Advanced Machine Learning
Final Project Report

Tackling the Land Cover Land Usage Challenge for German Cities with Deep Learning Approaches

Cem Daloglu
Matr. Nr. 3706511
M. Sc. Scientific Computing
cem.daloglu@stud.uni-heidelberg.de

Damjan Kalšan
Matr. Nr. 3707396
M. Sc. Data and Computer Science
damjan.kalsan@stud.uni-heidelberg.de

Lia Schmid
Matr. Nr. 3667760
M. Sc. Scientific Computing
lia.schmid@stud.uni-heidelberg.de

GitHub: https://github.com/cemdaloglu/AML_project

Heidelberg University
Institute of Mathematics & Computer Science

Contents

1 Contributions

Each group member contributed to all the sections of the report. The following subsections will state which sections are mostly written by the person on the title.

1.1 Cem Daloglu

Sections 2, 4.1.5, 4.2, 4.2.1, 4.3, 5, 5.1.

1.2 Damjan Kalšan

Sections 4, 4.1.1, 4.1.2, 4.1.3, 4.1.4, 4.2.2, 4.2.3 5.2, 5.3.

1.3 Lia Schmid

Sections 3, 5.4, 5.5, 6.

Abstract

Land cover classification is the process of segmenting the Earth's physical surface into a subset of classes from image-like data obtained by airborne objects. In this project, we prepare a custom dataset from Sentinel-2 multispectral imagery and Urban Atlas 2018 segmentation maps for eleven German cities and classify them into five classes. Three different deep learning models are applied to classify the labels of a land. U-Net, U-Net with a VGG16 feature encoder leveraging transfer learning, and the latter prepended with a shallow network that simulates trainable spectral index inputs. This report provides a comparison between the models in terms of accuracy and F1-score and is concluded with a segmentation of the entire Heidelberg and Frankfurt am Main functional urban areas, as well as a visual inspection of the best performing model.

Keywords: *Sentinel-2; Land Use Land Cover (LULC); U-Net; Transfer Learning, VGG16.*

2 Introduction

Understanding the transformation of land has proven to be a crucial task for urban planning to tackle the challenges of the growing population considering smart city expansion due to strategically beneficial locations or environmental impendencies. Especially with climate change progressing, an increasing amount of natural disasters such as floods or forest fires need to be taken into account.

Estimation of soil development used to be out carried by intense ground research which is time-consuming and locally limited to the city or country where extracting it (Talukdar et al., 2020). For this purpose, satellite images come in handy, as they open new and cost-effective opportunities to examine the evolvement of the world over time by capturing images of huge areas. There are several satellites providing accessible data. Examples are the Landsat, DigitalGLobe's satellites, and the Copernicus Sentinel-2 satellite. For the latter, the mission was launched in 2015 by the European Space Agency (ESA) providing high-resolution multispectral images with 13 spectral bands for a new viewpoint of the Earth's land and vegetation. Images obtained from the Sentinel-2 satellite are suitable for the land use land cover (LULC) classification problem. LULC classification aims to classify or segment lands with pre-determined classes. The dataset used for the project will be discussed in detail in Section 4 of this report.

To accomplish the LULC task, semantic segmentation using neural networks has proven to be beneficial. Several studies have used U-Net and VGG16 for the LULC classification problems and succeeded to distinguish several classes. Nevertheless, the problem has rarely been approached by transfer learning techniques. Transfer learning refers to the method of using knowledge from different but related source domains in the training process of a model. This should improve the learning process and has become a popular machine learning approach (Zhuang et al., 2020). Furthermore, which of the 13 spectral bands are most profitable is not yet fully understood.

We want to taylor the LULC task for investigating the vegetation of German cities, specifically to predict and segment Heidelberg and Frankfurt am Main using other German cities as training and validation sets. In our experiments, we used 4 spectral bands,

which can be found in Section 4.1.1, and two transfer learning models, one with pretrained weights and the other without, with VGG16 as an encoder of the U-Net model. Models will be explained and discussed in subsequent sections. In our literature review, we observed that overfitting and wrong predicting some unique labels are common problems of this task. Added features not to face these problems can be seen in Sections 4 and 5 of this report.

The objective of this project is to develop an automatic segmentation model incorporating a U-Net, VGG16, and transfer learning algorithms using images taken from Sentinel-2, preprocessing and labeling them accordingly to solve the LULC classification problem for German cities. Moreover, the results of the models will be compared in terms of accuracy, F1-score, and convergence time. We are expecting the transfer learning model with pretrained weights to converge faster. We expect to observe overfitting more in models containing more layers.

The outline of the papers is as follows. Section 3 surveys the related work for giving an opinion about previously implemented deep learning models and datasets. Section 4 gives an insight into the images and labels, pre-processing, model architectures, and evaluation metrics we used. Section 5 gives a detailed analysis of the conducted experiments and their results. The report is concluded with a discussion and summary in Section 6.

3 Related Work

Using satellite images for investigating surface composition is a recently tackled challenge for which machine learning approaches prove to be suitable. Based on prior work we elaborate on our research questions.

In 2018, Demir et al. (2018) presented a challenge targeting segmentation, detection and classification using remote sensing data. Providing datasets, their purpose was to assemble some benchmark data. With satellite images being more structured and homogeneous its objective is to improve population analysis, effective precision agriculture, autonomous driving, or map composition. In their Land Cover Classification Challenge the task is the automatic categorization and segmentation of land cover for sustainable development, agriculture, forestry, and urban planning. Additionally, they performed the LULC classification task on their generated data with a ResNet18 backbone and used data augmentation during training to prevent overfitting and weighted the classes according to their distributions. Thereby, they achieved an intersection over union (IoU) score of 0.433 in 30 epochs.

As mentioned before, a difficulty in LULC segmentation is the highly imbalanced class distribution also encountered in Chaudhuri, Dey, Datcu, Banerjee, and Bhattacharya (2021); Rakhlin, Davydow, and Nikolenko (2018). Therefore, a considered loss function and metrics need to be contemplated. Chaudhuri et al. (2021) accomplish an exhaustive analysis of the band properties of the Sentinel-2 data and investigate their abilities for performing different segmentation tasks. Thereby, they used a novel triplet loss function for

the model training. Also, for the 10m resolution bands, they used patches of size 120 $\times$ 120 pixels. They either use the weighted average of the number of samples or cleverly select batches to face the class imbalances. We aim to decrease this problem by already balancing the train, validation, and test data beforehand. Certainly, the class imbalance within each training, validation, and test set is still present. With their setup and training for 1000 epochs, they achieved a recall of about 69.72-72.75 on BigEarthNet-S2 (Sumbul, Charfuelan, Demir, & Markl, 2019). Their results indicate that the 10m resolution bands are good at classifying the underwater depth of ocean floors or lake floors but inferior for discerning finer-grained agro-foresty classes. This could suggest that our model will not perform optimally here either. One of their explanation was that human-made areas are assumed to be captured well by the 10m bands as it has a high spatial resolution.

To segment ten European countries Ulmas and Liiv (2020) used a modified U-Net architecture (Ronneberger, P.Fischer, & Brox, 2015) to target the LULC task. Data were retrieved from the BigEarthNet satellite image archive containing Sentinel-2 images. Their aim was to improve existing land cover maps as their creation still needs intensive preparation and time. As challenging they pose the task of gathering the ground truth labels for the satellite images for the training pipeline. This was achieved by first training a ResNet50 for 15 epochs for classification and then using the trained network as an encoder in the U-Net for segmentation. However, the ResNet was pretrained on the ImageNet dataset. It is important to notice that their dataset is much larger compared to ours. However, we aim to compensate for the size by restricting ourselves to German cities, thereby focusing on typically German geography and hence vegetation. Furthermore, they froze some parts for training and also encountered class imbalance trouble. For their experiments, they only used the RGB channels. In the segmentation task, the authors examined artificial surfaces, agricultural areas, forests, wetlands, and waterbodies. This will be a basis for our labeling as well. They achieved high F1 and IoU scores for forests, inland waters, and arable land.

A similar study was outcarried by Karra et al. (2021). They tackled the LULC task by examining Sentinel-2 satellite images. For this, they used data from California, Costa Rica, Belgium, and Laos, and trained a large U-Net model for semantic segmentation of the pixels into ten classes. As a loss function, they used the categorical Cross-Entropy and utilized inverse-log weighting for the ten classes, whereby unlabeled pixels were zero-weighted to ignore them. The authors made use of six bands of the L2A satellite data, namely red, green, blue, nir, swirl, and swir2. They augmented data by flipping vertically and horizontally. Also, they used a dropout of 20% and trained for 100 epochs from scratch without including transfer learning. With their architecture, they achieved overall accuracies of between 84% to 90% for the different countries. Facing segmentation difficulties for underrepresented classes they suggest that this might occur due to unevenly balanced classes in the validation set. As we target this problem before splitting our data, we hope to reduce the risk of confusion. The authors recommend including more data augmentation, class weighting, or other model architectures. Hence, in our study, these are parameters we want to investigate in more detail.

Similar classes are considered in Kotaridis and Lazaridou (2022). They use a U-Net to distinguish the classes built-up, vegetation, barren land, and water body and use a combination of spectral bands and indices for this purpose. With their computationally efficient network, the authors achieve an accuracy of over 90% training on the openly accessible Colab. They only used one city, namely Thessaloniki as training data and two cities for testing, which are Bari and Genova. They also retrieved their data from Sentinal-2 level-2A satellite images. This means, our training, validation, and test set is bigger but still in a comparable range. For their research Kotaridis and Lazaridou (2022) used the red, green, blue, and NIR bands and added three spectral indices. Their main concern was the data range of different cities. Hence, they applied data normalization to a range between zero and one, they used 64×64 windows and trained the model for 50 epochs.

An important linking point for our experiments is the study of Papoutsis, Bountos, Zavras, Michail, and Tryfonopoulos (2021). They explored a zoo of 56 models to work on a benchmark model studying LULC using Sentinel-2 data. Including CNN architectures, Multi-Layer Perceptrons, and Visual Transformers in their studies, they provide a good starting point in the design of our model. Furthermore, they stored all trained models and their pretrained models are accessible for further investigation. Our goal is to use their findings for transfer learning. As their VGG16 is a good trade-off with a high F1-score and a feasible training time, we decided to build upon that model.

4 Methodology

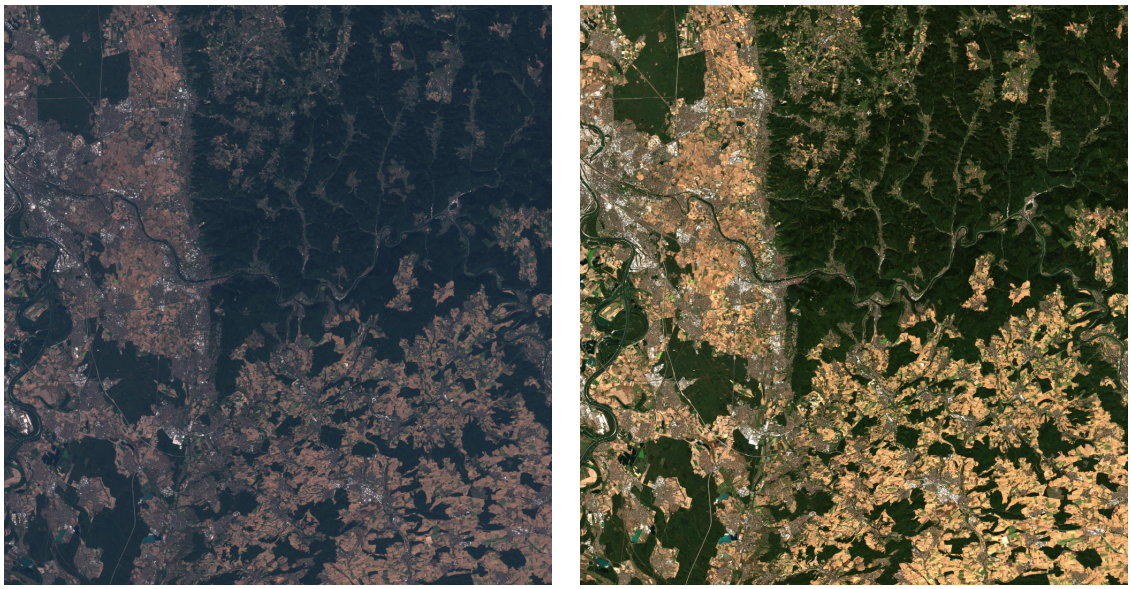
This section will provide detailed descriptions of the data we used, model architectures, and evaluation metrics we used. To be more specific, for the data subsection, the following topics will be surveyed; Sentinel-2 data, data preparation, preprocessing, patching, and the problems we have faced during these processes and how we overcame them. Three model architectures namely U-Net, transfer learning with VGG16, and U-Net with shallow network models will be investigated. Lastly, evaluation metrics that we used to monitor our model performances will be presented.

