS is to use fixed acquisition dates during training, e.g. (Ji et al., 2018; Kussul et al., 2017). Consequently, ideally images from the same dates should be available for inference, too. However, this requirement can hardly be fulfilled, e.g. due to cloud coverage or different satellite revisit times. Several ways to mitigate this problem have been suggested. Hagolle et al. (2015) used cloud and shadow masks to

find images with missing information and filled these gaps, e.g. by linear interpolation from images acquired at neighbouring time steps (Pelletier et al., 2019). To avoid such additional preprocessing we propose a more flexible way to generate multi-temporal input patches for a CNN. Instead of using fixed acquisition dates, we divide the year into several time intervals and use the image that is closest in time to the center of the corresponding time interval to generate training data, resulting in equi-distant sampling in time.

3. METHODOLOGY

3.1 The Convolutional Operation in 2D and 3D

The convolutional operation is the main component of a CNN. It is a linear operation that can be represented by $y = w * x + b$. The output tensor y is computed based on the input tensor x, the kernel w, and a bias b, with w and b being the parameters that are determined in training.

We refer to a convolution as d-dimensional if w is shifted in d dimensions of the input x. This operation results in a d-dimensional output tensor y. In many CNN architectures the input x has at least one dimension more than d; for instance 2D-CNNs are often used for image data having three dimensions (two spatial, one spectral) and the filter matrix is shifted in the two spatial dimensions. In this work, we assume that in the case of a d-dimensional convolution, i.e. if the kernel is shifted in d dimensions, the input of a convolutional layer l has one additional dimension. We refer to the size in this additional dimension as the number of input channels P. For each input channel p a different kernel w_p is used to compute the convolution. This results in P d-dimensional outputs, which are then added up to obtain one d-dimensional output tensor y. Another parameter for a layer l is the number of output channels O, which indicates the number of times the whole process is repeated with different kernels w, resulting in O d-dimensional output tensors y, which are then stacked and provide the input channels for layer $l + 1$. In the following, we introduce the computation of 2D- and 3D-convolutions for one of the output channels O.

The convolution in 2D is the one most widely used in CNNs for image processing. Here, the kernel is shifted in the two spatial dimensions. Of course, this operation is not restricted to be used in these dimensions. For a 2D-convolution in layer l a 2D-kernel w_p is used to slide over the p-th channel of input x^{l-1} and the results are summed to compute the output y^l for each individual pixel (for simplicity we omit the layer index l):

$$y_{rs} = b + \sum_{p=1}^{P} \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} w_{ij,p} \cdot x_{(r-i),(s-j),p}, \qquad (1)$$

where $w_{ij,p}$ is the shared weight at position (i, j) in the kernel of size I x J for the p-th input channel and P is the total number of input channels from the previous layer. $x_{rs,p}$ is the input at location (r, s) from input channel p and y_{rs} is the output at location (r, s), with r and s indicating the index of the current pixel in the two dimensions in which w is shifted.

In contrast to the 2D convolution, the 3D convolution shifts the 3D kernel in three dimensions:

$$y_{urs} = b + \sum_{p=1}^{P} \sum_{k=0}^{K-1} \sum_{i=0}^{I-1} \sum_{j=0}^{J-1} w_{kij,p} \cdot x_{(u-k),(r-i),(s-j),p}, \quad (2)$$

where $w_{kij,p}$ is the shared weight at position (k, i, j) in the kernel of size K x I x J for the p-th input channel, P is the total number of input channels from the previous layer, $x_{urs,p}$ is the input at location (u, r, s) from input channel p and y_{urs} denotes the output at location (u, r, s), with u, r and s denoting the index of the current pixel in the three dimensions in which w is shifted. Note that r and s are spatial indices like in equation 1, whereas the interpretation of k and u depends on the specific way in which the 3D convolution is applied. In the following section, the different FCN variants based on 2D (equation 1) and 3D convolutions (equation 2) that are used in this work are introduced.

