UNIVERSITY OF TWENTE.

Faculty of Geo-information Science and Earth Observation

Minor Earth Observation

Automatic Delineation of Agricultural Fields from Sentinel-2 Images

Floor Stefess
Lars van der Velde
Laurens Laarhuis
Matthijs Horst
Max Resing
Paula Janeka
Simonas Budėjis

Supervisor:
Claudio Persello

University of Twente
Faculty ITC
P.O. Box 217
7500 AE Enschede
The Netherlands

Contents

1 Introduction

Satellite images are a popular choice of research data. It tends to be up-to-date, accurate, covers large areas and can be freely accessed using openly available data sets. However, processing and evaluating remotely sensed data is difficult. Researchers overcome this challenge by applying a powerful technique in the field of deep learning, such as convolutional neural networks.

This research describes the process of training a fully convolutional neural network (FCN) to extract the boundaries of agricultural fields based on satellite imagery. The remainder of this paper is organized as follows. Section 2 describes how satellite images of the Netherlands and highly accurate reference data is prepared in order to setup up an FCN. Afterwards, section 3 covers different implementations of convolutional neural networks that are considered as a candidate for the final choice of network. The section includes an accuracy assessment to compare the network's performances. Next, the results of the research is discussed in order to support the reader to assess the value of the results. The paper concludes by examining potential fields of application as well as further research.

2 Methodology

Google Colab and the Python programming language are used for the collaborative work on this assignment. A lot of Python libraries are available that support researchers in the field of data processing and machine learning. The code base makes use of tools and libraries such as TensorFlow [1], Keras [2] and NumPy [3]. TensorFlow and Keras are used to build, train and test neural networks. NumPy allows to efficiently process large data sets such as images.

2.1 Acquisition of data

Each group member was assigned a province for which they had to obtain Sentinel-2 images. These were downloaded from the Copernicus Open Access Hub, provided for free by the European Space Agency [4]. One of the qualitative requirements for the images was a low cloud coverage. In addition, it is important to have current imagery, preferably from October or November 2020. This is to keep land coverage consistent across multiple images. In order for the neural network to learn to predict boundaries, the training images needed accurate reference data, i.e. field boundaries. The reference data is available as the dataset Agrarisch Areaal Nederland (AAN) from the platform PDOK.nl [5]. The same platform provides the dataset Bestuurlijke Grenzen, which contains the borders of the provinces [6]. This is required, as the Sentinel-2 images are not shape-fitted to the provinces. Since both data sets are equipped with geotagged information, software such as ArcMap [7] can be used to align both data sets with each other.

Only the 10-meter resolution bands of the Sentinel-2 images are relevant in this research. Thus, band 2, 3, 4 and 8 are the only bands that are of consideration in this step. The data is prepared as follows. Firstly, the satellite images and the reference data of the field boundaries are cropped on the dimension of a Dutch province. Afterwards, the vector data of the AAN dataset has to be rasterized. This creates an additional layer with the same resolution as the underlying satellite data. The field boundaries have a width of 1 pixel. The new layer should be recoloured such that the field boundary pixels are white and the background pixels are black. Finally, the image should be exported in tiles of the size of 800 x 800 pixels. The exported files share a regular file name pattern, i.e. `[classified|original]_<province_name>_<tile_index>.TIF`.
In table 2.1 the amount of tiles per province that is used is shown.

Province	Number of Tiles
Flevoland	19
Friesland	24
Gelderland	3
Zeeland	32
Limburg	5
Overijssel	36
Zuid-Holland	3

Table 2.1: Tiles per province

Not all tiles from the satellite images and their associated AAN boundaries are used. Images that are partially in border regions of neighbouring countries, for which no reference data is available are removed. The

final dataset is displayed in figure 2.1.

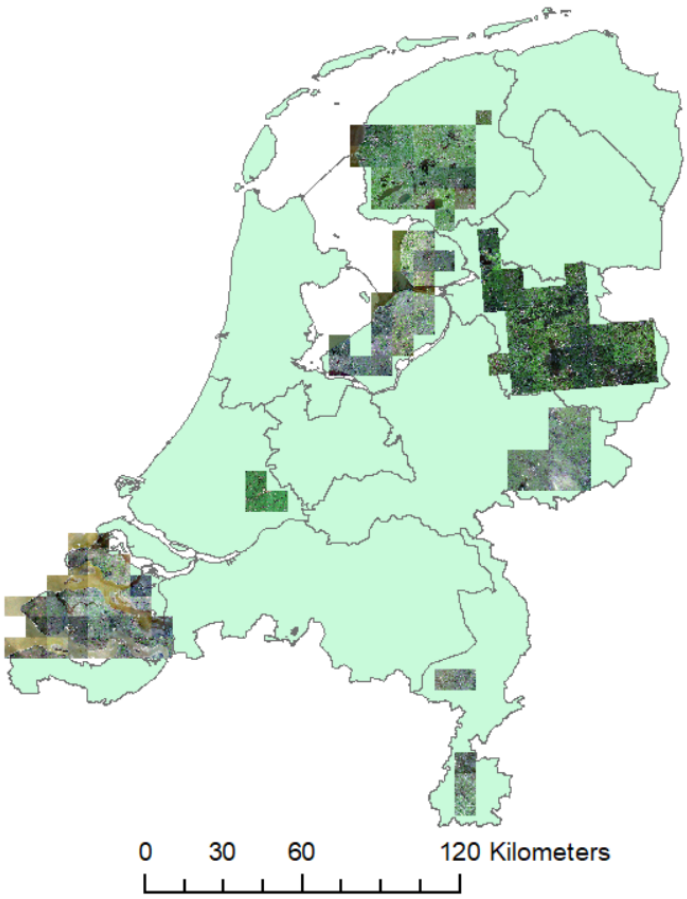


Figure 2.1: Overview of satellite image tiles used for training an FCN aligned on a schematic map of the Netherlands

2.2 Image representation

A popular tool for image processing with Python are NumPy data structures. NumPy is a Python library designed to efficiently create, process and modify data structures with uniform data types. The different tiles are imported file per file. First, the tiles of the satellite images are imported. The image information consists of four bands, thus, the image is mapped onto a 3-dimensional NumPy array. Dimension 1 and 2 are used to represent the row and column of a pixel in an image. Dimension 3 consists of four elements, where each element represents the numeric value of one band. The sentinel bands are assigned as follows:
`[ Band 2 - Blue, Band 3 - Green, Band 4 - Red, Band 8 - Near Infrared]`

The reference images are imported likewise with two differences: firstly, the innermost dimension of each reference image has just a single element, instead of 4. Secondly, the value of each pixel is altered. Formerly white is assigned to the numeric value 1, representing the class *Other*. Similarly, black is assigned to the numeric value 2, representing *Field boundary*.

Both image sets are stored in a dictionary. The key is generated based on the file names of each file. It extracts the province name and the tile index. Therefore, the resulting key is a 2-tuple. The dictionary of the satellite images and one of the reference counterparts share the same key set. This allows an easy option to query the satellite image and the associated labels from the data structures.