4.1 Data Description and Preparation

Satellite data was new for us wherefore it needed investigation to be understood. The following subsections will provide some insight into the data and how we dealt with the Sentinel-2 data and prepared it for the models.

4.1.1 Sentinel-2 Multispectral Imagery

The Sentinel-2 mission consists of two satellites with a sensing swath of 290 kilometers that revisit our area of interest every five days and capture high-resolution multispectral imagery (*Sentinel-2 MSI User Guide*, n.d.). The latter is freely accessible to the public through an online portal (*Copernicus Open Access Hub*, 2014) and consists of 13 distinct spectral



(a) L1C (b) L2A

Figure 1: Comparison of L1C and L2A products of Heidelberg FUA. The washed out appearance in L1C is due to atmospheric interference. Images are generated as per True Color Image (TCI) definition in (*Sentinel-2 MSI User Guide*, n.d.) subsection Definitions.

bands at different resolutions. In this project, we focus on bands B02 (blue), B03 (green), B04 (red), and B08 (near infrared), which are all available at 10-meter resolution. For more details about the available bands, we refer the reader to (*Sentinel-2 MSI User Guide*, n.d.), subsection Resolutions. The Sentinel-2 data is commonly distributed in the form of tiles, each spanning an area of 100×100 kilometers. These are available as two separate products: L1C or L2A, top-of-atmosphere or bottom-of-atmosphere reflectances respectively. L2A product can be generated by the user with freely available processing software named Sen2Cor, which performs atmospheric-, terrain, and cirrus correction of L1C data (*Sen2Cor - STEP*, 2021). This product is often used for land use land cover classification. The difference between the two is shown in Figure 1.

4.1.2 Urban Atlas

Urban Atlas (UA) is a product of Copernicus that provides high-resolution land use land cover maps for nearly 800 cities with more than 50,000 inhabitants distributed among EU, EFTA, and West Balkan countries plus the United Kingdom and Turkey (*Urban Atlas*, 2021). The cities together with some surrounding territories are considered as Functional Urban Areas (FUAs). An updated edition of the product is released approximately every 6 years, with currently available releases from the years 2006, 2012, and 2018. The land of an FUA is segmented into 27 distinct classes, hierarchically structured into four levels. In this project, we focus on the highest level, which considers five classes: ”Artificial surfaces”, ”Agricultural areas”, ”Natural and (semi-)natural areas”, ”Wetlands”, and ”Water”. Artificial surfaces represent the land where there is a strong human influence, such as buildings and

their associated land, roads, rails, ports, airports, construction sites, mines, dump sites, sports and leisure facilities, and green urban areas. Agricultural areas consist of arable land, permanent crops such as orchards and vineyards, pastures, and complex and mixed cultivation. Natural and (semi-)natural areas encompass forests, herbaceous vegetation associations such as bushy areas and natural grassland, and open spaces with little or no vegetation, for example, bare rocks and sand areas. Wetlands are split into coastal and inland and represent areas with specific vegetation that are flooded or are prone to flooding during a large part of the year. Lastly, the water class incorporates seas, lakes, ponds, rivers, and canals. For a more detailed description of the individual classes, we refer the reader to *Urban Atlas Mapping Guide v6.2* (2020), pp. 19–36. Urban Atlas presents a lucrative option for being used as a ground truth segmentation map for machine learning algorithms, however, its drawback is the temporal sparsity with which it is published. When used as such, one should acknowledge that the accuracy of the map labels lies at $\geq 80\%$.

4.1.3 Abandoned Initial Dataset

Initially, we set out to work with the MSLCC dataset (Bahmanyar, Espinoza-Molina, & Datcu, 2018), which contains a Sentinel-1 and a Sentinel-2 image with a detailed segmentation mask for two German cities, namely Munich and Berlin. The reason for our choice was that these cities are located in the same biome as Heidelberg to which we ultimately wanted to apply our model. Consequently, the visual features such as city design, agricultural layout, buildings, and natural land are similar and create a good baseline for the training data of our models. Moreover, as the annotations restrict the problem to four classes: "Built-up", "Agriculture field", "Forest", and "Water", we deemed the latter to be tractable within the given time constraints.

However, starting with the preparation of the test data for Heidelberg, we ran into issues. Specifically, we observed that the annotation process employed in MSLCC cannot be reproduced as it is not clearly stated in the paper. The authors mention that they fused annotations from two sources: Urban Atlas 2012 and OpenStreetMap (OSM) (OpenStreetMap contributors, 2017), but do not disclose which OSM tags and UA classes comprise a given MSLCC annotation label. After a detailed inspection, we found the following:

- "Fast transit roads and associated land" (UA code 12210), and "Open spaces with little or no vegetation" (UA code 33000) are not assigned to any MSLCC class.
- "Herbaceous vegetation associations" (UA code 32000) is assigned to the "Agriculture field", while in UA it is assigned to "Natural and (semi-)natural areas" which are more closely related to the MSLCC "Forest" class.
- "Wetlands" (UA code 40000) is assigned to "Water", while in UA these two classes are considered separately.
- The UA annotations on their own appear to be more detailed than MSLCC fused annotations. Moreover, the annotation contours are sometimes different.

- In MSLCC there are some unlabeled regions, that appear as shadows related to object contours even on entities such as rivers where shadows should not be present.

We presume that the authors decided to fuse the annotations because the UA segmentation masks available at their time of writing were already six years old and hence not representative of the Sentinel imagery they extracted for the year 2018. OSM data on the other hand is updated on a regular basis, due to its community-driven nature.

As we believed that the labeling process for the entire dataset should be the same, we decided to abandon the MSLCC in favor of using only UA annotations. This is possible because in 2018 an updated UA has been published and the satellite imagery for this period is also freely available to the public.

4.1.4 Final Dataset

We chose to create a custom dataset by ourselves, fusing Sentinel-2 imagery with Urban Atlas 2018 annotation masks. We restricted the extracted data to fall 2018, which is around the same time when the UA labels have been released and visually inspected by experts. Extracting the data close to inspection time gives us more confidence in the label correctness, as the entities on the ground could not drastically change in such a short period of time. This, however, restricts us from working only with a single season, meaning that the machine learning models trained on this dataset are highly unlikely to extrapolate predictions to other seasons. In the interest of time, we leave that to future works. The final criterion for choosing the extraction date was the very low presence or ideally no presence of clouds. Ultimately, we managed to fulfill all of these temporal constraints by extracting data on two separate days, as shown in Table 1.

As already mentioned in Subsection 4.1.3, we focus on a single biome, which forms the basis for the choice of FUAs that comprise the dataset. We chose 11 German cities, which are spatially close to Heidelberg and share similarities with it, such as for example architectural style or the presence of a river. Out of those, four are larger in size to additionally challenge the models' capabilities of predicting different-sized urban areas and structures that might not be present in smaller cities, such as airports and skyscrapers. This is also something we take into account when splitting the cities into the train, validation, and test subsets. Specifically, we made sure that each subset contains at least one larger city, and that the class distributions among the subsets are approximately the same. The chosen cities and their assigned subsets are shown in Table 1. The class distributions within the subsets and the proportion of each subset with regard to the entire dataset are shown in Table 2. One can observe that the dataset is rather imbalanced, which presents a challenge for learning, especially for the "Wetlands" class. We kept the latter in order to comply with the UA class hierarchy presented in 4.1.2, however, we acknowledge that the data for it is extremely lacking and that this will probably highly limit the models' ability to learn.

FUA	Geographic Bounding Box	Image Size	Sensing Date	Subset
Heilbronn	(8.82°, 49.02°, 9.53°, 49.39°)	5238 × 4152	2018-09-12	Train
Karlsruhe	(8.26°, 48.81°, 8.90°, 49.29°)	4745 × 5351	2018-09-12	Train
München	(10.75°, 47.82°, 12.26°, 48.62°)	11241 × 9044	2018-09-16	Train
Stuttgart	(8.75°, 48.50°, 10.00°, 49.08°)	9288 × 6516	2018-09-12	Train
Tübingen	(8.76°, 48.36°, 9.18°, 48.62°)	3145 × 2930	2018-09-12	Train
Würzburg	(9.37°, 49.48°, 10.56°, 50.24°)	8598 × 8611	2018-09-12	Train
Darmstadt	(8.52°, 49.72°, 9.05°, 50.00°)	3854 × 3147	2018-09-12	Validation
Freiburg am Breisgau	(7.53°, 47.74°, 8.44°, 48.26°)	6915 × 5895	2018-09-12	Validation
Mainz	(7.66°, 49.75°, 8.40°, 50.08°)	5261 × 3886	2018-09-12	Validation
Frankfurt am Main	(8.29°, 49.72°, 9.75°, 50.50°)	10598 × 8791	2018-09-12	Test
Heidelberg	(8.45°, 49.17°, 9.10°, 49.63°)	4826 × 5108	2018-09-12	Test

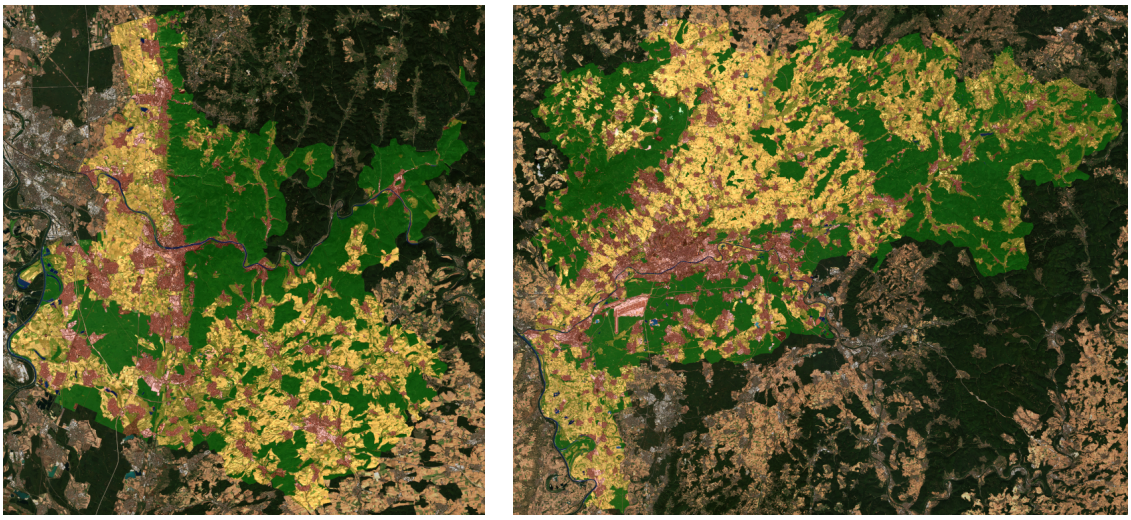
Table 1: General information about the selected FUAs comprising the dataset. The bounding boxes denote the enclosing area of the UA annotation mask as (min Lng, min Lat, max Lng, max Lat) in WGS84 projection. Extracted image size in pixels is given in (width × height), where 1 pixel equals 10 meters on the ground. Note that only a portion of the image is annotated, due to irregularly shaped UA annotation masks. The date column denotes the sensing date of the Sentinel-2 imagery.

Subset	Artificial	Agricultural	Natural and (semi-)natural	Wetlands	Water	Dataset proportion
Train	17.51%	48.99%	31.93%	0.02%	1.55%	62.37%
Validation	14.77%	44.29%	39.84%	0.02%	1.08%	15.17%
Test	19.89%	40.37%	38.86%	0.02%	0.86%	22.46%

Table 2: Class distributions and subset proportions of the final dataset. Percentages are computed on the basis of annotated pixels, ignoring the non-labeled regions.