3.2 Network Architecture

3.2.1 Mono-temporal 2D-U-Net: The baseline network architecture used in this paper is a variant of U-Net (Ronneberger et al., 2015) designed for monotemporal Sentinel-2 imagery; it is shown in figure 1. The input layer has a size of B×(H×W), where H and W are the image height and width, respectively, and B indicates the number of spectral bands. We use the brackets to show in which dimensions the filter matrix is shifted, which means that $P = B$ and r and s from equation 1 are the row and column indices of the input. The encoder is composed of four convolutional blocks, each consisting of two convolutional layers with $I = J = 3$ followed by batch normalization (Ioffe and Szegedy, 2015) and a rectified linear unit (ReLU) as the activation function. We use zero-padding and to reduce the spatial dimensions by a factor of 2, we add a max-pooling layer at the end of each encoder block with a window size of 2×2 and a stride of 2. The black numbers in figure 1 indicate the number of output channels O in the convolutional blocks, which is doubled whenever the spatial resolution is reduced. The encoder is linked to the decoder by another convolutional block without a downsampling layer. The decoder consists of four upsampling layers that use bilinear interpolation, each followed by another convolutional block. Similar to U-Net, there are skip connections between corresponding layers of the encoder and the decoder; the corresponding features are concatenated before further processing. Finally, a 1×1 convolution maps the feature vectors to raw class scores, which are normalized by a softmax layer. In this paper, we refer to this variant as *2D-Unet*.

In the following the different modifications to this baseline are described that are used to integrate multi-temporal images.

3.2.2 Multi-temporal 2D-U-Net: To integrate multi-temporal images into *2D-Unet*, the spectral bands of the different timesteps are stacked by iteratively placing all spectral bands from the next timestep on top of the stack from the previous timestep. This is done to again obtain a 3-dimensional input tensor, resulting in an input of size $B \cdot T$×$(H \times W)$ with T being the number of used timesteps ($P = B \cdot T$ in equation 1). Again, r and s are the row and column indices of the input, indicating the dimensions in which the 2D kernel is shifted. We refer to this variant as *2D-Unet-ext* in the remainder of this paper.

3.2.3 3D-2D-U-Net: Following Oehmcke et al. (2019), who used a combination of 3D and 2D convolutions, we create an architecture with 3D convolutions in the first encoder block and 2D convolutions in all others. We create two variants of this architecture: In the first one the input is of size B×$(T \times H \times W)$, which means that the 3D kernel shifts in the spatial and temporal dimensions (the former corresponding to the indices r and s and the latter to u in equation 2), while the index p is

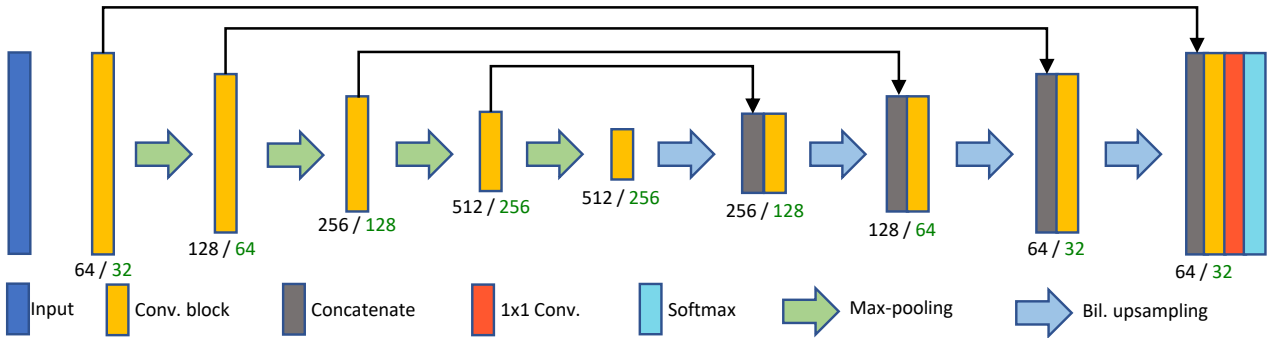


Figure 1. The network architecture. Black numbers at the convolutional blocks indicate the number of feature maps for *2D-U-Net*, *3D-2D-Unet* or *3D-U-Net*. Green numbers indicate the number of feature maps for *3-D-U-Net-light*.

related to the spectral bands, thus $P = B$. We refer to this variant as *3D-2D-Unet*^{t}. In the second variant, the input is of size $T\times(B \times H \times W)$ and the 3D kernel shifts in the spatial and spectral dimensions; the index u corresponds to the spectral dimension and p is the index of the temporal dimension, thus, $P = T$. We will refer to this variant as *3D-2D-Unet*^{b}.