2.2.1 Normalization of satellite data

Normalization of data ensures that numeric data is transformed to a uniform scale. This process increases the accuracy and speed of convergence. Additionally, it helps to prevent the problem of vanishing or exploding gradient, which is common in neural networks. Therefore, all input images are normalized into pixel intensity values within the boundaries of the interval [0, 1]. Each band of the image data is normalized separately by finding the minimum and maximum value of the band. Furthermore, the satellite image data and the reference

data are visualised to verify the quality of imported data. The next step is to split the entire set of input data into two mutually exclusive sets for training and testing. This process is described in section 2.3.

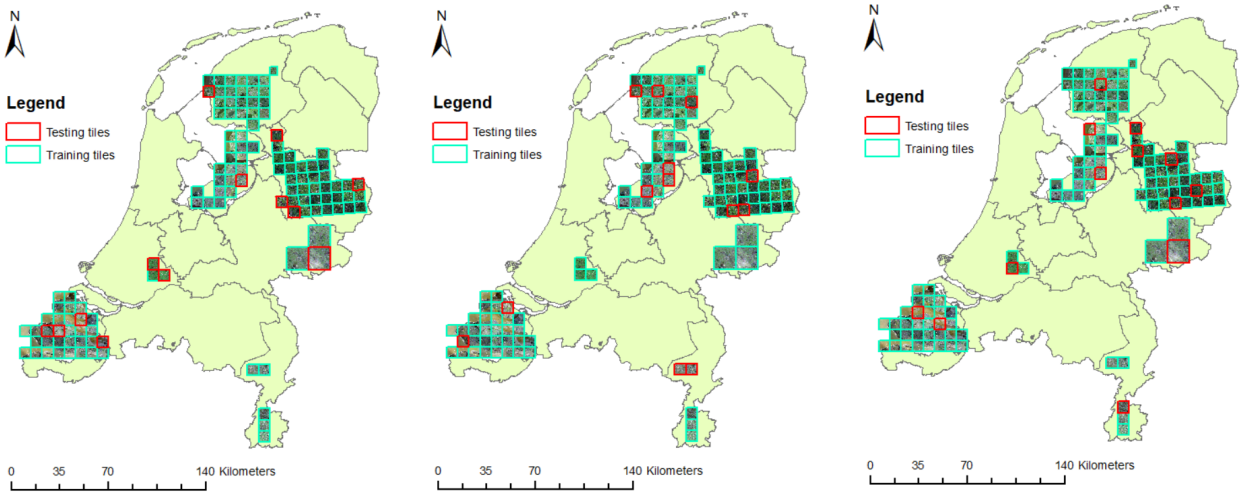
2.3 Preparing data

As not all fields in the Netherlands have similar patterns and structures it is important to have diverse training and test data. The images that are imported into the program are categorized based on their province name. Simply splitting the data into two halves would result in a lack of diversity for both of the data sets. Both data sets would not be representative, since they just include the satellite data of specific provinces. For example, the training data might not cover any of the provinces represented in the testing data and vice versa.

The input data is split into training and testing data by applying stratified random sampling [8]. As already explained in section 2.2, the input data set is organized in a dictionary-like data structure. The set of keys is used to split the input data set. Thus, a random number generator selects a random sample of the key set as a representative of the training data. The remaining keys in the key set are used to define the test data set. By applying this method, not only are two mutually exclusive subsets achieved, but diversity in both of these sets is also ensured. This means that the training, as well as the smaller test data sets, have representative tiles of data of multiple provinces. The overview of the proportion between training and testing tiles can be seen in the table 2.2 below. Figure 2.2 visualizes training and testing tiles of three different networks on three schematic maps of the Netherlands.

	FCN-DK5		FCN-DK6		U-Net3	
Province	Training	Testing	Training	Testing	Training	Testing
Flevoland	18	1	16	3	17	2
Friesland	23	1	21	3	23	1
Gelderland	2	1	3	0	2	1
Zeeland	28	4	30	2	30	2
Limburg	5	0	3	2	4	1
Overijssel	32	4	33	3	31	5
Zuid-Holland	1	2	3	0	2	1

Table 2.2: Training and testing tiles per province



(a) FCN-DK5 (b) FCN-DK6 (c) U-Net3

Figure 2.2: Training and testing tiles per province visualized

Furthermore, the random number generator is initialized with a seed, which ensures consistency of the training and test data split. Also, the relative size of the training set was controlled with a parameter. The described implementation allows to easily test and verify the quality of the two different subsets.

A fully convolutional neural network can be trained more efficiently if the patches that are fed to the network are small. The size of the patches depends on the depth of the convolutional neural network. The depth of a network is defined by the number of layers. The deeper a network the larger the patches have to be. First, to create the patches, each tile is split grid-wise into patches. Afterwards, the patches are prepared to train the FCN. This requires to restructure the reference data to a so-called "one-hot" encoding. The numeric value 1 or 2 will be replaced with an array that represents the defined labels. Label `1` is changed to `[1, 0]`. Similarly, label `2` is changed to `[0, 1]`. As one can see, the innermost dimension of each patch is altered to an array of size 2 - corresponding to the number of labels.

2.4 Training a network

Previously, it was explained how the training data is obtained. The training data set consists of the input data, the satellite images and the target data, i.e. the reference data. This section describes the process of training a particular network.

Before training a network, the training data is split into two sets. One set is used as the actual input to train the network, whereas the other set is excluded from the training. This separate set is solely used for validation. The validation is explained later in this section.

In general, training a convolutional neural network is done in iterations. Those iterations, called *epochs*, are a pass over the entire training data set. In one epoch, the training data is split into batches of a predefined size. Each time a batch is processed, the network adjusts the weights of the connections between the neurons. The networks optimize the weights based on the reference data. In principle, after each batch, the network should be able to predict the results of the batch data more accurately. The weights are adjusted with the help of an optimizer. In this research, the two common optimizers "SGD" and "Adam" have been tested.

After each epoch, the validation data set verifies the accuracy of the network. This is achieved by simulating a test run with the validation data set. Based on the input data, the network tries to predict the labels of each pixel. In conclusion, the prediction is compared to the reference data to estimate the network's accuracy for unknown input images.

2.4.1 Optimizer: Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an optimizer which only considers a single training point at a time and tries to move the training point to a function's minimum. It does this by updating the parameters in the opposite direction of the gradient of the objective function w.r.t. the parameters for each training example. The learning rate determines the size of the steps that approach the minimum. Momentum is a method that helps to accelerate SGD in the relevant direction and dampens the oscillations. It does this by adding a fraction of the update vector of the previous step to the current update vector.

2.4.2 Optimizer: Adam

Adam is another optimizer that is commonly used. This optimizer uses along with the momentum also an adaptive learning rate which influences the learning rate as the optimizer approaches the minimum. The learning rate is the same as in the previous SGD optimizer. Adam also has two terms for the momentum instead of one. The first momentum being β_1 and the second being β_2. These terms determine the speed of the decay of the momentum.

2.5 Hyper-parameter optimization

The training of a network is highly dependent on a variety of hyper-parameters. In this section, the approach to find optimal values for those hyper-parameters for different networks will be discussed.