In order to actually construct the final dataset, we followed the procedure shown in Figure 3. We decided to work with the L2A product Sentinel-2 imagery as it has been frequently used in related works. Firstly, we exported the UA 2018 annotations for our selected cities and a selection of Sentinel-2 multispectral images that completely covered the annotated regions. Then we semi-automatically generated the corresponding image and mask pairs for each separate city using GDAL/OGR (GDAL/OGR contributors, 2022) and Sen2Cor, the former being a common choice for handling GeoTiff files. In our case, we used its capabilities to merge multiple Sentinel-2 tiles into a single file, picking only a subset of bands, projecting UA into the common geographic coordinate system, and cropping the images to a specific bounding box. This entire process is rather delicate because a single mistake can result in an incorrect overlay of multispectral image and annotation mask, ultimately rendering the dataset unusable. To this end, we also decided to generate preview images, which can be used to inspect the dataset. An example of such previews is shown in Figure 2.

Lastly, we would like to point out that even though we only chose 11 cities, this procedure could be also automized to extract all the image-mask pairs available for the cities in Urban Atlas. Moreover, one could further expand it by integrating OSM tags, and hence



(a) Heidelberg (b) Frankfurt am Main

Figure 2: Test set FUAs overlaid with their respective UA segmentation masks. Note that some areas of the image are left unlabeled. These pixels are denoted as a special "No data" class in the mask.
■ Artificial areas, ■ Agricultural areas, ■ Natural & (semi-)natural, ■ Wetlands, ■ Water.

create a framework, where the models could be chronologically retrained with minimal human intervention.

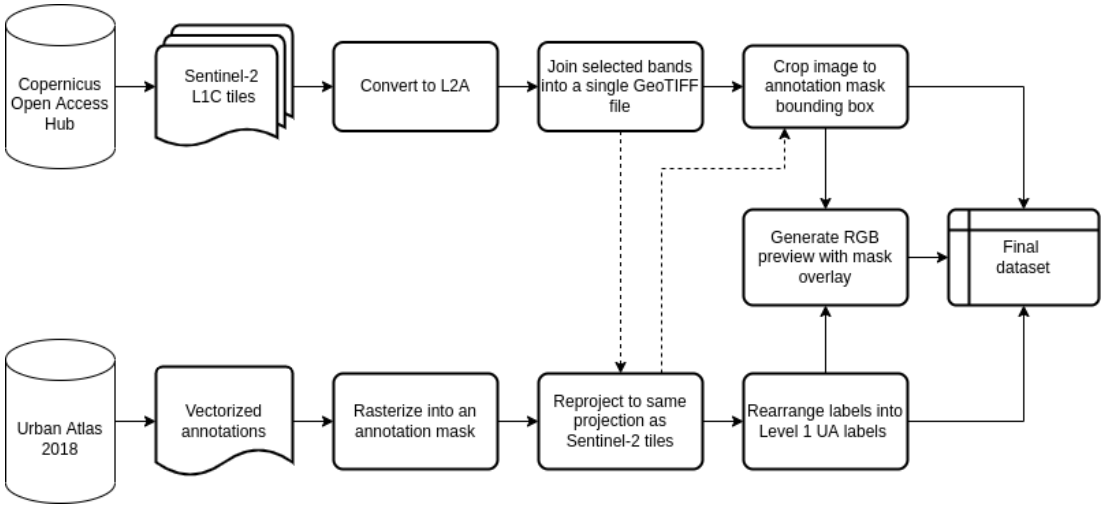


Figure 3: Description of the process used for transforming the raw Sentinel-2 and UA data into our final dataset.

4.1.5 Patching

The city images obtained from the Sentinel-2 imagery have very large image sizes and they are unique. Therefore, to work on the same image size we created patches with size 128×128 for each city. Image sizes (128×128) of the patches are decided according to the publications in the previous section. Since not all the images are divisible by 128, we

added a ”padding” option. ”padding” option adds an adequate number of pixels to the right side of the rows and bottom side of the columns of the image to make the original image divisible by 128. Added pixels are labeled as ”No-data” which does not contribute to the machine learning model. In addition to that, we also added an ”intersection” option which creates patch images with some intersected pixels with the previous patch image. Intersected pixels can be both from the vertical and horizontal sides. With ”intersection” we aim to achieve the border context. A visual explanation of ”padding”, and ”intersection” options can be seen in Figure 4.

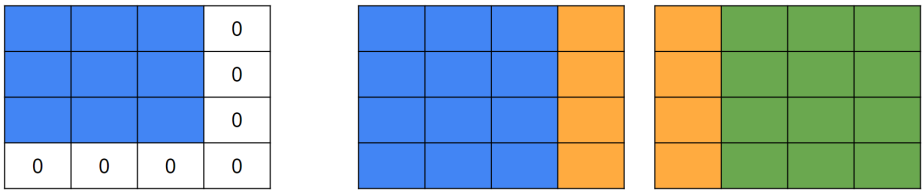


Figure 4: Impact of ”padding” (left) and ”intersection” (right) options.

The original image is considered as the 3x3 blue pixels on left in Figure 4. ”padding” option inserts 0 spectral values to the both right and bottom parts of the image. The right side of the same image represents two different patches with the ”intersection” option. Orange pixels show the intersected parts of the two images. It is not shown in the Figure but the bottom and the upper pixels can also be intersected.

Since we have enough data for training, neither ”padding” nor ”intersection” options are used. The last patches from the right and the bottom part of the image are discarded if they violate the 128x128 image size. On the other hand, we did not want to throw anything from the test set. Thus, ”padding” is used but not ”intersection” due to time conspiracy since we would have needed to define custom metrics. 8 pixels from both lateral and longitudinal parts are intersected.

As it can be seen in Figure 2, there are lots of pixels that are out of the city borders. Those pixels are considered ”No-data” and we obtained numerous patches full of ”No-data” labels. To save computation time and only use images with useful data, we decided to discard those patch images from the dataset.

4.2 Model Architecture

This section describes our merged model architectures. Our fundamental idea was to start with a U-Net and expand it to a transfer learning model by combining it with a VGG16 network. After the literature review, we decided to add a shallow network for spectral index computation before the U-Net to check if its performance improves. The following subsections will provide more details about the models.

4.2.1 U-Net

The U-Net algorithm is constructed by two paths named the contraction path which is also known as the encoder path and the symmetric expanding path which can be called

the decoder of the algorithm. The encoding algorithm consists of convolutional and max pooling layers that are added to the base sequential model. The decoding layer allows the localization of the data using transposed convolutional layers. With the two parts, an end-to-end fully convolutional network is constructed. Since the model contains only convolutional layers, the input shape can be formed for any given image shape.

Given the preliminary model as illustrated in Ronneberger et al. (2015), the convolutional layer outputs that are first used by the encoder paths use the traditional convolve-max pool strategy. The encoder layer ends up with 32×32 images at the lowest resolution for determining effective features. The lowest resolution images are then up-sampled with a recursive approach in which the previously encoded outputs are copied on the decoding stage. The up-sampling process continues with the neural network finding the output with the same shape as the input ending at the end of the output segmentation layers by using convolutional layers with decreasing filter size.

For this project, the created U-Net model also has the same depth as the inspired U-Net model which means there are four encoding layers, four decoding layers, and a bottleneck layer. However, there are some differences between our U-Net model and the one from Ronneberger et al. (2015). Our created model can be seen in Figure 5.

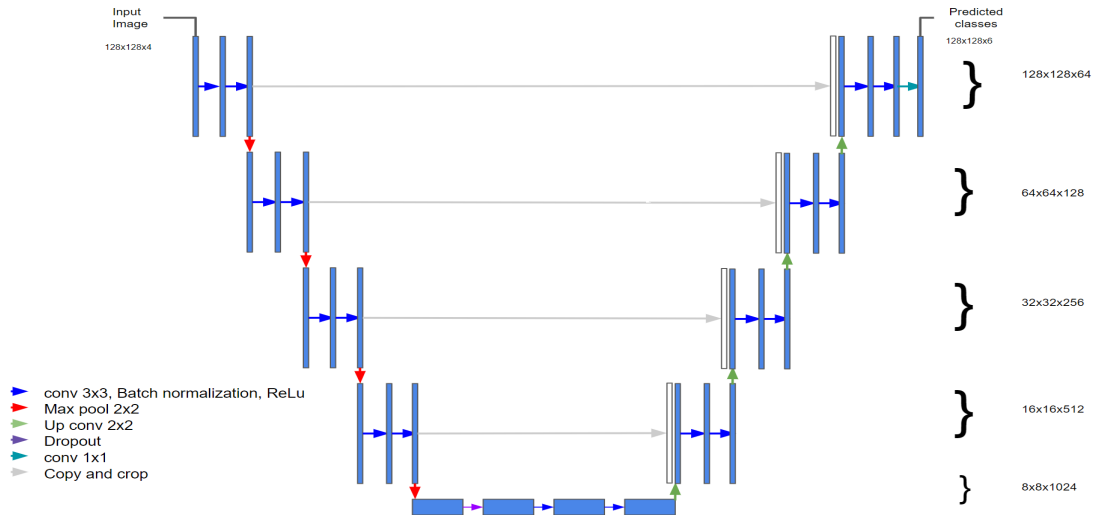


Figure 5: Implemented U-Net model.

The first difference is, batch normalization is applied after every convolution with kernel size 3×3. The objective to use batch normalization is to standardize the inputs to a layer for each mini-batch. This has the effect of stabilizing the learning process and dramatically reducing the number of training epochs required to train deep networks Brownlee (2019a). Batch Normalization is also a regularization technique, but that does not fully work like $l1$ or $l2$ regularizations. However, adding Batch Normalization can reduce the internal covariate shift and instability in distributions of layer activations. Thus, it can reduce the effect of overfitting Varma (2021).

Furthermore, a dropout layer is inserted after the encoder part with a $dropout\_rate = 0.25$. The dropout layer randomly sets input units to 0 with a frequency of rate at each step

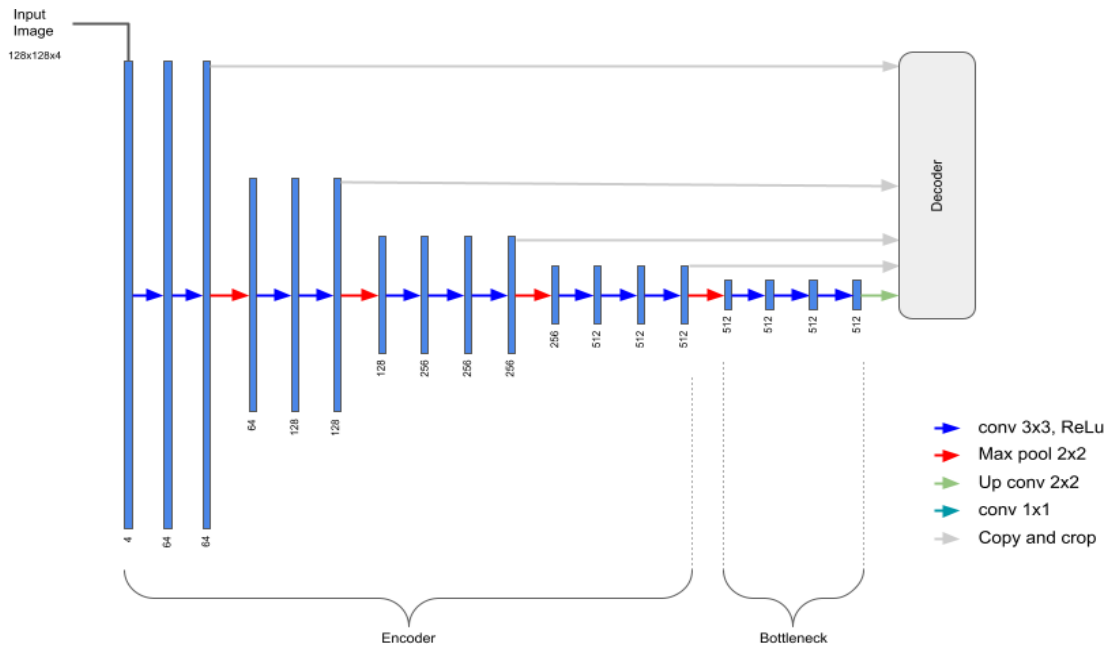


Figure 6: U-Net architecture with a VGG16 spanning the encoder and bottleneck part. The decoder is symmetric to the encoder and not explicitly drawn for brevity.

during training time. Inputs not set to 0 are scaled up by $1/(1 - rate)$ such that the sum of overall inputs is unchanged. The purpose of those changes is to prevent the model from overfitting Brownlee (2019b).