The 3D kernel has a size of $K = I = J = 3$. After the 3D convolutions the tensor needs to be reshaped for all following convolutional blocks, which are based on 2D convolutions (equation 1). This is done by stacking all the individual 3D-feature maps that are obtained when iterating in the first dimension (P in equation 2). The feature maps of the first and second dimension (T and B) are stacked, retaining the size in the two spatial dimensions H and W, which results in a tensor of size $B^1 \cdot T^1\times(H^1 \times W^1)$, where the superscript indicates that the values correspond to the size of the feature maps in layer 1. This is followed by a 1 x 1 convolution to reduce the number of feature maps to 64; these maps are passed on to the second convolutional layer (note that we use the same number of output channels O as for *2D-Unet*, see black numbers in figure 1).

3.2.4 3D-U-Net: For 3D-U-Net, we replace every 2D-convolution from the baseline architecture by a 3D-convolution. As for the *3D-2D-Unet* we create two variants: *3D-Unet*^{t} for an input of size $B\times(T \times H \times W)$ with the 3D kernel shifting in the spatial-temporal dimensions and *3D-Unet*^{b} for an input of size $T\times(B \times H \times W)$ with the 3D kernel shifting in the spatial-spectral dimensions. When we use the same number of output channels O as for the other variants the number of parameters increases significantly, namely from 13.5M (*2D-Unet*) to 40.2M. Therefore, we reduce the number of output channels by half (green numbers in figure 1), which results in 10.1M trainable parameters. We refer to the variant with half the number of output channels as *3D-Unet-light* and to the full one as *3D-Unet*. In the experiments, we focus on *3D-Unet-light* for a comparison with the other variants, as the number of trainable parameters is approximately equal.

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:

$$L_{CrEn} = -\frac{1}{N}\sum_n\sum_c C_n^c \cdot ln(y_n^c) \cdot cw_c. \quad (3)$$

In equation 3, $C_n^c = 1$ 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:

$$IoU_c = \frac{TP_c}{TP_c + FP_c + FN_c} \quad (4)$$

In equation 4, 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:

$$cw_c = (1 - \Delta IoU_c)^\kappa = [1 - (IoU_c - \frac{1}{l}\sum_{h=0}^{l} IoU_h)]^\kappa, \quad (5)$$

where ΔIoU_c is the difference between the mean IoU of all classes and the IoU of class c, and the hyperparameter κ is used to scale the influence of classes with a lower IoU on the results. These class weights are used in the loss (equation 3) during the following epoch.

4. EXPERIMENTS

4.1 Dataset

The 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 2020. 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) and the six bands having 20 m GSD. The latter are upsampled to 10 m using bilinear interpolation. All bands are normalized to zero-mean and unit standard deviation by using $v'_{i,b} = (v_{i,b} - \mu_b)/\sigma_b$, where μ_b and σ_b denote the mean and standard deviation of band b, respectively, which are computed for a part of the dataset that covers approximately 15% of the test site. The cloud mask is used to exclude parts of the images that contain more than 5% cloud coverage, which results in a different number of available images for different regions, with a number varying between 14 and 105 for the chosen time period of two years.

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 104 different land use classes, which is too detailed for automatic classification. To define a suitable class structure for land cover, 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 updated at irregular intervals that can vary between a few days and three years. For the experiments reported in this paper, one reference label image at the geometric resolution of the satellite imagery is created for each of the two years, and each Sentinel-2 image is combined with the label image corresponding to the year of its acquisition. 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 is split into tiles of 8×8 km^2 (800 × 800 pixels, we refer to them as BE8 tiles in the following), which leads to a total number of 950 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 2019 and 2020 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 to the amount of label noise to be expected in the remaining data. Most changes occur between the classes *Greenland* and *Agriculture*. Figure 3 shows the Sentinel-2 image for two acquisition dates for one of the corrected tiles and the corresponding corrected reference.