The approach of finding these optimal values starts with determining which ones have significant impact on training an FCN. Since most hyper-parameters have optimal standard parameters, there are only two hyper-parameters which are chosen to be changed among the different network configurations. These are the batch size, the optimizer and the corresponding learning rate.

The batch size regulates how many samples are propagated to the network. The smaller the batch size, the more fluctuation in loss and accuracy values. However, the larger the batch size, the more memory needs to be allocated when training the network[9].

The learning rate changes the model based on the estimated error after the weights of the model are updated. When a small value is assigned, the model will be more accurate, but this goes paired with a longer training process. On the other hand, a large value may result in an unstable training process [10].

The number of epochs will be set to a small number so the training can be run smoothly and fairly fast. Since the differences in accuracy will not be clear enough if the number of epochs is too small it is chosen to have a number in between.

All these different hyper-parameters have to be tested on different networks to determine which network is the most accurate. This is done through a comparison of the loss function and accuracy graphs [11]. The aim is to have a loss function with a tendency to remain to decrease and an accuracy graph with a tendency to remain increasing. The networks which showed these results were then compared to see which network had the lowest loss values and the highest accuracy values.

For this, the validation data set becomes important. As long as the validation loss and accuracy follows the same trend as the training data, one can conclude that the network is still increasing in accuracy with each epoch. The training only overfits the network if the validation curves flip and their trends worsen.

2.6 Accuracy Assessment

After optimal hyper-parameters were found for the different networks, the training process was repeated. This training should maximize the prediction accuracy while not overfitting the network with the training data. To achieve this, the networks are trained with a considerably larger number of epochs. It is important that the resulting validation curves still do not show signs of an overfitted network. An accuracy assessment helps to understand how accurate a trained network can predict field boundaries.

The accuracy assessment uses the trained network to predict field boundaries of given input images. Here the network took the training images as input and tried to assign a class to each pixel. The resulting predictions are compared to the expected outcome, i.e. the reference data. A confusion matrix displays the true negatives, true positives, false negatives and false positives. The true positive and true negative values show respectively what pixels are correctly identified *field boundary* or *other*. A false positive is an *other* pixel identified as *field boundary*, false negative is the other way around. The Confusion Matrix also displays the sum, which is helpful for calculating recall and precision. An example Confusion Matrix is shown in table 2.3.

Using the values of this confusion matrix the overall accuracy precision, recall and F1-score is calculated. The precision is the percentage of correctly identified field boundaries with respect to all the identified field boundaries (true and false). The recall is the percentage of correctly identified field boundaries with respect to the correctly identified field boundaries and the falsely identified *other* class. See equations 2.1, 2.2 and 2.3.

The F1-score of the network is based on the accuracy of the predictions made of all the test images fed into it. This F1-score is based on the values for precision and recall. It shows the weighted accuracy, which is necessary for a good comparison of the accuracy of the networks. This is because the classified images contain a large number of black pixels, making these more influential in accuracy determination. This means that if the whole predicted image would be black, the accuracy would still be high, as all black pixels are identified correctly. Therefore the F1-score a better measurement, to create a weighted score in which the ratio of black and white pixels is taken into account.

	Actual Other	**Actual Field Boundary**	**Sum**
Prediction Other	True Negative	False Negative	TN + FN
Prediction Field Boundary	False Positive	True Positive	FP + TP
Sum	TN + FP	FN + TP	

Table 2.3: Confusion Matrix standard form

$$\text{Precision} = \frac{TP}{FP + TP} \quad (2.1) \qquad \text{Recall} = \frac{TP}{FN + TP} \quad (2.2) \qquad \text{F1} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.3)$$

3 Results

The following section of this report addresses the steps taken to get from multiple potential networks and hyper-parameters to one final network which has the most optimal set of weights between its neuron layers. Such a network is able to precisely predict the field-boundaries based on a satellite image.

3.1 Choice of networks

To establish a properly functioning algorithm, a choice needs to be made which fully convolutional neural network with which hyper-parameters are best at fulfilling this purpose. This is done by taking multiple networks with different configurations to see which combination results in the most accurate predictions.

One option is the FCN-DK network [12], as this is suitable for deep learning purposes similar to the current field boundary delineation. Initially, the network was designed to detect informal settlements on satellite images. However, this is an FCN that aims to process satellite images by design. Since an FCN-DK network is compact and has a small performance footprint it is an optimal candidate for this assignment.

The original network was designed to work with six layers. However, the number of layers can deviate. A general rule is, that more layers tend to result in more accurate predictions. Based on this, the decision is made to choose 4, 5 and 6 layers, to achieve a variety of results which predict field boundaries at different levels of accuracy.

A counterpart to the FCN-DK network is the widely used U-Net. The implementation of the U-Net network is well-known for considering a large number of features of the input data. Although U-Net is initially designed to serve in the biomedical domain [13] there also exist derivatives for satellite image processing [14]. These derivatives are especially interesting for our research, as there exists a Keras package suitable for the field boundary delineation research [15]. As each layer contains filters that consider a large number of features, the network is quite heavy to run. This means that fewer layers can be used in Google Colab if compared to the FCN-DK network. See section 4.3.

Besides choosing the network, there are hyper-parameter values that need to be determined. For each FCN-DK network, these are the optimizer, batch size and learning rate. For the U-Net network, different configurations are tested as well. Since Adam is self-adapting it is the better choice for U-Net.

Restrictions to the memory usage of Google Colab led to limited options in batch sizes. As larger the batch size, as more memory is required. Test are done with batch sizes of 32, 64, 128 for FCN-DK networks and 8 for U-Net. These values are chosen as they are a compromise between fluctuation in loss and accuracy values and the allocated memory at run-time.

For the learning rate the values of 0.1, 0.01, 0.001, 0.0001 have been chosen for FCN-DK networks and 0.01, 0.001, 0.0015, 0.0001 for U-Net. These values are chosen as they seem to be a good trade-off between a short training process and accuracy.

For U-Net network, the epsilon values of 1e-5, 1e-6, 1e-7, 1e-8 are selected. This decision to tune the epsilon value is made based on Tensorflow's notes [16]. In the end, the value of 1e-6 is chosen as the best one for U-Net. The epsilon value makes sure you don't divide by zero.

For each run of FCN-DK networks, the constant amount of 200 epochs was used. For the U-Net network, only 10 epochs were used and later the best configuration was run for 50 epochs to ensure the least amount of fluctuation in loss and accuracy. This constant value is chosen as this seems to be a good trade-off between training time and accuracy.

Based on all the hyper-parameters and network options, a set of networks with a specific configuration for parameters is established. The table 3.1 portrays which networks are tested with which configuration to determine optimal training parameters.

3.2 Determine optimal hyper-parameters

The different configurations of Table 3.1 are all trained with the same number of epochs and the same input data set. The training data consists of patches of the tiles. The patch size depends on the corresponding network (compare Table 3.2). Before starting the training, a validation set is chosen. For each training run, 5% of the

Network	Batch size	Optimizer	Learning rate
FCN-DK4	32, 64	Adam	0.1 0.01 0.001
FCN-DK5	32, 64	SGD	0.1 0.01 0.001
FCN-DK5	32, 64, 128	Adam	0.1 0.01 0.001 0.0001
FCN-DK6	32, 64	SGD	0.1 0.01 0.001
FCN-DK6	32, 64, 128	Adam	0.1 0.01 0.001 0.0001
U-Net3	8	Adam	0.01 0.001 0.0015 0.0001

Table 3.1: Networks and corresponding hyper-parameters

patches are excluded from the training to function as validation data.