4.2.2 Transfer Learning with VGG16

For this model, we tried to leverage one of the pretrained models from Papoutsis et al. (2021) as the feature encoder of our U-Net architecture, shown in Figure 6. The authors trained 56 different classification models on the BigEarthNet dataset (Sumbul et al., 2019) and performed an in-depth comparison. Out of those, we decided to take the VGG16 model (Simonyan & Zisserman, 2014), due to its good performance in their experiments, reasonable training time, and ease of implementation. One important thing to note is, that their model is pretrained on 10 Sentinel-2 bands (all bands, excluding B01, B09, and B10), while our model only makes use of four. This means that only the pretrained kernels of the very first convolutional layer work on the exact same input as they were trained on, and all subsequent might not be useful at all, since their kernels assume that the layers before them worked with 10 bands.

4.2.3 Shallow Network for Index Computation

As mentioned in Subsection 3, some researchers have previously shown that using special image indices as additional inputs to the network can improve its performance. Indices are functions computed from the original bands working on a per-pixel basis and are used to emphasize a specific phenomenon. For example, the normalized difference vegetation

index defined as

$$NDVI = \frac{NIR - Red}{NIR + Red},$$

expresses the greenness of an image, and can therefore be used to better locate vegetation. In the equation above NIR denotes Band 8 and Red denotes Band 4. There are many more such indices, each serving a specific purpose, however, the list is finite. Hence, we were curious whether we could improve our model's performance, by allowing it to mix its own palette of indices from the input bands. Therefore, the network as a whole could enrich its input by computing indices that make specific phenomena stand out and thus improve the prediction. We decided to realize this idea by prepending a simple shallow subnetwork to the input of our U-Net model, as shown in Figure 7.

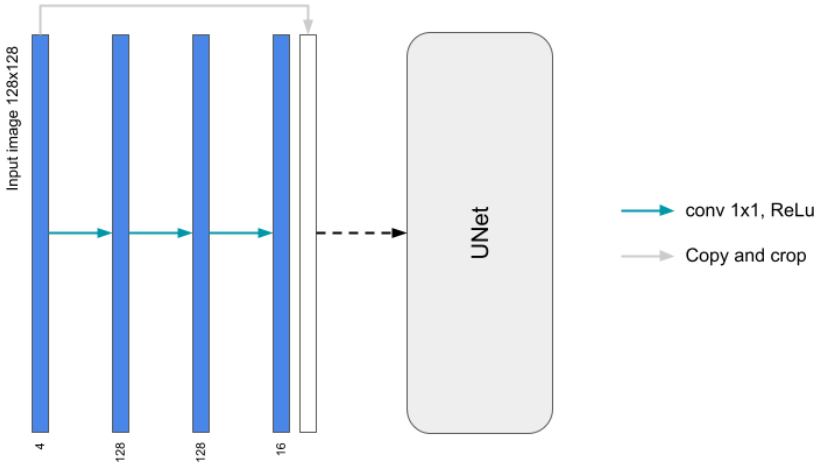


Figure 7: U-Net architecture prepended by a shallow network used for computing image indices. The 1x1 convolutions ignore spatial information and simulate the mixing of input bands into a palette of learnable indices. Skip connection retains the original bands and allows the subnetwork to focus exclusively on indices.

4.3 Evaluation Metrics

Dealing with a multi-class segmentation problem, there are some commonly used metrics to investigate the model's performance. Thereby, we consider the accuracy, precision, recall, and F1-score. Let $Y = \{y_i\}_{i=1..N}$ and $\hat{Y} = \{\hat{y}_i\}_{i=1..N}$ be the true labels and the model predictions respectively. Let us furthermore define True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). A true positive is an outcome where the model correctly predicts the positive class. Similarly, a true negative is an outcome where the model correctly predicts the negative class. A false positive is an outcome where the model incorrectly predicts the positive class. And a false negative is an outcome where the model incorrectly predicts the negative class. In the following Table 3, it is listed how to evaluate the classification models using metrics derived from these four outcomes and model predictions. Furthermore, we examine the confusion matrices for the test case. This results in a 6×6 matrix again considering the TP, TN, FP, and FN. In the multiclass case, the TP value is where the true label and the prediction are the same. The FN value

for a class refers to the sum of values of corresponding rows except for the TP value. The TN value for a class is summing all values of all columns and rows except the values of that class for which one is calculating the values.

Accuracy	**Precision**	**Recall**	**F1-Score**
$\frac{1}{N}\sum_{i}^{N} 1\left(y_i = \hat{y}_i\right)$	$\frac{\mathrm{TP}}{\mathrm{TP+FP}}$	$\frac{\mathrm{TP}}{\mathrm{TP+FN}}$	$2\frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$

Table 3: Used metrics

To train the model we either used the Cross-Entropy loss or the weighted Cross-Entropy loss (1), ignoring the unknown class labels, which are 0. The formula only differs in the weighting factor w_{y_n} which is uniform in the unweighted case.

$$\ell(x, y) = L = \{l_1, \ldots, l_N\}^\top, \quad l_n = -w_{y_n} \log \frac{\exp\left(x_{n,y_n}\right)}{\sum_{c=1}^{5} \exp\left(x_{n,c}\right)} \cdot 1\left\{y_n \neq 0\right\} \tag{1}$$

5 Experiments and Results

In this section, we present all of our outcarried experiments. Optimally, we only altered one specific part of the training process for better comparison. In addition to the main experiments, we also conducted several experiments to tune the hyperparameters. Among these, there are different batch sizes (32 vs. 64 vs. 128), whether using weighted or unweighted Cross-Entropy loss, the probability of flipping in data augmentation, or the number of layers of the network architecture. Also, the learning rate and weight decay of the optimizer can be tuned. As a baseline, we use a simple U-Net model which is explained in Section 4.2.1. Section 5.2 focuses on the Transfer Learning approach and Section 5.3 is using a shallow network for index computation.

For all experiments, we were in between the Stochastic Gradient Descent (SGD) and the adaptive moment estimation (ADAM) to use as optimizers. We chose the ADAM optimizer due to several reasons. We found that both optimizers can outperform each other in terms of accuracy and the difference is significantly small Brownlee (2021). However, convergence time or epochs were always less for the ADAM optimizers in our research Kingma and Ba (2014), Shaoanlu (2017). Publications in Section 3 mostly used the ADAM optimizer. Moreover, we clipped all band values to the range $[0, 6000]$, due to their occurrence distribution.

After tuning the hyperparameters, the models which gave the best results for the decided metrics is used to test the model. Experiments and results of each model are shown in different sections and the best results of these models are shown at the end of this section. Tables and plots are created to give a clearer understanding.

5.1 Simple U-Net Experiments

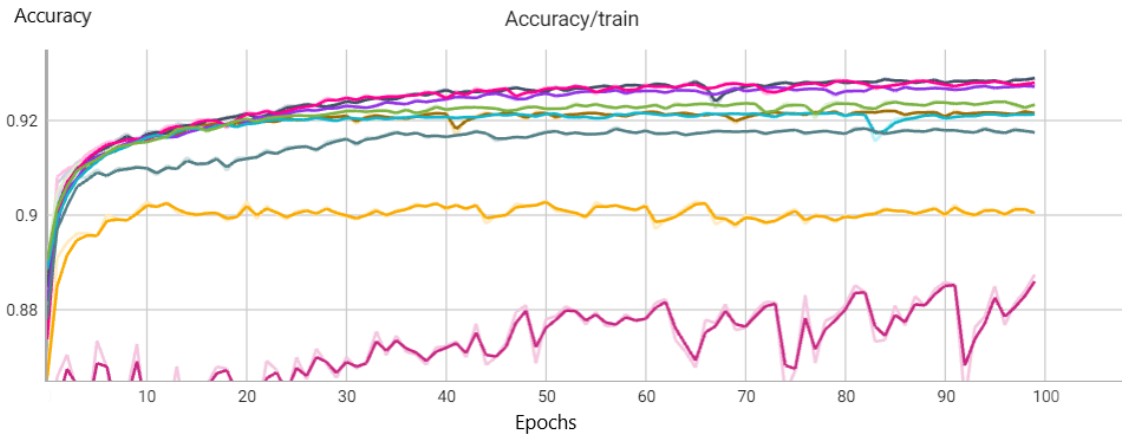
To investigate the U-Net model performance for the objective problem, we conducted several experiments. The purpose of these experiments is to obtain the best model for predicting the labels on the test dataset. The best metric results of the experiments for both training and validation data are discussed in the subsequent section and an overview of the experiments is displayed in Table 4. The progress of the metrics for 100 epochs of these experiments can be seen in Figure 8.

U-Net							Train		Validation		
Exp No	Layers	Batch Size	Learning Rate	Data Aug	Loss	Weight Decay	Accuracy	F1-score	Accuracy	F1-score	Epoch Time
1	4	32	0.001	20%	CEL	0.001	92.17%	87.87%	92.82%	86.75%	3.50
2	4	64	0.001	20%	wCEL	0.001	88.74%	84.60%	89.18%	83.61%	3.50
3	4	64	0.01	20%	CEL	0.001	90.31%	86.09%	90.96%	85.29%	3.50
4	4	64	0.001	50%	CEL	0.001	92.40%	88.08%	92.82%	87.02%	3.50
5	4	128	0.001	20%	CEL	0.001	92.76%	88.34%	93.17%	87.35%	3.50
6	4	64	0.001	20%	CEL	0.001	92.27%	88.41%	92.86%	86.52%	3.50
7	4	128	0.001	20%	CEL	0.01	91.78%	87.56%	91.66%	86.46%	3.50
8	3	128	0.001	20%	CEL	0.001	92.85%	88.49%	93.16%	87.34%	3.20
9	2	128	0.001	20%	CEL	0.001	92.86%	88.54%	93.17%	87.35%	2.50

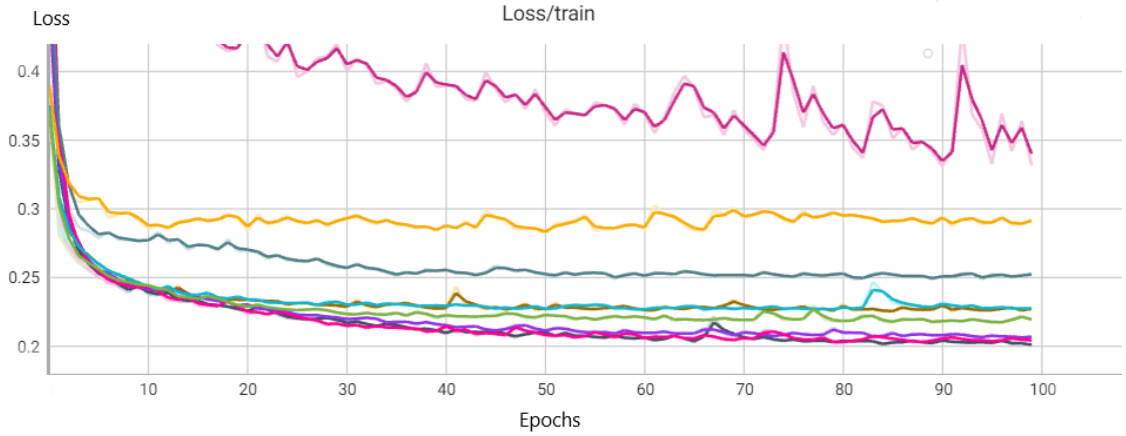
Table 4: Conducted experiments of U-Net and accuracy and F1-score results on the train and validation sets.

The first experiment we did was to tune the batch size. By looking at experiments 1, 5, and 6, it can be said that there is not much difference between the batch sizes 32 and 64, but 128 is slightly better than the others. For the loss function, experiments 2, and 6 show a difference, and the Cross-Entropy loss (CEL) performed much better than the weighted CEL. Comparing experiments 4 and 6 reveals that the percentage of the horizontal and vertical flips does not have much impact. Experiments 3 and 6 show that the smaller learning rate performs better and by checking experiments 5 and 7 the same holds for weight decay. Lastly, we wanted to check the effect of the number of layers. We used the same hyperparameters as in experiment 5 since it gave the best metric results for the 4-layer sized case, to construct 3 and 2-layer sized models. The layer size informs about the number of encoder and decoder layers in the U-Net model. Experiments 5, 8, and 9 show that increasing the number of layers did not improve the metric results. Although the metric results are almost the same, one epoch takes much less time as the layer size decreases.