To create mono-temporal input data the training patches are randomly chosen from all available dates over the time period. To generate multi-temporal patches we split every year (January to December) into four time intervals. This results in the following time intervals: January - March, April - June, July - September and October - December. For each interval, the Sentinel-2 image acquired most closely in time to the middle of the interval is selected, e.g. the one acquired most closely to February 15th is selected for the first time interval. Thus, for each area in the dataset it is possible to create a multi-temporal input patch for 2019 or 2020 consisting of four images (one per interval). To be able to compare the FCNs trained on mono-temporal images with the FCNs trained on multi-temporal images, we use the BE8 tiles with the same acquisition dates during the evaluation

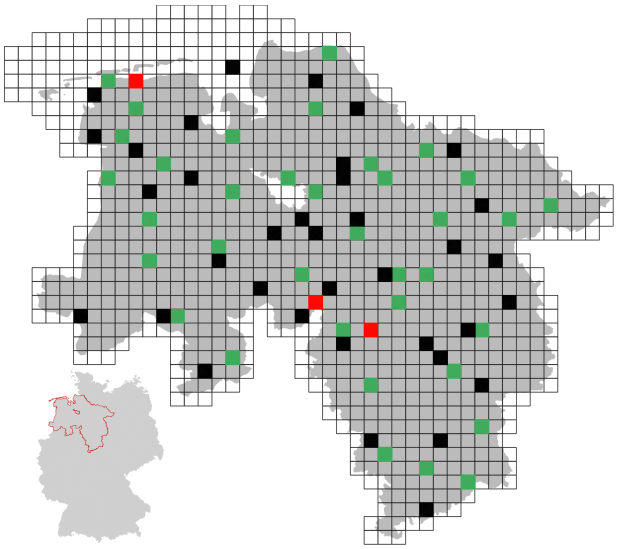


Figure 2. Overview of the available BE-8 tiles of 8×8 km^2 each. Grey / green: potential training / validation tiles. Red: test tiles with manually corrected reference (dataset R_1). Black: test tiles without corrected reference (dataset R_2).

Figure 3. A Sentinel-2 tile of size 8×8 km^2 for April 2019 (left), September 2020 (middle), and the corrected reference for 2020 (right). The colours correspond to: red - *bld.*, grey - *sld.*, yellow - *agr.*, light green - *grl.*, dark green - *for.*, dark blue - *fwt.*, light blue - *swt*, turquoise - *sea*, brown - *bar.*

of all experiments, which results in four times as many classified test tiles for the mono-temporal network variant as for the multi-temporal ones.

4.2 Experimental protocol

4.2.1 Experimental setup: For all experiments, we split our dataset into a set of 875 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 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 8 and is reduced to 6 for *3D-Unet-light* and to 2 for *3D-Unet*. During training the ADAM optimizer (Kingma and Ba, 2015) is used with the parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$ and a learning rate of 0.01 that decreases by a factor of 0.7 every 10 epochs.

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-V-3-2022-271-2022 | © Author(s) 2022. CC BY 4.0 License.

The parameter κ is set to 3, as this value resulted in a good trade-off between the accuracies of the over- and underrepresented classes. The learning rate and κ were tuned by using *2D-Unet* with mono-temporal images by choosing the parameter values with the highest mean F1-score on the validation dataset and these values were confirmed by a smaller set of experiments with the other architectures.

4.2.2 Evaluation protocol: For evaluation, the results of the FCNs achieved for the test tiles are compared to the available reference. As these tiles are larger than the input size to the network, the evaluation is done with a sliding window approach with a horizontal and vertical shift of 128 pixels. This results in four predictions per pixel, except at the edges of the BE8 tiles. 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, i.e. the harmonic mean of precision and recall, and the mean F1-score (m-F1), i.e. the mean of the F1-scores of the individual classes. While the OA can be biased by imbalanced class distributions, the m-F1 averages the F1-scores of all classes, so that the impact of a class with few samples on this metric is equal to the one of a class with many samples. These indicators are determined based on the three corrected test tiles (referred to as set R_1) as well as on the 39 non-corrected test tiles (referred to as set R_2). While the former is based on a small set of samples that is not affected by the errors in the reference, the latter forms a larger set of samples which, however, may be effected by label noise, i.e. errors in the reference labels. The distribution of class labels in the training and test datasets is shown in table 1. It can be seen that classes like *Agriculture*, *Greenland* and *Forest* clearly dominate the label distribution, while the classes *Barren land*, *Flowing water*, *Standing water* and *Sealed area* are clearly under-represented.