Network	Patch Size
FCN-DK5	50 x 50 px
FCN-DK6	80 x 80 px
U-Net3	400 x 400 px

Table 3.2: Overview of the training patch sizes for different networks

These patch sizes are different for each network because the more layers a network has, the larger the patch sizes have to be. This is because each layer adds more features to the network and therefore it needs to be fed more pixels to train. U-Net needs even more pixels since it has a large amount of features when compared to FCN-DK 5 or 6.

After each epoch, the network's accuracy and loss are calculated first for the training set but more importantly, also for the validation set. The results were plotted with the number of epochs on the x-axis and with the accuracy and the loss on the y-axis. The plots are shown in the figures 3.1 - 3.4.

To determine which hyper-parameters are best, the plots are judged based on the trend of validation accuracy and loss. The validation scores are assessed in relation to the training accuracy and loss. Furthermore, the curve's fluctuation of the validation accuracy should be minimal. The plot analysis helped to determine the optimal hyper-parameters for each network. Table 3.3 presents these results. As described in the previous section, these networks are chosen to be trained with 600 epochs to maximize a network's accuracy.

	Batch Size	Optimizer	Learn. Rate	Val. Acc.	Tr. Acc.	Val. Loss	Tr. Loss
FCN-DK5	32	Adam	0.001	91.2%	89.6%	20.0%	23.8%
FCN-DK6	64	Adam	0.01	88.5%	88.5%	26.5%	23.5%
FCN-DK6	128	Adam	0.01	88.7%	89.6%	24.0%	23.8%
U-Net3	8	Adam	0.0015	87.8%	89.9%	25.4%	22.4%

Table 3.3: Results of plot analysis on optimal hyper-parameters

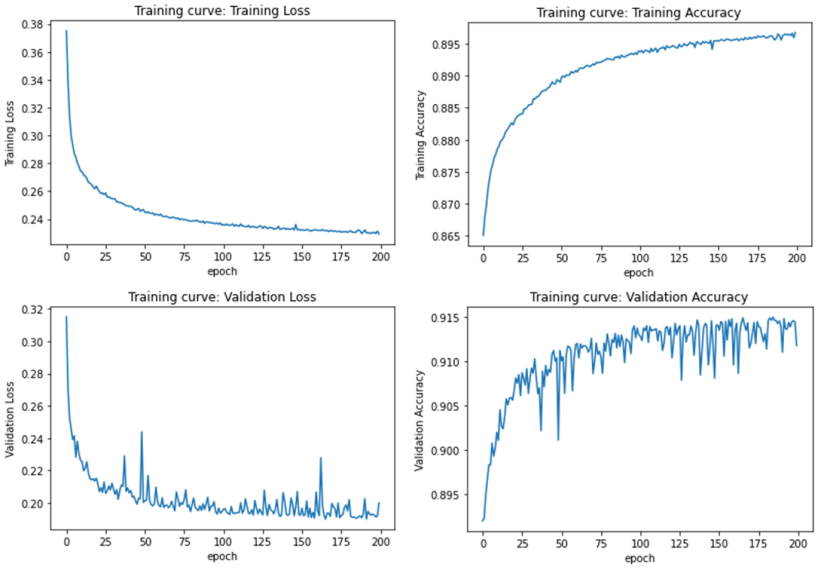


Figure 3.1: FCN-DK, 5 layers, Batch size 32, Adam optimizer & learning rate 0.01

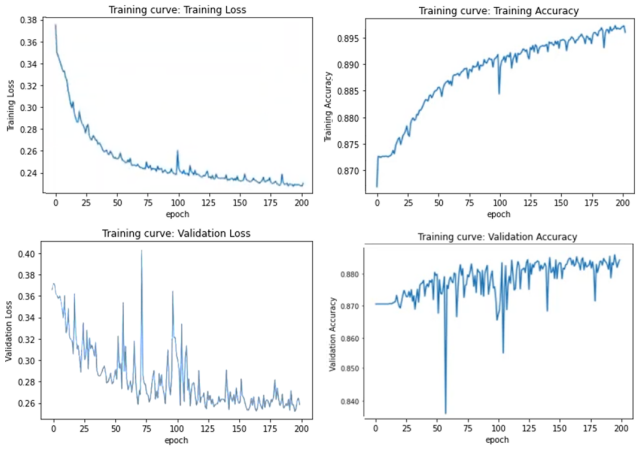


Figure 3.2: FCN-DK, 6 layers, Batch size 64, Adam optimizer & learning rate 0.001

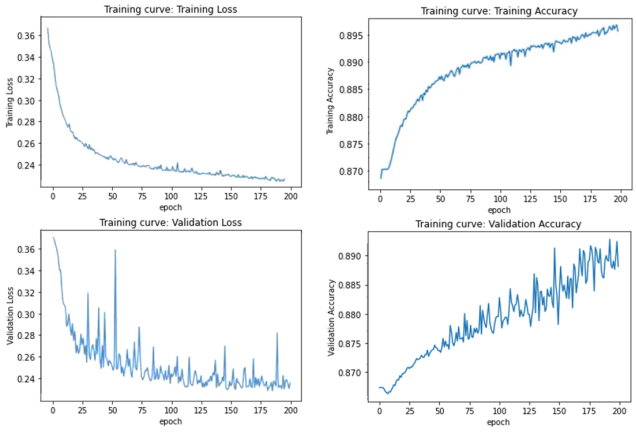


Figure 3.3: FCN-DK, 6 layers, Batch size 128 Adam optimizer & learning rate 0.001

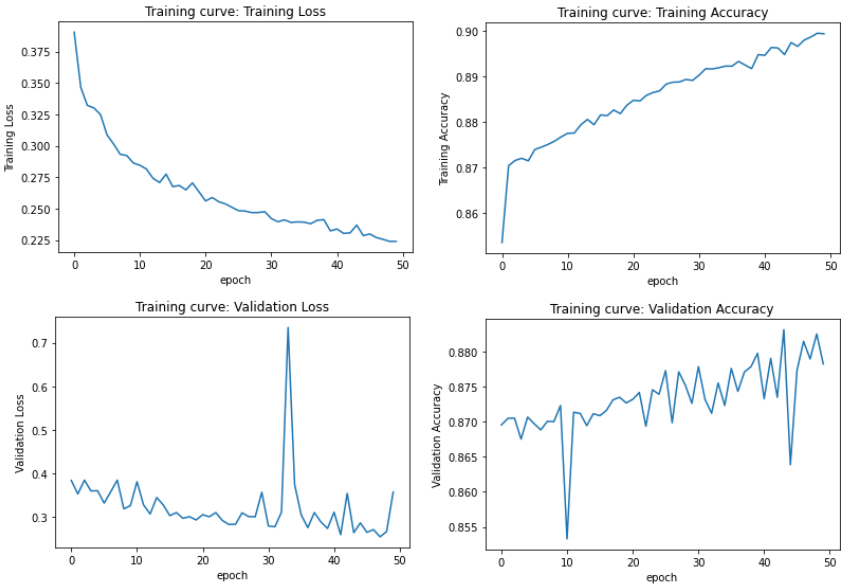


Figure 3.4: U-Net, 3 layers, Batch size 8 Adam optimizer, learning rate 0.0015 & epsilon 1e-06