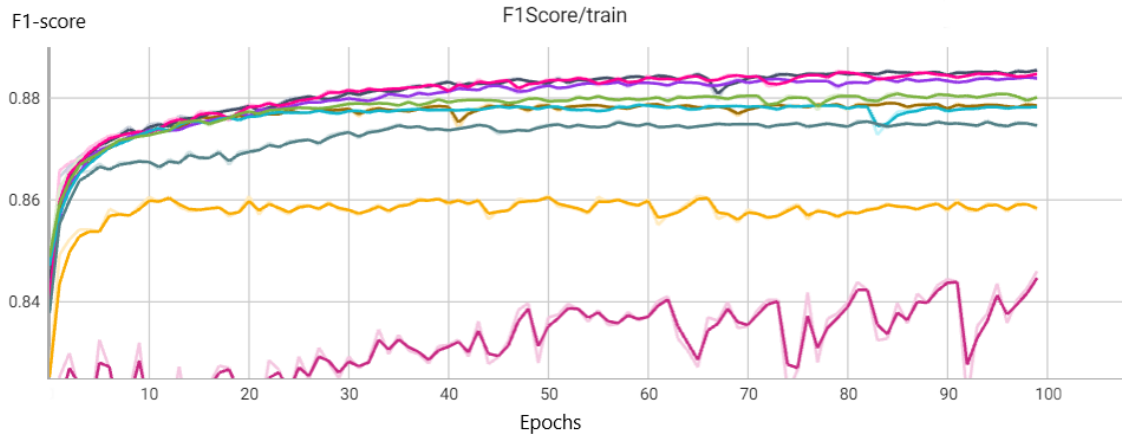
For testing, the models with experiment numbers 2 and 9 are chosen. Experiment 9 is chosen because it gave the best metric results and 2 using the weighted Cross-Entropy loss gave the worst training results. The purpose of choosing experiment 2 for testing is to monitor how training and validation differences between experiment 9 result in the test data.



(a) Accuracy



(b) Loss



(c) F1-Score

Figure 8: Train accuracy (a), train loss (b), and F1-Score (c) of U-Net experiments.
■ Exp. 1, ■ Exp. 2, ■ Exp. 3, ■ Exp. 4, ■ Exp. 5, ■ Exp. 6, ■ Exp. 7, ■ Exp. 8, ■ Exp. 9.

5.2 Transfer Learning With VGG16 Experiments

Based on the model description in 4.2.2, we decided to explore two questions:

1. Does the pretrained model save us training time and/or converge to a better solution?

2. Is the pretrained model useful if one uses just a subset of bands it was trained on?

For the experiments, we completely fixed the architecture, such that we could later load the pretrained parameters, and focused mostly on the hyperparameters. All conducted experiments are presented in Table 5 and Figure 9. We started off by comparing learning rates 0.001, 0.005, and 0.01 with batch size 64 and weighted Cross-Entropy loss. With 0.01, the loss exploded in the very first epoch, catapulting the model into a very bad spot. The same thing happened with the learning rate of 0.005, while with 0.001 the model remained stable and started converging. Therefore, we decided to fix the learning rate to 0.001. Next up, we tried increasing the batch size to 128, which again brought up the issue of exploding loss, however, this now happened mid-training around the 40^{th} epoch. We remedied this issue by adding L2-norm gradient clipping to 1.0 over all gradients. As we observed that the models began to overfit to the training data around 40^{th} epoch, we explored some regularization possibilities such as increasing the weight decay from the initial 0.001 to 0.01 and augmentation in the form of random horizontal and vertical flips, increased from initial 20% to 50%. None of these improved the results, hence we returned back to default values. Lastly, we decided to test out the unweighted Cross-Entropy loss, which resulted in an approximate 3.5% absolute increase in all validation metrics. This is not surprising, since our chosen metrics disregard the prior class probabilities, hence a loss optimizing for the unweighted case should perform better. Importantly, we also observed that using this loss, the model did not overfit even after 90 epochs, and decided to use it in our final model, with which we concluded the hyperparameter search. The final model was therefore trained for 135 epochs with an initial learning rate 0.001 decayed to 0.0001 and 0.00001 after 80^{th} and 120^{th} epoch respectively, batch size 128, L2-norm gradient clipping to 1.0, weight decay 0.001, and random horizontal and vertical flips with 20% probability. The total training took 7 hours and 1 minute on a single Tesla P100 GPU and the model scored 92.03% accuracy, 86.28% F1-score, 81.21% precision, and 92.03% recall on the validation set.

Having trained a model from scratch, the next task was to compare it to a fine-tuned pretrained model. In the interest of time, we used the previously found hyperparameters under the assumption that the optimal ones should not drastically change, since the entire architecture remains the same. The training procedure was as follows: we froze the trainable parameters of the encoder and bottleneck for the first 10 epochs, such that the decoder could adjust to the pretrained kernels. Then we enabled training for the entire network and trained it for additional 140 epochs. The learning rate schedule was the same as in the model trained from scratch and the total training took 7 hours and 19 minutes on a single Tesla P100 GPU, with the model scoring 92.61% accuracy, 86.82% F1-score, 81.72%

precision, and 92.61% recall on the validation set. Interestingly, the fine-tuned model outperformed the one trained from scratch in all measured metrics as well as the loss. The 0.6% absolute increase might seem marginal at first glance, but the previous model had already scored quite high, meaning that improving on it is not necessarily trivial and required the fine-tuned model to tend to the details. Unfortunately, the latter still took about the same amount of epochs to converge, which we assume is due to having to readjust a large part of the encoder and bottleneck parameters, as they assume a 10-band input. Nonetheless, the fact that we can use the pretrained model with just a subset of bands is an interesting discovery. We could attribute this to the high importance of selected bands, however, this hypothesis is something that would have to be explored in future works.

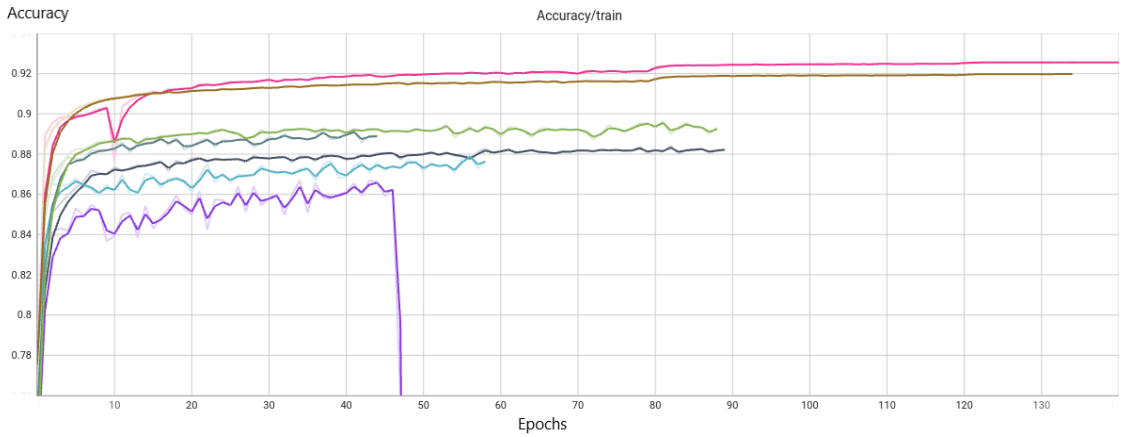
Transfer Learning								Train		Validation	
Exp No	Pretrain	Grad Clip	Batch Size	Learning Rate	Data Aug	Loss	Weight Decay	Accuracy	F1-score	Accuracy	F1-score
1	No	∅	64	0.001	20%	wCEL	0.001	87.69%	83.60%	84.24%	79.02%
2	No	∅	64	0.005	20%	wCEL	0.001	48.06%	45.81%	75.65%	70.97%
3	No	∅	64	0.01	20%	wCEL	0.001	32.76%	31.23%	14.78%	13.86%
4	No	∅	128	0.001	20%	wCEL	0.001	31.93%	30.44%	39.83%	37.34%
5	No	1.0	128	0.001	20%	wCEL	0.001	89.33%	85.15%	86.00%	80.63%
6	No	1.0	128	0.001	20%	wCEL	0.01	88.20%	84.09%	85.74%	80.39%
7	No	1.0	128	0.001	50%	wCEL	0.001	88.90%	84.76%	88.66%	83.12%
8	No	1.0	128	0.001	20%	CEL	0.001	91.92%	87.61%	92.03%	86.28%
9	Yes	1.0	128	0.001	20%	CEL	0.001	92.48%	88.15%	92.61%	86.82%

Table 5: Conducted exploratory experiments for the U-Net with VGG16 feature encoder. Experiments were ran with a varying number of training epochs with the goal of quickly determining the influence of hyperparameters, as described in Subsection 5.2. Hence, the experiment metrics should not be directly compared just based on this table. Experiment 8 represents the final model trained from scratch, and experiment 9 is the one using pretrained weights.

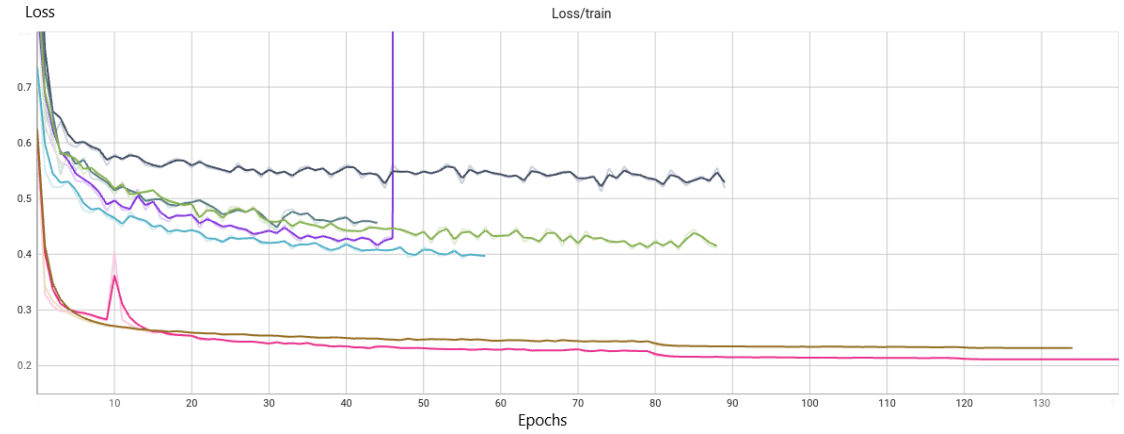
In conclusion, we find that the fine-tuned model using a pretrained VGG16 feature encoder outperforms the one trained from scratch even though we used just a subset of the original bands it was trained on. However, it still requires approximately the same amount of epochs to converge. We also note that the final model did not exhibit any overfitting behavior, despite the fact that it does not make use of additional regularization techniques such as dropout or batch normalization, but rather only weight decay and some training data augmentation.

5.3 Shallow Network for Index Computation Experiment

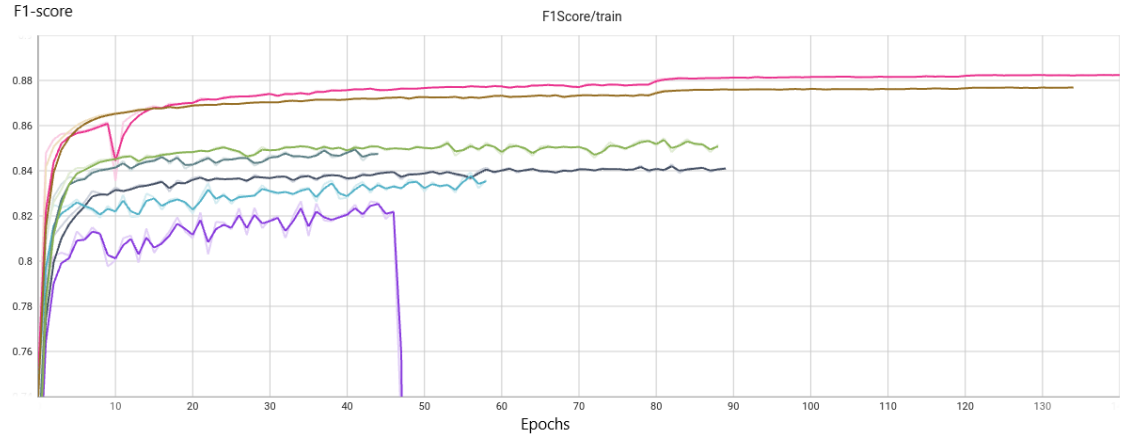
To examine whether a shallow network for index computation is beneficial, we began the experiment by testing whether this subnetwork can indeed learn to represent some of the existing indices. To this end, we generated synthetic data as our four-band input, and computed a selection of existing indices: NDVI, MSAVI2, BAI, and VARI, taken from (*Indices Gallery – ArcGIS Pro*, n.d.). The network was then trained to predict these values. We observed that it had no issues learning the first two, but failed to converge for the remaining



(a) Accuracy



(b) Loss



(c) F1-Score

Figure 9: Train accuracy (a), train loss (b), and F1-Score (c) of Transfer Learning experiments. Experiments 2 and 3 are out of view, due to extreme values caused by exploding loss.
Exp. 1, Exp. 2, Exp. 3, Exp. 4, Exp. 5, Exp. 6, Exp. 7, Exp. 8, Exp. 9.

two. Upon closer inspection, we found that the latter are defined on an infinite interval and hence cannot be well represented by a neural network, since the latter works well as a function approximator on a specific interval, but does not extrapolate well. Nonetheless, we continued with this idea.