Set	Percentage of samples for each class [%]								
	stl.	*sld.*	*agr.*	*grl.*	*for.*	*fwt.*	*swt.*	*sea*	*bar.*
Train	7.3	0.2	31.0	17.3	16.4	1.4	0.6	9.9	0.6
R_2	9.0	0.3	44.1	19.5	18.8	0.2	0.8	4.0	0.6
R_1	7.9	2.2	56.5	12.0	9.0	1.2	1.6	9.0	0.6

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

4.2.3 Test setup: The evaluation is split into two parts. In a first set of experiments, we compare mono-temporal to multi-temporal classification. This is done using the architectures described in sections 3.2.1 and 3.2.2. *2D-Unet* is trained with an input size of 10×(256× 256), and for the multi-temporal *2D-Unet-ext* architecture, the four time steps per year are stacked to generate an input patch of size 40×(256× 256). For both architectures, the 2D kernels are shifted in the two spatial dimensions.

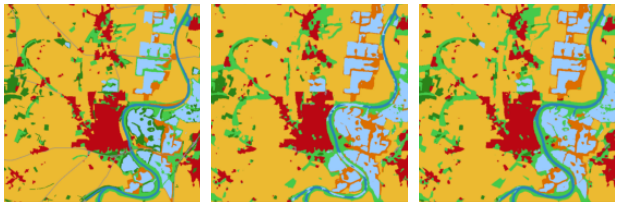
In the second set of experiments, we investigate how the classification performance is influenced by the application of a 3D convolution and by the way in which the it is performed. Accordingly, we use the *3D-2D-Unet*, *3D-Unet-light* and *3D-Unet* architectures described in sections 3.2.3 and 3.2.4. The input patches are of size 10×(4×256×256) when the kernel is shifted in the spatial-temporal dimensions and of size 4×(10× 256×256) when it is shifted in the spatial-spectral dimensions. All variants and the corresponding numbers of parameters are shown in table 2. For *3D-2D-Unet* the number of parameters only increases by a small amount compared to *2D-Unet*. Compared to the 2D variants, the number of parameters is about 18% smaller for *3D-Unet-light*. For the full *3D-Unet* architecture, however, the number is four times larger than the one for *3D-Unet-light*. This also has an effect on the training time, which is three to four times longer than that for *3D-Unet-light*. For this reason, in the discussion, we will focus on *3D-Unet-light* for the comparison.

Variant	convolution	# Par.
2D-Unet	spatial	13.4M
2D-Unet-ext	spatial	13.4M
3D-2D-Unet^t	spatial, temporal	13.5M
3D-2D-Unet^b	spatial, spectral	13.5M
3D-Unet-light^t	spatial, temporal	10.1M
3D-Unet-light^b	spatial, spectral	10.1M
3D-Unet^t	spatial, temporal	40.2M
3D-Unet^b	spatial, spectral	40.2M

Table 2. Different network variants, dimensions in which the convolutional kernels are shifted (convolution) and corresponding number of trainable parameters (# Par.).

4.3 Evaluation of the networks based on 2D convolutions

To assess the impact of integrating multi-temporal input data on the classification performance of a 2D-U-Net, we carried out experiments based on mono-temporal images (*2D-Unet*) and compare the results to those with multi-temporal input images (*2D-Unet-ext*) as described in section 4.2. Table 3 shows the results for both test datasets (R_1 and R_2). Examples for predictions on R_1 are shown in figure 4.



(a) reference (b) 2D-Unet (c) 2D-Unet-ext

Figure 4. Exemplary prediction results for one tile in R_1 using *2D-Unet* and *2D-Unet-ext*. Colour code: cf. figure 3.

The results show an increase of 4% in the m-F1 and 1.3% in OA on R_1, and an increase of 5% in m-F1 and 2% in OA on R_2 when multi-temporal images are used. For a comparison of the F1-scores of the individual classes, we use dataset R_1, as the manually corrected tiles are not affected by label noise. As expected, when using multi-temporal image data, the classes that contain vegetation (*Agriculture*, *Greenland* and *Forest*) show an increase in the F1-score; it is 3.5% for *Forest* and 2.4% for *Greenland*. The F1-score for *Agriculture* only increases by 0.4%, which could be due to the fact that the F1-score for this class is already high when using mono-temporal images. An inspection of the confusion matrix shows that the low F1-score of 51% for *Greenland* is mainly due to areas that are classified as *Greenland* but labelled as *Agriculture* in the reference (cf. the centre of the upper part of figure 4 for an example). This effect was also observed during the manual correction of R_1: in some areas, it was very difficult to distinguish between grass and young crops, and the usage of multi-temporal images seems to compensate this ambiguity only to a small amount. Surprisingly, the different classes of water show the largest improvement (14% for *Flowing water*, 7% for *Standing water* and 2% for *Sea*). The main reason for this effect seems to be a reduction