Network	Batch size	Optimizer	Learning rate
FCN-DK5	32	Adam	0.001
FCN-DK6	96	Adam	0.01
U-Net3	8	Adam	0.0015

Table 3.4: Choice of networks and hyper-parameters for advanced training run

As the results for the FCN-DK 6 were similar for the batch sizes of 64 and 128, it is decided to train the network FCN-DK with a batch size of 96. To improve the network, as already mentioned, a significantly larger number of epochs is used. Instead of 200 epochs, the chosen networks are trained with 600 epochs. After the training, the validation curves were interpreted and it is concluded that the networks are not over-fitted. The following results were obtained by training the networks with 600 epochs (Figures 3.5 and 3.6).

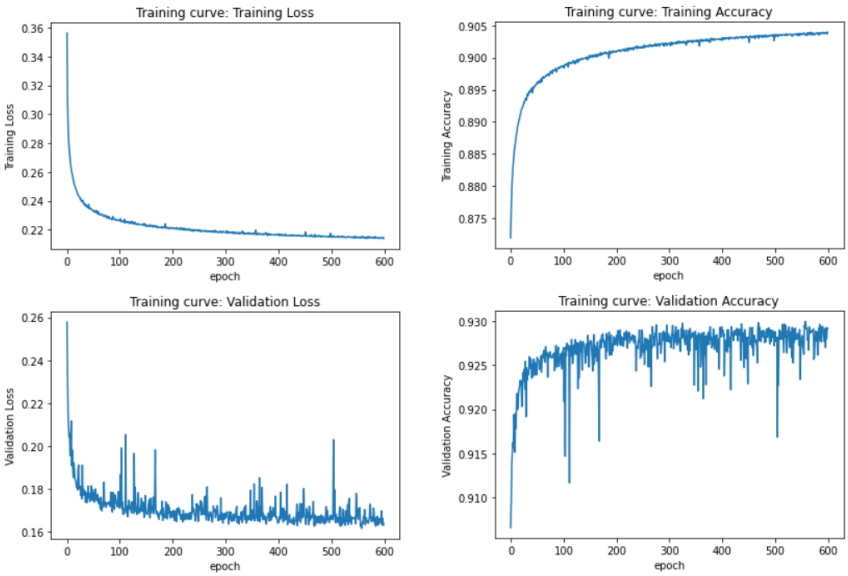


Figure 3.5: FCN-DK5 using 600 epochs with batch size 32

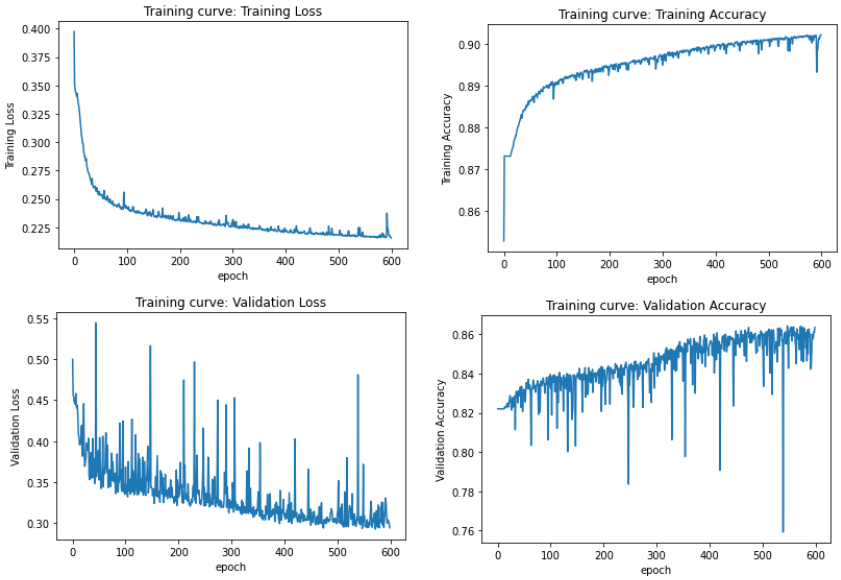


Figure 3.6: FCN-DK6 using 600 epochs with batch size 96

Contrary to the FCN-DK networks, U-Net3 network figures, mainly the validation accuracy figure, show slight overfit starting from the 200th epoch. The validation loss does not show a downwards trend similar to the training loss which indicates overfit as well.

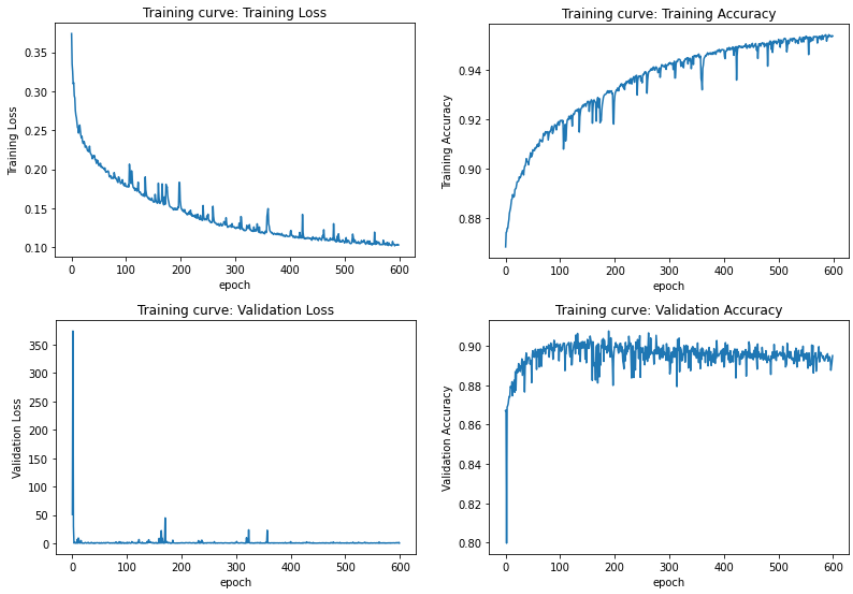


Figure 3.7: U-Net3 using 600 epochs with batch size 8

3.3 Accuracy Assessment

To show whether results of the used network are promising, the "Most accurate" and "Least accurate" results of the field boundary predictions are portrayed. Their respective confusion matrix and accuracy assessment have been carried out and are shown as well.