In the next step, we prepended the shallow subnetwork to the architecture from Subsection 5.2 and followed the exact same training procedure, while also using the pretrained parameters for the encoder and bottleneck. The resulting model converged to the near-identical solution as the one trained without the prepended subnetwork, scoring 92.55% accuracy, 86.77% F1-score, 81.67% precision, and 92.55% recall on the validation set. This makes for an improvement of $\leq 0.1\%$ on each metric, hence we cannot conclude that this extension contributed much to the performance of the original network. At the same time, we note that this might be just the limitation of our dataset, due to label noise mentioned in Subsection 4.1.2, and should be further investigated in future work.

5.4 Summary of Results

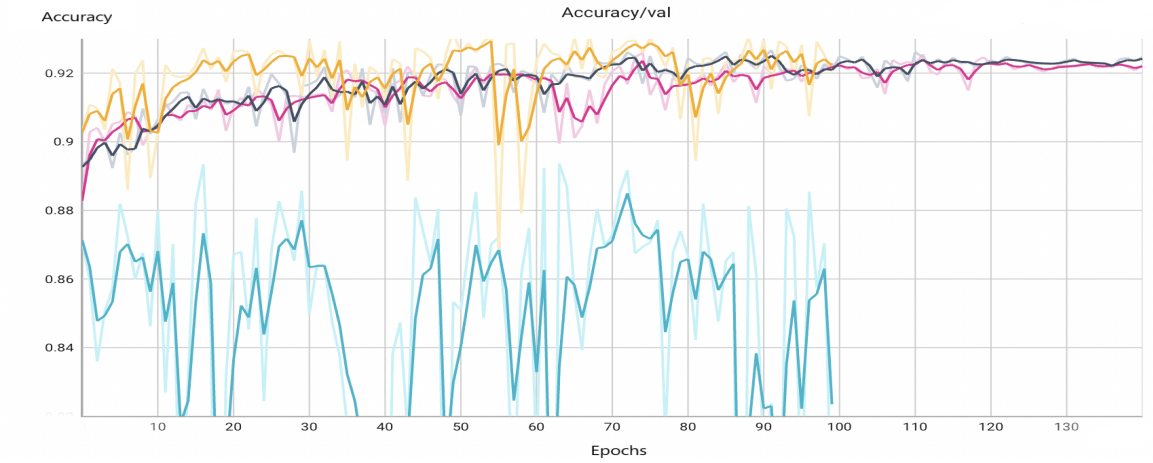
In this Section, we present the summary of the accuracy and F1-score for all of our main experiments for the training and validation phase and then the performance on the test dataset. The overview is shown in Table 6. For our final comparison, we chose experiments number 2 and 9 from the U-Net experiments as they had the worst and best training and validation accuracies respectively, and experiment 9 from the transfer learning inference. Furthermore, we chose the index computation experiment to run our test set on. Hence, we investigate four models in more detail. The validation accuracy, loss and F1-score is displayed in Figure 10.

The accuracy, as well as the F1-score, are the highest for the 2-layer U-Net. This means that more layers did not contribute to better learning. A reason for this might be that the four bands already provide sufficient information while remaining general and not overfitting and we could adjust the weights properly due to a large amount of data. Yet, we would have expected the transfer learning approach on satellite data to be favorable. However, probably as already indicated in Subsection 4.2.2 only the first convolutional layer has proven to be useful as the subsequent ones assume that the input before had been ten bands instead of just four as for our data. The shallow network for the index computation did not improve training but was in a similar range as the transfer learning approach.

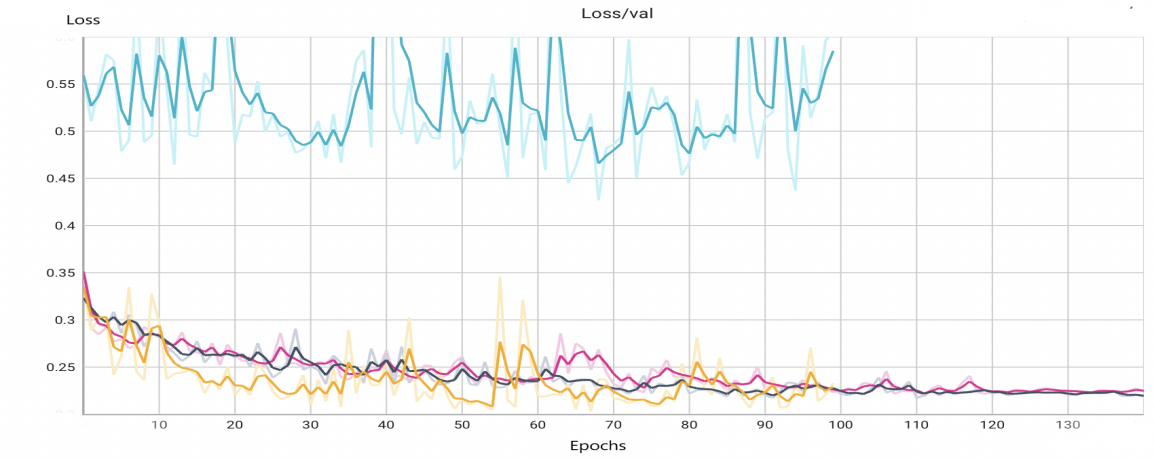
For testing, the 2-layer network performed best with an accuracy of 92.96% and an F1-score of 88.21%. Still one has to mention, that all models perform similarly good as the accuracy difference is just in a range of around 0.6%.

Our accuracies lie in a similar range as in Kotaridis and Lazaridou (2022) which is reasonable as they also used the RGB and near infrared channels. Also, our accuracy is better than in Karra et al. (2021), whereby it is important to mention that they used finer grained classes and evaluated on four entire countries.

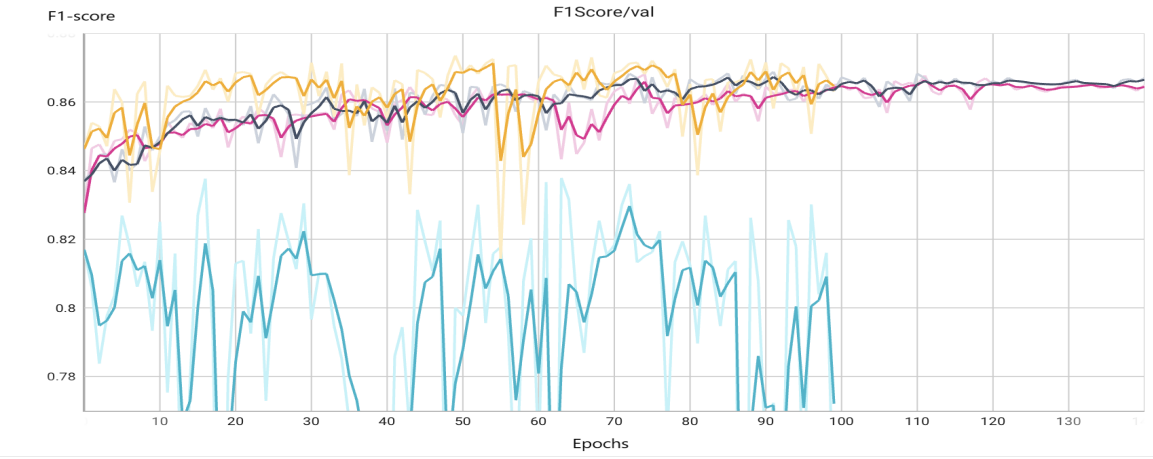
Likewise, it is important to inspect the confusion matrices of our four main models



(a) Validation accuracies of the chosen experiments.



(b) Validation losses of the chosen experiments.



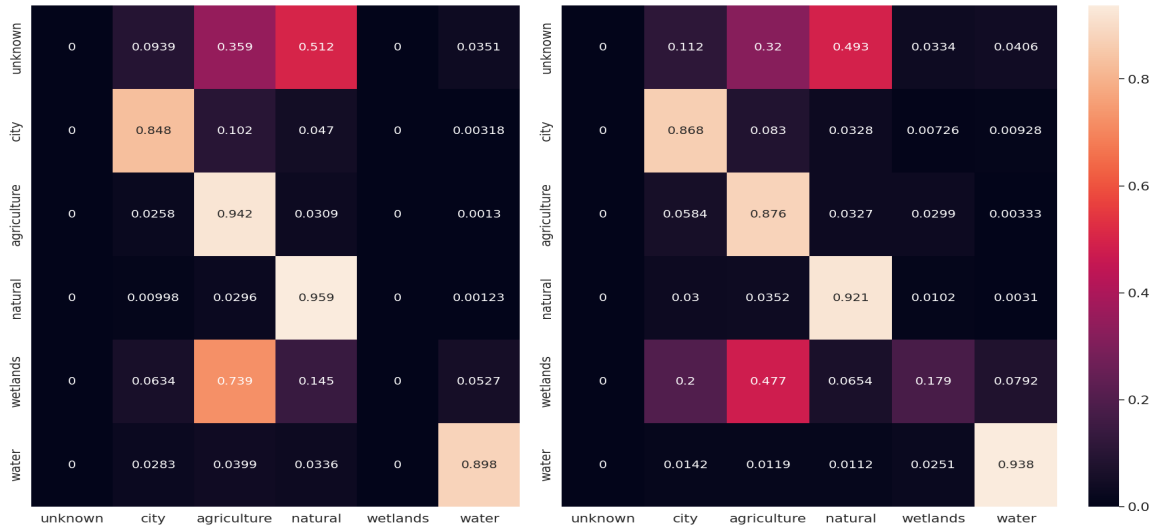
(c) Validation F1-Scores of the chosen experiments.

Figure 10: Validation accuracy (a), validation loss (b), and F1-Score (c) of the chosen four experiments.
U-Net Experiment 2, Shallow network experiment, U-Net Experiment 9, Transfer learning experiment 9.

	Train		Validation		Test	
Model	**Accuracy**	**F1-score**	**Accuracy**	**F1-score**	**Accuracy**	**F1-score**
4-layer U-Net	88.74%	84.60%	89.18%	83.61%	92.27%	87.55%
2-layer U-Net	92.85%	88.49%	93.17%	87.34%	92.96%	88.21%
Transfer Learning	92.48%	88.15%	92.61%	86.82%	92.78%	87.97%
Shallow Network with U-Net	92.38%	88.05%	92.55%	86.77%	92.68%	87.87%

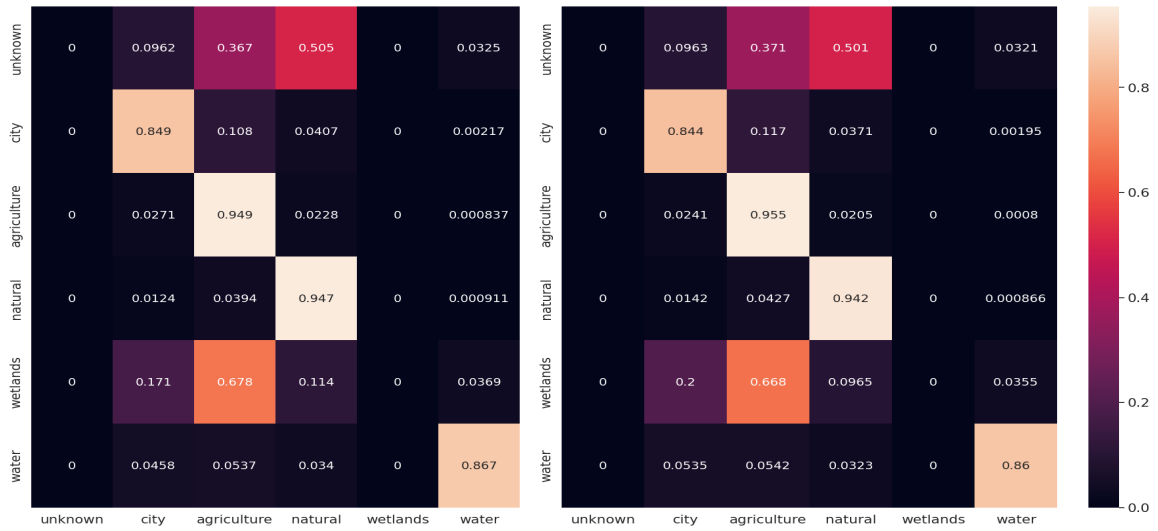
Table 6: Accuracy & F1-scores of the model's best weights on train, validation and test sets.

to get an quantitative insight for which classes the models predict well and where the difficulties of the predictions lie. The test results are shown in Figure 11 and show the predicted labels along the columns and the ground truth labels along the rows.