in the confusions between the classes *Flowing water* and *Standing water*. For instance, in the result of mono-temporal classification shown in figure 4(b), the river is predicted as *Standing water* instead of *Flowing water* in some areas that are close to lakes. These errors are completely eliminated when multi-temporal images are used to train the network (figure 4(c)). The class *Sealed area*, mostly corresponding to streets, has the worst results in all the experiments. In the results in figure 4, none of the streets are predicted correctly by any of the trained models. This is probably due to the small amount of training data (0.2% in the training dataset) and the low GSD of the data which results in a loss of fine structures corresponding to streets.

Overall, the usage of multi-temporal input data created in the way described in section 4.1 clearly improves the performance of the networks based on 2D convolution for all classes without requiring more trainable parameters.

4.4 Evaluation of the networks based on 3D convolutions

The experiments using the FCN variants based on 3D convolutions are designed to investigate if the performance of the models can be further improved in this way. Furthermore, we compare the two variants *3D-2D-Unet* and *3D-Unet-light* to find out how the performance varies depending on whether the 3D convolutions are only used in the first convolutional block or in the entire network. The results for all variants are shown in table 4.

4.4.1 3D-2D-U-Net: Both variants of *3D-2D-Unet* show a slight improvement in m-F1 and OA on R_1 compared to *2D-Unet-ext*. For *3D-2D-Unet*^{t} this improvement amounts to 0.4% in m-F1 and to 0.6% in OA; for *3D-2D-Unet*^{b} there is an improvement of 0.5% in m-F1 and 0.5% in OA. The class *Barren land*, which is also one of the under-represented classes, has the largest improvement of 5% and 7% for *3D-2D-Unet*^{t} and *3D-2D-Unet*^{b}, respectively. There is also a small improvement for *Greenland* (+1%) and *Agriculture* (+0.4%) for both variants. A slight decrease is observed for the classes *Forest*, *Flowing water* and *Sealed area*. These results are generally confirmed on R_2, where *Barren land* is again the class with the largest improvement (+5% for *3D-2D-Unet*^{t} and +4% for *3D-2D-Unet*^{b}) and small improvements for *Greenland*, *Agriculture* and *Settlement* can be achieved. There is a slight increase in OA for both variants but a slight decrease in m-F1. This is due to a larger drop in accuracy for *Flowing water* on R_2, which is mainly caused by confusion between areas corresponding to *Sea* that are classified as *Flowing water*. The results show that it barely matters whether the kernel is shifted in the spatial-temporal and the spatial-spectral dimensions: the corresponding network variants perform equally well and deliver slightly better results than *2D-Unet-ext*, in particular leading to a large increase of the accuracy for *Barren land*.

4.4.2 3D-U-Net: Using *3D-Unet*^{t}*-light* leads to a small increase in accuracy on both test datasets compared to *2D-Unet-ext* (0.4% in m-F1 and 1.2% in OA on R_1). Again the class *Barren land* is improved by the largest margin (+3.2% in F1, followed by *Greenland* (+1.2%) and *Agriculture* (+0.9%). We also observe a slight decrease in performance for the classes *Flowing water* and *Forest*. Overall, *3D-Unet*^{t}*-light* is the best performing variant of all our experiments with the highest F1-score for four of the nine classes. The model *3D-Unet*^{b}*-light*, which shifts the 3D kernel in the spatial-spectral dimensions shows a decrease of 0.1% in m-F1 but an improvement of 0.4% in OA compared to *2D-Unet-ext*. The decrease of the F1-score for *3D-Unet*^{b} is mainly due to the low F1-score for *Sealed area*, which is lower than for *2D-Unet-ext*. The F1-score of the class *Barren land* improves by 3.8% again, and there are slight improvements for *Settlement*, *Agriculture*, *Greenland* and *Flowing water*. Regarding *3D-Unet-light*, the usage of the 3D kernel in the spatial-temporal dimension outperforms the model with the shift in the spatial-spectral dimension, which is confirmed by similar results on R_2. To see whether the performance of *3D-Unet-light* can further be improved by using more trainable parameters, we also tested version *3D-Unet*. The results show very similar performance but no improvement, which indicates that the capacity of *3D-Unet-light* and *2D-Unet* as expressed in the number of trainable parameters of these networks is already sufficient for our classification task.