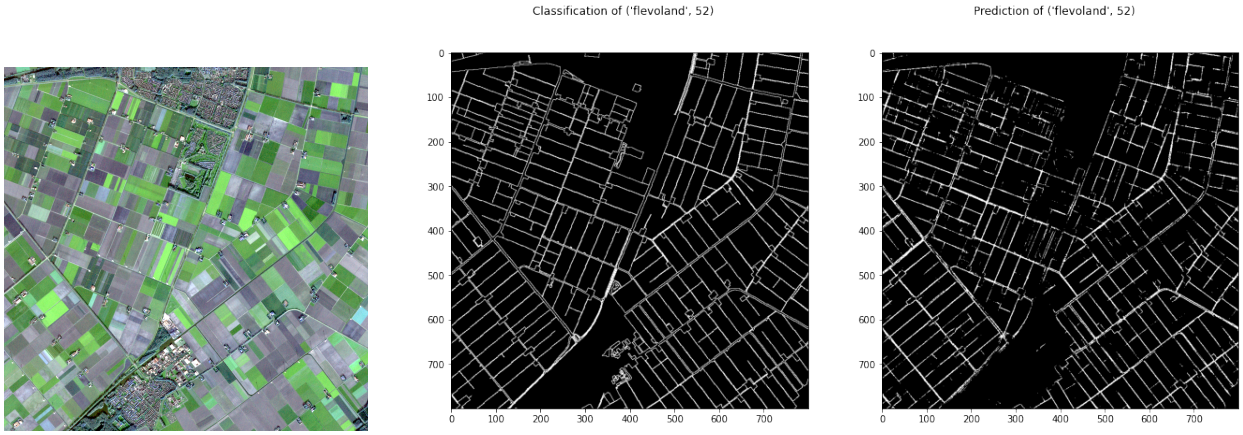
The terms "Most accurate" and "Least accurate" are, in this case, based on the total accuracy results that are obtained. This means that the images and corresponding reference and prediction that are portrayed show relatively more accurate or less accurate results compared to the cumulative accuracy. By observing the predicted images, field boundaries can be more accurately predicted based on external factors such as linearity of the boundaries and the size of the parts in between boundaries.

Usually, the overall accuracy is around 80% to 90%. This is because the predominant class is the class *other*. The network will therefore classify the vast majority as *other*. This results in high overall accuracy. Therefore an F1-score is more meaningful. When considering the extreme imbalanced number of pixels of the two classes *field boundary* and *other* it becomes clear that the F1-score will deviate from the overall accuracy. This means that an F1-score of 50% or higher is considered a good score.

In the accuracy assessment of each network, a representative accurate and inaccurate sample is depicted. Furthermore, the confusion matrices and accuracy scores for the cumulated assessments are included to give an overview of the total result.

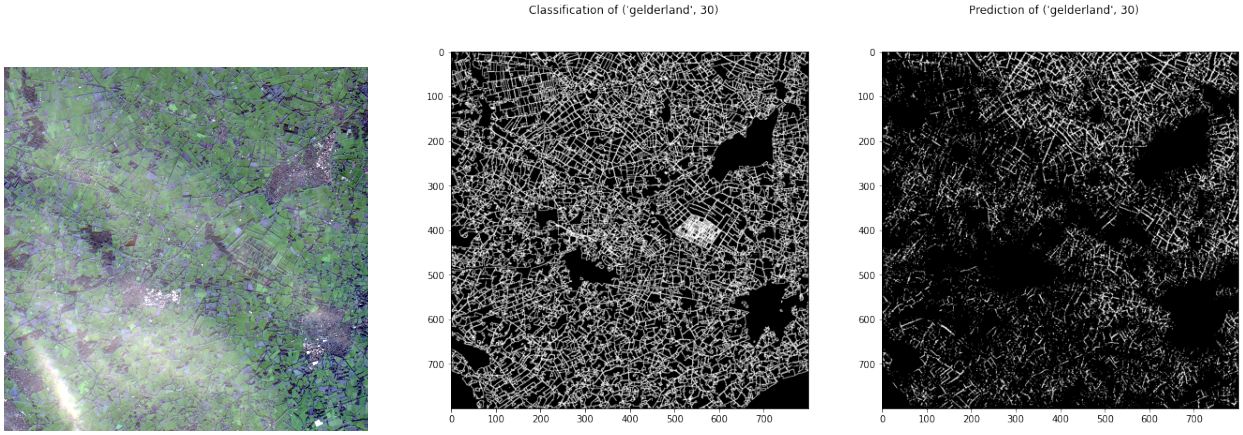
3.3.1 Accuracy of network FCN-DK5

From the data of the pre-trained FCN-DK5 with 600 epochs, the overall accuracy is at 86% with an F1-score of 49%. A particularly accurate result can be found in Figure 3.8 (a sample of Flevoland) and a less accurate result in Figure 3.9 (a sample of Gelderland). Table 3.5 and Table 3.7 show the confusion matrix and the accuracy scores corresponding to the most accurate prediction. Table 3.6 and Table 3.8 portray these results for the least accurate prediction. The cumulative confusion matrix and accuracy score can be found in Table 3.9 and Table 3.10 respectively.



(a) Original (b) Reference (c) Prediction

Figure 3.8: Most accurate prediction of field boundaries (Flevoland)



(a) Original (b) Reference (c) Prediction

Figure 3.9: Least accurate prediction of field boundaries (Gelderland)

	Actual Other	**Actual Field Boundary**	**Sum**
Prediction Other	559696 (87.452%)	25859 (4.04%)	585555
Prediction Field Boundary	18122 (2.832%)	36323 (5.675%)	54445
Sum	577818	62182	

Table 3.5: Confusion Matrix Flevoland predicted with FCN-DK5

	Actual Other	**Actual Field Boundary**	**Sum**
Prediction Other	424035 (66.255%)	141444 (22.101%)	565479
Prediction Field Boundary	29572 (4.621%)	44949 (7.023%)	74521
Sum	453607	186393	

Table 3.6: Confusion Matrix Gelderland predicted with FCN-DK5

Outcome	Percentage
Overall accuracy	93.128 %
Precision	66.715 %
Recall	58.414 %
F1-score	62.289 %

Table 3.7: Accuracy assessment Flevoland (FCN-DK5)

Outcome	Percentage
Overall accuracy	73.279 %
Precision	60.317 %
Recall	24.115 %
F1-score	34.455 %

Table 3.8: Accuracy assessment Gelderland (FCN-DK5)

	Actual Other	Actual Field Boundary	Sum
Prediction Other	6597089 (79.292%)	838738 (10.081%)	7435827
Prediction Field Boundary	320954 (3.858%)	563219 (6.769%)	884173
Sum	6918043	1401957	