(a) 2-layer U-Net

(b) worst 4-layer U-Net



(c) U-Net VGG16 pretrained

(d) U-Net VGG16 pretrained index

Figure 11: Confusion matrices for (a) the 2-layer U-Net, (b) the worst 4-layer U-Net, (c) the transfer learning approach with the VGG16 encoder and (d) the pretrained index approach.

Except for the labels "wetlands" and "water", the 2-layer U-Net model outperformed

the worst 4-layer one. Importantly, note that the four layer network was trained with the weighted Cross-Entropy loss function. The other 4-layer networks using the unweighted Cross-Entropy loss performed similarly compared to the 2-layer U-Net. Hence, the weighting did indeed improve the wetland and water prediction. Also, note that the 2-layer network does not predict the wetlands, which is predicted by the 4-layer network which is desirable. Both models have in common that they mistakenly predict wetlands as being agricultural areas. This is even worse for the 2-layer network.

The transfer learning approach and the index approach are both worse in predicting water and lie in a similar range for doing so. Again, no weighting was used in the loss function explaining why wetlands do not contribute much to the loss. In general, the performance of both models is quite similar, indicating that the index computation was not really advantageous. Again, one can observe, that the scores for the right prediction are reasonably high for all models. This suggests, that a neural network approach is indeed a good choice for the LULC task.

In general, our F1-scores are a little less than the ones in Kotaridis and Lazaridou (2022) probably due to the fact that we did not only use one city as test data but two. Thereby, natural and semi-natural areas of Frankfurt are a little spread out and artificial areas and agricultural areas are more interconnected. Interestingly, their F1-score for water bodies is 100% which is very different than ours. Compared to the results in Ulmas and Liiv (2020), our models had fewer difficulties in predicting Artificial areas (cities) but their model was much better at predicting wetlands. However, they also used a pretrained ResNet50 on ImageNet to obtain the weights for the segmentation model.

5.5 Visual Inspection

To see how well our experiments work for the cities we chose for testing, namely, Frankfurt am Main and Heidelberg, it is crucial to do a visual examination after putting all patches together to embed the predictions into context and evaluate the models' performances also along borders of the patches. Herefore, we compare the ground truth labels with the prediction of the best model and the difference plots of our four main experiments, the 2-layer U-Net, the U-Net with the VGG16 trained from scratch, its pretrained version, and using a shallow network for index computation. Figures 12 and 13 show these plots for Frankfurt and Heidelberg respectively. One can observe that the edges to the black area representing unknown labels are not smooth in the best prediction image.

This is because during training we ignored these unknown labels. As expected, the models are hence not able to predict them and predict another label instead, resulting in sharp edges. Interestingly, also wetlands along the water are predicted although it is heavily underrepresented in the training, validation, and also test dataset. One explanation why the models are capable to detect wetlands not so badly might be as they occur near water as well. However, when examining it closely, one can still see that some wetlands are not detected at all.

For the difference plots, we discarded exactly these sharp edges and set them to zero.

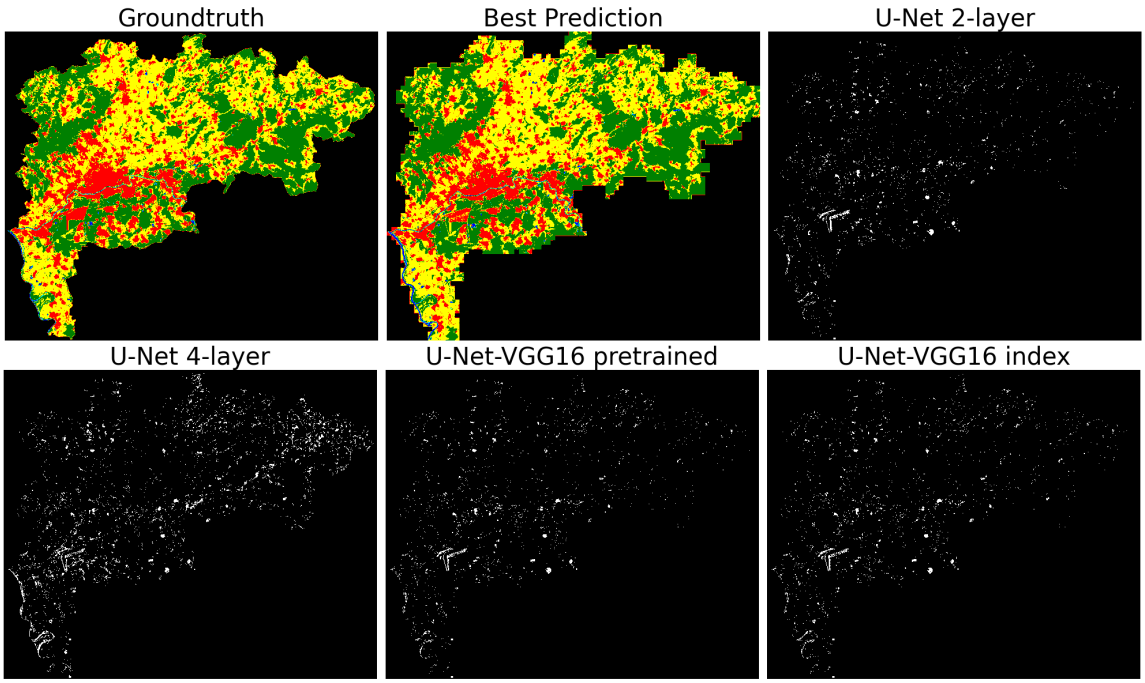


Figure 12: Frankfurt am Main. Ground truth, best prediction, and the difference plots for the corresponding networks.
■ Artificial areas, ■ Agricultural areas, ■ Natural & (semi-)natural, ■ Wetlands, ■ Water.

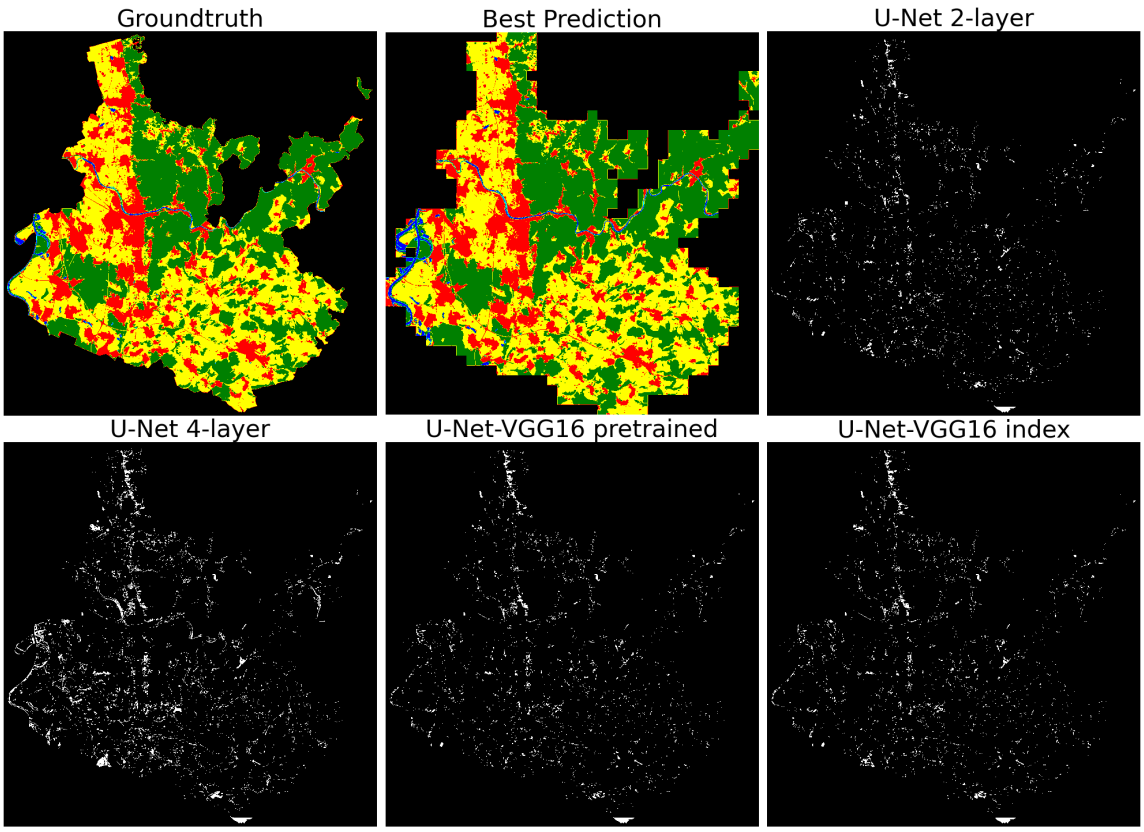


Figure 13: Heidelberg. Ground truth, best prediction, and the difference plots for the corresponding networks.
■ Artificial areas, ■ Agricultural areas, ■ Natural & (semi-)natural, ■ Wetlands, ■ Water.

The white areas depict the incorrectly labeled pixels. Noticeably, all models perform similarly in a visual inspection. One might see a slight disparity in the bottom left corner of the difference plots of Frankfurt.

The colors suggest that our model mixes up some artificial areas (red) with agricultural areas (yellow). This might be due to the usual rectangular shape of farmland and arable areas which are also typical for buildings. However, all difference plots have in common that they show an accumulation of white spots in the center of the images, and also a larger area to the left is distinctly mislabeled.

To study in more detail which classes the models fail, Figure 14 shows the five worst predicted segmentations in the upper row, their respective ground truth mask in the second row, the corresponding RGB image, and the near infrared image that were the input to the model in the third and bottom row respectively. For this illustration, we used the shallow U-Net, as it overall performed best.