In comparison to *3D-2D-Unet* the variant *3D-Unet*^{t} slightly improves the performance for most classes. In contrast, the performance of *3D-Unet*^{b} is lower than both variants of *3D-2D-Unet*. The results show that the usage of 3D convolutions can indeed improve the accuracy for land cover classification, in our results especially for the class *Barren land*, but they are not needed in all convolutional layers of the network to achieve good performance.

Overall, the multi-temporal *2D-Unet-ext* already provides a good baseline with competitive performance compared to all the variants with 3D convolutions, which only improve the results by a small amount. The larger improvement between the mono-temporal and multi-temporal 2D-FCN indicates that the *2D-Unet-ext* is already able to extract the most important temporal and spectral features.

5. CONCLUSION

In this paper, we introduced a flexible way to generate multi-temporal input data for land cover classification based on time intervals. We investigated how the usage of these multi-temporal input data improves the performance of 2D-CNNs with a shift of the kernel in the spatial dimensions only and we also evaluated the impact of using different variants of 3D-CNNs, in which a 3D kernel is shifted in the spatial-spectral or the spatial-temporal dimensions. Our results show that the variant 2D-CNN with multi-temporal images as input already achieves competitive performance, outperforming the mono-temporal variant by 4% in m-F1 and 1.3% in OA. These results can be slightly improved when 3D convolutions are used in the first convolutional block or the entire network. All investigated variants based on 3D convolutions achieve relatively similar results, with a slightly higher performance of the *3D-Unet-light* using 3D convolutions in the spatial-temporal dimensions. A further increase in the number of trainable parameters, on the other hand, did not lead to an improvement of the results. We conclude that the 2D-U-Net trained with multi-temporal images learns the temporal and spectral features very well and should be sufficient for most practical applications.

In future research, we plan to investigate the performance of the suggested models when more time steps are used. This would allow to experiment with kernels of different sizes on the one hand and also to model time-dependent changes on the other hand. For the latter, the CNN needs to be adapted in a way that multiple outputs are possible, e.g. for every time step. Additionally, it would be interesting to see if the usage of all available image data during a time interval can further increase the classification performance. To achieve this goal, a selection step, e.g. based on an attention model, has to be integrated to

Variant	F1-scores on R_1 [%]									R_1 [%]		R_2 [%]	
	stl.	*sld.*	*agr.*	*grl.*	*for.*	*fwt.*	*swt.*	*sea*	*bar.*	m F1	OA	m F1	OA
2D-Unet	86.8	23.7	88.4	48.6	81.8	65.3	80.9	94.8	40.4	67.8	81.2	74.9	85.7
2D-Unet-ext	87.2	25.2	88.8	50.9	85.3	79.5	88.0	97.2	44.9	71.9	82.5	79.9	87.7

Table 3. Evaluation of land cover classification with mono-temporal and multi-temporal input data based on 2D convolutions.

Variant	F1-scores on R_1 [%]									R_1 [%]		R_2 [%]	
	stl.	*sld.*	*agr.*	*grl.*	*for.*	*fwt.*	*swt.*	*sea*	*bar.*	m F1	OA	m F1	OA
2D-Unet-ext	87.2	25.2	88.8	50.9	**85.3**	**79.5**	88.0	97.2	44.9	71.9	82.5	79.9	87.7
3D-2D-Unet^t	87.1	22.2	89.2	52.0	84.8	78.0	**89.6**	97.2	50.1	72.2	83.1	78.7	87.8
3D-2D-Unet^b	87.3	22.8	89.2	52.0	85.1	77.8	88.0	97.0	**52.3**	**72.4**	83.0	79.7	**88.1**
3D-Unet^t	87.0	**25.6**	**89.7**	**52.1**	84.5	78.3	88.1	**97.3**	48.1	72.3	**83.7**	**80.6**	87.9
3D-Unet^b	**87.5**	20.6	89.1	51.5	84.9	78.3	88.4	97.0	48.7	71.8	82.9	78.6	87.9

Table 4. Evaluation of land cover classification with multi-temporal input data based on different network architectures (cf. table 2).

obtain an input for the CNN with equal size for every patch. To improve the performance of classes with fine structures like streets we finally plan to integrate class labels with a finer resolution than the 10 m GSD from the Sentinel-2 images.