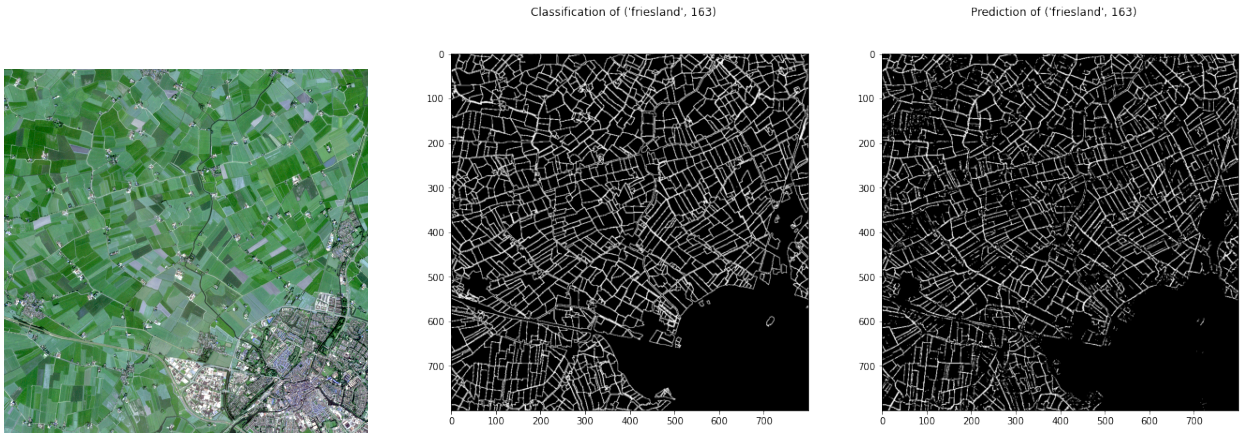
Table 3.9: Confusion Matrix cumulated predicted with FCN-DK5

Outcome	Percentage
Overall accuracy	86.061 %
Precision	63.7 %
Recall	40.174 %
F1-score	49.273 %

Table 3.10: Accuracy assessment cumulated (FCN-DK5)

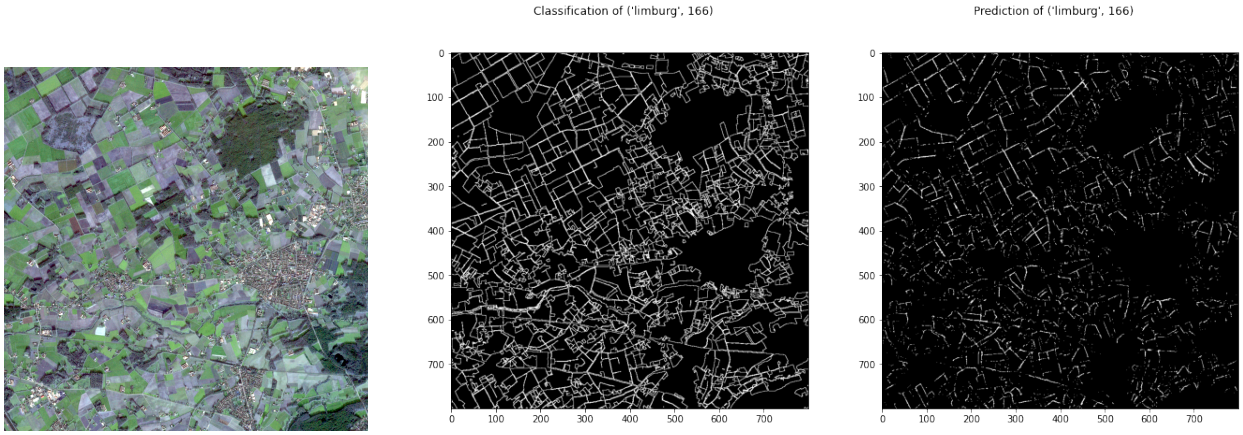
3.3.2 Accuracy of network FCN-DK6

The network FCN-DK6, which is trained with 600 epochs gave similar results in comparison to the FCN-DK5. The overall accuracy is at 90% and the F1-score at 50%. Figure 3.10, a sample of Friesland, presents an accurately predicted image of the test data set, whereas figure 3.11, a sample of Limburg covers a considerable inaccurate prediction. The corresponding confusion matrices in 3.11, 3.12 and accuracy scores in table 3.13 and 3.14. The cumulative confusion matrices and the accuracy scores are displayed in tables 3.15 and 3.16.



(a) Original (b) Reference (c) Prediction

Figure 3.10: Most accurate prediction of field boundaries (Friesland)



(a) Original (b) Reference (c) Prediction

Figure 3.11: Least accurate prediction of field boundaries (Limburg)

	Actual Other	**Actual Field Boundary**	**Sum**
Prediction Other	507276 (79.262%)	47072 (7.355%)	554348
Prediction Field Boundary	27530 (4.302%)	58122 (9.082%)	85652
Sum	534806	105194	

Table 3.11: Confusion Matrix Friesland predicted with FCN-DK6

	Actual Other	**Actual Field Boundary**	**Sum**
Prediction Other	536072 (83.761%)	78307 (12.235%)	614379
Prediction Field Boundary	10764 (1.682%)	14857 (2.321%)	25621
Sum	546836	93164	

Table 3.12: Confusion Matrix Limburg predicted with FCN-DK6

Outcome	**Percentage**
Overall accuracy	88.343 %
Precision	67.858 %
Recall	55.252 %
F1-score	60.91 %

Table 3.13: Accuracy assessment Friesland (FCN-DK6)

Outcome	**Percentage**
Overall accuracy	86.083 %
Precision	57.988 %
Recall	15.947 %
F1-score	25.015 %

Table 3.14: Accuracy assessment Limburg (FCN-DK6)

	Actual Other	**Actual Field Boundary**	**Sum**
Prediction Other	7074523 (85.03%)	599480 (7.205%)	7674003
Prediction Field Boundary	230343 (2.769%)	415654 (4.996%)	645997
Sum	7304866	1015134	

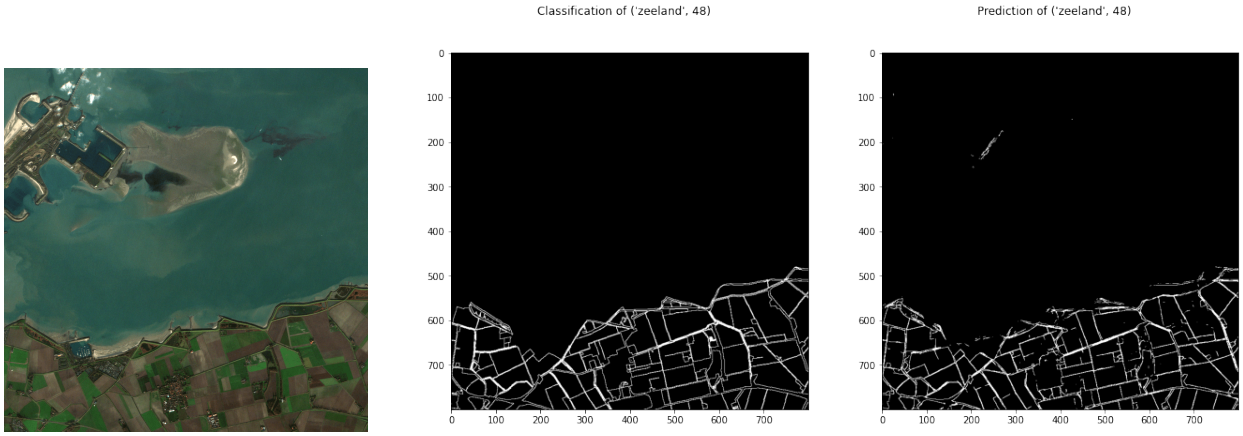
Table 3.15: Confusion Matrix cumulated predicted with FCN-DK6