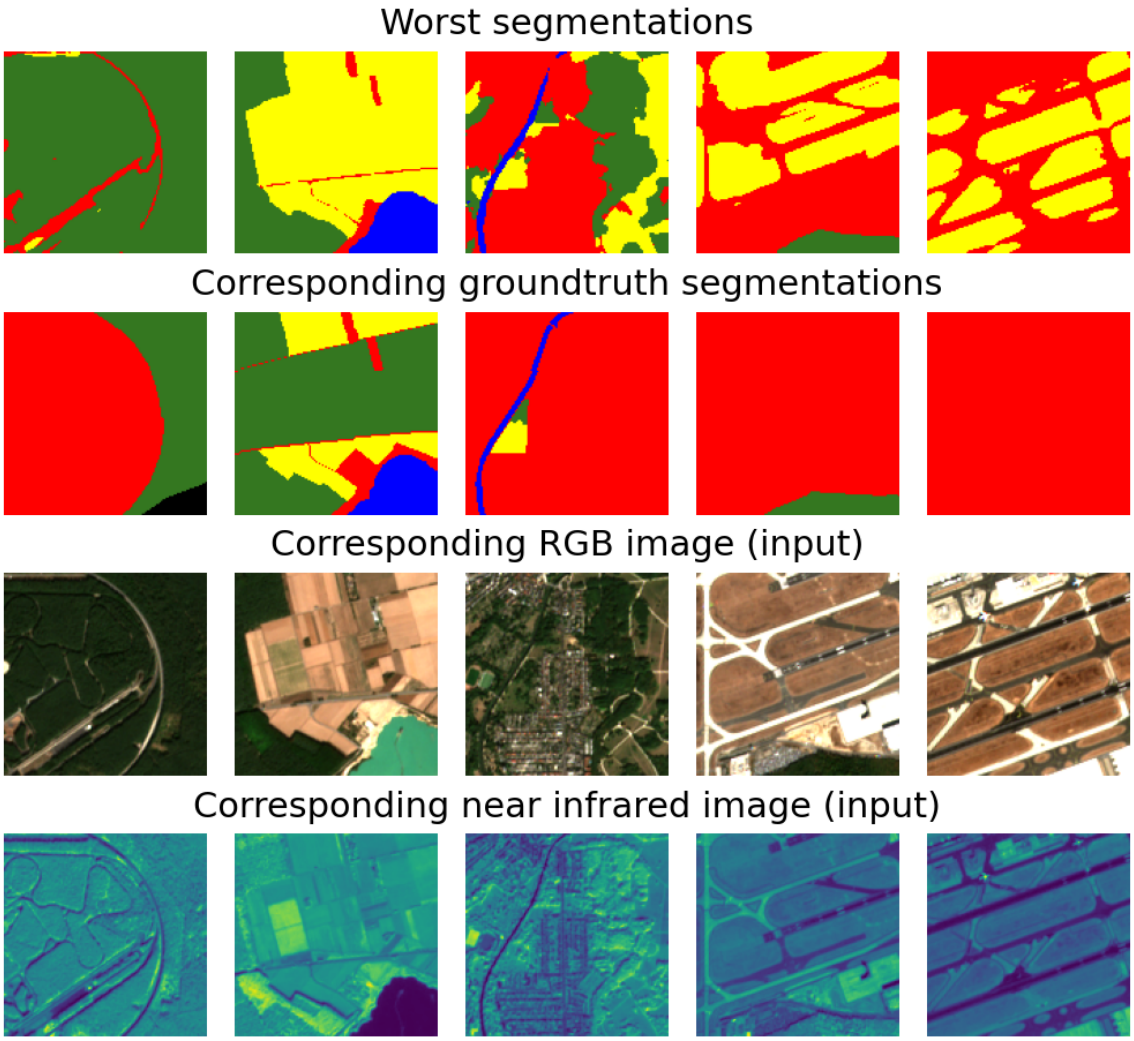


Figure 14: Worst segmentations of the 2-layer U-Net which was our best performing model in the upper row with the ground truth comparison in the second row, the RGB input image in the third row, and the corresponding near infrared image in the bottom row.
■ Artificial areas, ■ Agricultural areas, ■ Natural & (semi-)natural, ■ Wetlands, ■ Water.

For the first sample, our model predicted more natural area compared to the ground truth labeling. However, when visually examining the image, there indeed seems to be a forestal area which was identified by our trained model. For the second column, our model predicted more agricultural areas instead of natural and semi natural areas, which again seems reasonable when considering the corresponding RGB-input patch. Even the green area in between the artificial area of the third sample is detected by our model, which is again visible on both, the infrared and the RGB image. For the fourth and fifth samples, again the ground truth does not capture agriculture. However, in these cases, the visual inspection is indeed hard, as one cannot really tell what exactly is depicted in the patches. Our model as well as the ground truth caught the natural area in this case.

Interestingly, the wetlands are not contributing to our five worst patches. A reason for this could be that they in general occur so rarely not adding much to missing accuracy when evaluating the performance of the model. Astonishingly, our model seems to do better than the ground truth if one visually inspects the RGB input images and also the corresponding near infrared images.

6 Discussion and Conclusions

Surprisingly, examining the U-Net results of Section 4.2.1, using the weighted Cross-Entropy loss did not contribute to the model performing better compared to the common Cross-Entropy loss. This is against our expectations as the weighting should force learning underrepresented classes as well and correct prediction for frequently occurring classes should contribute less to the accuracy score. One reason for this might be that the imbalance is so intense that the punishment for overrepresented classes needs to be revisited, meaning that the weighting should further be investigated and adapted how to weight ”agriculture” or ”natural” and ”seminatural” areas compared to the underrepresented ”water” and ”wetlands”. Also, an entirely different loss function might even contribute to more promising results. Rakhlin et al. (2018) suggest using the Lovász-Softmax Loss (Berman, Triki, & Blaschko, 2018) which apparently also tackles the class imbalance challenge for satellite images. However, this is mostly used for the Intersection over Union (IoU) metric. Hence, it is a good starting point for investigation for the next work to incorporate the IoU with different losses.

Next, data augmentation is usually helpful for the generalization of the model. Still, using a flipping probability of 0.2 or 0.5 did not enhance training. This can be explained by the type of augmentation. Having enough training data, we encounter multiple perspectives and angles of different vegetation and construction. Hence, it is reasonable that horizontal and vertical flipping does not include additional valuable information to the data. In further experiments, one should consider alternative augmentation techniques. Preferably, one could apply color augmentations to the first three color channels. For instance brightness, contrast changes, saturation or hue modifications in data could be favorable as they might compensate for the different weather conditions and light reflections during the sensing period of the respective cities. The near infrared channel could be augmented

by adding some noise to it.

The learning rate of a model handles how much the model responds to the estimated error to variation each time the model weights are updated. The result of a too big learning rate being disadvantageous in training suggests that the model converges too quickly to a suboptimal solution learning unfavorable weights of the model. This is also visible in the training plots. Hence, with the tuned learning rate of 0.001, we found an adequate learning rate such that the model continues learning without becoming a long-drawn-out process that falters.

Often, a deeper network results in more accurate predictions. However, we faced that the depth of the U-Net did not improve training but just accelerated it. An intuitive explanation might be the following. When performing downsampling the U-Net learns what occurs in the image while losing information about its location. Due to class imbalances, the model might easily identify the most represented classes. Hence, the trade-off of losing the information about the class is inconvenient for the model, or in our case, it is just not beneficial and more time-consuming. This was seen in our experiments as the 2-layer and the 4-layer U-Net had similar performances when using the unweighted Cross-Entropy loss.

Examining our worst predictions as in Figure 14 we observe that they are not that bad overall. Keeping in mind that the correctness of the map labels lies at $\geq 80\%$, one could cautiously conclude that our approach would indeed improve the LULC problem and contribute to better labeling of cities or entire vegetation areas and countries. Karra et al. (2021) came to a similar conclusion. This can be a crucial step to drive precautionary safety for land development to investigate how cities grow, the environment changes due to climate change, or humane interference.

A step for further research based on our analysis is on using one of the pretrained visual transformers from Papoutsis et al. (2021) as the encoder. Due to restrictions in computing power and the limited amount of data, this was not possible for this project, as visual transformers are very data hungry. However, if the augmentations described above are included and the dataset is increased by a considerable amount, using a pretrained visual transformer would be an interesting experiment. Also, the inclusion of further bands enlarges the information that is retrievable from the data and could therefore be useful when training a visual transformer. The shallow network for index computation might compute more informative indices when using all 10 bands due to its nature of how each index is computed. We only retrieved data sensed at one specific time as shown in Table 1. Additionally, to include different seasons and certain changes over time in the cities already in training, different sensing times could improve the challenging task of predicting how land cover changes over time. We see our approach as the right step into a promising but very data hungry direction, that certainly will contribute to intelligent urban expansion and settlement of industries and agriculture to prevent settling in places that are unfavorable and not compatible with nature and to expand and improve warning systems in case there has already been ill-considered or imprudent humane construction.

References

Bahmanyar, R., Espinoza-Molina, D., & Datcu, M. (2018). Multisensor earth observation image classification based on a multimodal latent dirichlet allocation model. *IEEE Geoscience and Remote Sensing Letters*, *15*(3), 459–463.

Berman, M., Triki, A. R., & Blaschko, M. B. (2018). The lovász-softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4413–4421).

Brownlee, J. (2019a). *A gentle introduction to batch normalization for Deep Neural Networks.* Retrieved from `https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/`

Brownlee, J. (2019b). *A gentle introduction to dropout for Regularizing Deep Neural Networks.* Retrieved from `https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/`

Brownlee, J. (2021). *Gentle introduction to the adam optimization algorithm for deep learning*. Retrieved from `https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/`

Chaudhuri, U., Dey, S., Datcu, M., Banerjee, B., & Bhattacharya, A. (2021). Interband retrieval and classification using the multilabeled sentinel-2 bigearthnet archive. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, *14*, 9884–9898.

Copernicus Open Access Hub. (2014). Retrieved from `https://scihub.copernicus.eu/`

Demir, I., Koperski, K., Lindenbaum, D., Pang, G., Huang, J., Basu, S., ... Raskar, R. (2018). Deepglobe 2018: A challenge to parse the earth through satellite images. In *Proceedings of the ieee conference on computer vision and pattern recognition workshops* (pp. 172–181).

GDAL/OGR contributors. (2022). GDAL/OGR geospatial data abstraction software library [Computer software manual]. Retrieved from `https://gdal.org` doi: 10.5281/zenodo.5884351

Indices Gallery – ArcGIS Pro. (n.d.). Retrieved from `https://pro.arcgis.com/en/pro-app/2.8/help/data/imagery/indices-gallery.htm`

Karra, K., Kontgis, C., Statman-Weil, Z., Mazzariello, J. C., Mathis, M., & Brumby, S. P. (2021). Global land use/land cover with sentinel 2 and deep learning. In *2021 ieee international geoscience and remote sensing symposium igarss* (pp. 4704–4707).

Kingma, D. P., & Ba, J. (2014). *Adam: A method for stochastic optimization.* arXiv. Retrieved from `https://arxiv.org/abs/1412.6980` doi: 10.48550/ARXIV.1412.6980

Kotaridis, I., & Lazaridou, M. (2022). Semantic segmentation using a unet architecture on sentinel-2 data. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *43*, 119–126.

OpenStreetMap contributors. (2017). *Planet dump retrieved from https://planet.osm.org .*

https://www.openstreetmap.org.

Papoutsis, I., Bountos, N.-I., Zavras, A., Michail, D., & Tryfonopoulos, C. (2021). Efficient deep learning models for land cover image classification. *arXiv preprint arXiv:2111.09451*.

Rakhlin, A., Davydow, A., & Nikolenko, S. (2018). Land cover classification from satellite imagery with u-net and lovász-softmax loss. In *Proceedings of the ieee conference on computer vision and pattern recognition workshops* (pp. 262–266).

Ronneberger, O., P.Fischer, & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention (miccai)* (Vol. 9351, pp. 234–241). Springer. Retrieved from http://lmb.informatik.uni-freiburg.de/Publications/2015/RFB15a ((available on arXiv:1505.04597 [cs.CV]))

Sen2Cor - STEP. (2021, Dec). Retrieved from https://step.esa.int/main/snap-supported-plugins/sen2cor/

Sentinel-2 MSI User Guide. (n.d.). Retrieved from https://sentinel.esa.int/web/sentinel/user-guides/sentinel-2-msi

Shaoanlu. (2017). *SGD ¿ adam?? which One is the best optimizer: Dogs-Vs-Cats Toy Experiment.* Retrieved from https://shaoanlu.wordpress.com/2017/05/29/sgd-all-which-one-is-the-best-optimizer-dogs-vs-cats-toy-experiment/

Simonyan, K., & Zisserman, A. (2014, September). Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv e-prints*, arXiv:1409.1556.

Sumbul, G., Charfuelan, M., Demir, B., & Markl, V. (2019). Bigearthnet: A large-scale benchmark archive for remote sensing image understanding. In *Igarss 2019-2019 ieee international geoscience and remote sensing symposium* (pp. 5901–5904).

Talukdar, S., Singha, P., Mahato, S., Pal, S., Liou, Y.-A., & Rahman, A. (2020). Land-use land-cover classification by machine learning classifiers for satellite observations—a review. *Remote Sensing*, *12*(7), 1135.

Ulmas, P., & Liiv, I. (2020). Segmentation of satellite imagery using u-net models for land cover classification. *arXiv preprint arXiv:2003.02899*.

Urban Atlas. (2021, Jul). Retrieved from https://land.copernicus.eu/local/urban-atlas

Urban Atlas Mapping Guide v6.2. (2020). Retrieved from https://land.copernicus.eu/user-corner/technical-library/urban_atlas_2012_2018_mapping_guide

Varma, P. (2021). *Hands-on guide to implement batch normalization in Deep Learning Models.* Retrieved from https://analyticsindiamag.com/hands-on-guide-to-implement-batch-normalization-in-deep-learning-models/#:~:text=Batch%20Normalization%20is%20also%20a,of%20overfitting%20and%20works%20well

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., … He, Q. (2020). A comprehensive survey on transfer learning. *Proceedings of the IEEE*, *109*(1), 43–76.