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 also want to thank the Erasmus Student Exchange program which made this joint research work possible.

REFERENCES

Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland (AdV), 2008. ATKIS®-Objektartenkatalog für das Digitale Basis-Landschaftsmodell 6.0. Available online (accessed 17 December 2021): http://www.adv-online.de/GeoInfoDok/GeoInfoDok-6.0/Dokumente/.

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

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.

Debella-Gilo, M., Gjertsen, A. K., 2021. Mapping seasonal agricultural land use types using deep learning on Sentinel-2 image time series. *Remote Sensing*, 13(2), Paper 289.

Fernandez-Beltran, R., Baidar, T., Kang, J., 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.

Ge, Z., Cao, G., Li, X., Fu, P., 2020. Hyperspectral image classification method based on 2D–3D CNN and multibranch feature fusion. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 13, 5776-5788.

Hagolle, O., Huc, M., Villa Pascual, D., Dedieu, G., 2015. A multi-temporal and multi-spectral method to estimate aerosol optical thickness over land, for the atmospheric correction of FormoSat-2, LandSat, VENμS and Sentinel-2 images. *Remote Sensing*, 7(3), 2668–2691.

Han, Y., Wei, C., Zhou, R., Hong, Z., Zhang, Y., Yang, S., 2020. Combining 3D-CNN and squeeze-and-excitation networks for remote sensing sea ice image classification. *Mathematical Problems in Engineering*, 2020, Paper 8065396.

Ho Tong Minh, D., Ienco, D., Gaetano, R., Lalande, N., Ndikumana, E., Osman, F., Maurel, P., 2018. Deep recurrent neural networks for winter vegetation quality mapping via multitemporal SAR Sentinel-1. *IEEE Geoscience and Remote Sensing Letters*, 15(3), 464-468.

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., Xu, W., Yang, M., Yu, K., 2013. 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1), 221-231.

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. *3rd International Conference on Learning Representations (ICLR 2015)*. Conference Track Proceedings.

Kussul, N., Lavreniuk, M., Skakun, S., Shelestov, A., 2017. Deep learning classification of land cover and crop types using remote sensing data. *IEEE Geoscience and Remote Sensing Letters*, 14(5), 778-782.

Li, W., Wu, G., Zhang, F., Du, Q., 2017a. Hyperspectral image classification using deep pixel-pair features. *IEEE Transactions on Geoscience and Remote Sensing*, 55(2), 844-853.

Li, Y., Zhang, H., Shen, Q., 2017b. Spectral–spatial classification of hyperspectral imagery with 3D convolutional neural network. *Remote Sensing*, 9(1), Paper 67.

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

Ma, W., Xiong, Y., Wu, Y., Yang, H., Zhang, X., Jiao, L., 2019. Change detection in remote sensing images based on image mapping and a deep capsule network. *Remote Sensing*, 11(6), Paper 626.

Mou, L., Bruzzone, L., Zhu, X. X., 2019. Learning spectral-spatial-temporal features via a recurrent convolutional neural network for change detection in multispectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 57(2), 924-935.

Oehmcke, S., Thrysøe, C., Borgstad, A., Salles, M. A. V., Brandt, M., Gieseke, F., 2019. Detecting hardly visible roads in low-resolution satellite time series data. *2019 IEEE International Conference on Big Data*, 2403–2412.

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.

Sellami, A., Ben Abbes, A., Barra, V., Farah, I. R., 2020. Fused 3-D spectral-spatial deep neural networks and spectral clustering for hyperspectral image classification. *Pattern Recognition Letters*, 138, 594-600.

Stoian, A., Poulain, V., Inglada, J., Poughon, V., Derksen, D., 2019. Land cover maps production with high resolution satellite image time series and convolutional neural networks: Adaptations and limits for operational systems. *Remote Sensing*, 11(17), Paper 1986.

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.

Zhang, H., Li, Y., Zhang, Y., Shen, Q., 2017. Spectral-spatial classification of hyperspectral imagery using a dual-channel convolutional neural network. *Remote Sensing Letters*, 8(5), 438-447.

Zhang, J., Wei, F., Feng, F., Wang, C., 2020. Spatial–spectral feature refinement for hyperspectral image classification based on attention-dense 3D-2D-CNN. *Sensors*, 20(18), Paper 5191.