Outcome	**Percentage**
Overall accuracy	90.026 %
Precision	64.343 %
Recall	40.946 %
F1-score	50.045 %

Table 3.16: Accuracy assessment cumulated (FCN-DK6)

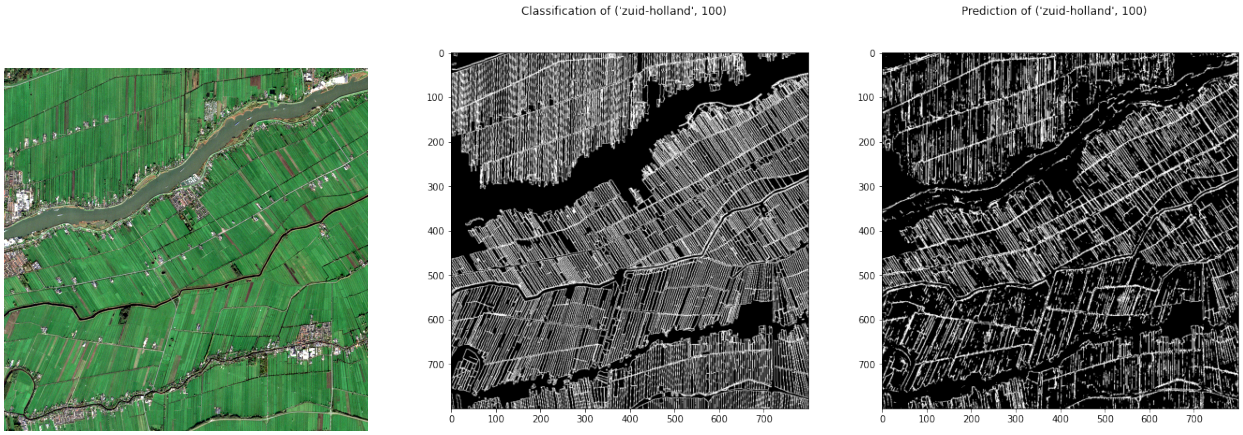
3.3.3 Accuracy of network U-Net3

The third network, UNet-3, which is also trained with 600 epochs gave similar results to the previously discussed two networks. The overall accuracy is at 88%, but the the F1-score is slightly better at almost 58%. Figure 3.12, a sample of Zeeland, shows an accurately predicted image of the test data set. In contrast, figure 3.13, a sample of Zuid-Holland shows a less accurate prediction. The corresponding confusion matrices are in tables 3.17 and 3.18, accuracy scores in tables 3.19 and 3.20. The cumulative confusion matrices and the accuracy scores are displayed in tables 3.21 and 3.22, respectively.



(a) Original (b) Reference (c) Prediction

Figure 3.12: Most accurate prediction of field boundaries (Zeeland)



(a) Original (b) Reference (c) Prediction

Figure 3.13: Least accurate prediction of field boundaries (Zuid-Holland)

	Actual Other	**Actual Field Boundary**	**Sum**
Prediction Other	607850 (94.977%)	9881 (1.544%)	617731
Prediction Field Boundary	8604 (1.344%)	13665 (2.135%)	22269
Sum	616454	23546	

Table 3.17: Confusion Matrix Zeeland predicted with U-Net3

	Actual Other	**Actual Field Boundary**	**Sum**
Prediction Other	384862 (60.135%)	116634 (18.224%)	501496
Prediction Field Boundary	42539 (6.647%)	95965 (14.995%)	138504
Sum	427401	212599	

Table 3.18: Confusion Matrix Zuid-Holland predicted with U-Net3

Outcome	Percentage
Overall accuracy	97.112 %
Precision	61.363 %
Recall	58.035 %
F1-score	59.653 %

Table 3.19: Accuracy assessment Zeeland (U-Net3)

Outcome	Percentage
Overall accuracy	75.129 %
Precision	69.287 %
Recall	45.139 %
F1-score	54.665 %

Table 3.20: Accuracy assessment Zuid-Holland (U-Net3)

	Actual Other	Actual Field Boundary	Sum
Prediction Other	6652752 (79.961%)	580514 (6.977%)	7233266
Prediction Field Boundary	406095 (4.881%)	680639 (8.181%)	1086734
Sum	7058847	1261153	

Table 3.21: Confusion Matrix cumulated predicted with U-Net3

Outcome	Percentage
Overall accuracy	88.142 %
Precision	62.632 %
Recall	53.970 %
F1-score	57.979 %

Table 3.22: Accuracy assessment cumulated (U-Net3)

4 Discussion

In here the results of the previous section are discussed and the quality of those results are analysed. In section 4.1, it is analysed how precise the predictions are that were made before starting this project. In section 4.1.2 is looked upon and argued about how well the F1-scores are. In 4.2 there is discussed what would happen if each province had an even number of input data tiles. Finally in section 4.3 each network is compared with each other and discussed which network is the best.

4.1 Accurate and inaccurate predictions

At the beginning of this assignment, some predictions were made about how precise the network can predict each type of landscape. Expected was that structured fields would perform well. The accuracy assessment confirms these expectations. An additional positive discoveries is that areas on the coastlines are predicted accurately, despite the large sea area in an image. Previously, we expected inaccurate results, because image tiles of the coastline are rare and the network is mostly trained with inland images.

Furthermore, we expected irregular and unstructured field boundaries to be a challenge. We could not identify significant difficulties that the network has with those input images. Nevertheless, the accuracy of those regions is slightly worse.

4.1.1 Analysis of predictions

From the results, it can be seen that there exist more accurate and less accurate predictions. When observing these predictions, one can identify different causes have an impact on the accuracy.

On the one hand, in Figure 3.8.a, the fields are places in a structured way across the area and they have a nearly equal size when compared to each other. The road network in between the fields represent field borders and are hereby a visual detection point for field boundaries. Aside from the structure, there is a little urban area and the fields have a clear contrast, making it easy to differentiate the fields from each other, resulting in an accurate prediction.

On the other hand, Figure 3.9 shows a remarkably less accurate prediction than Figure 3.8. This can be explained through the presence of clouds in the original image. The prediction of the right top corner of the image is comparably accurate. However, the overall prediction is negatively influenced by the haze of cloud coverage. Even though the fields might still be visible, it becomes difficult for the algorithm to find the field boundaries through the lack of contrast created by the haze.

Another example can be found in Figure 3.10, where the predicted field boundaries are quite similar to the reference image. This can be explained through the clarity of the original image. There is only a little amount of non-fields present among the class of *other* and the fields are of a similar size.

When observing Figure 3.11, the distinctness of the original image is noticeably lower. One reason to explain this is the lack of contrast between fields and non-fields, as well as the inconsistent size of the fields. Besides that, there are forests and cities present, scattered among the area, making it more difficult for the algorithm to identify this land coverage to be a field boundary or part of the non-field classification. This results in a near-black prediction image in which a little amount of field boundaries is detected.

4.1.2 F1-scores

The F1-scores for the networks show the weighted result of the accuracy with respect to the amount of black and white pixels. Considering F1-scores, the highest cumulative score of 58%, and thus the most accurate network, is obtained by the U-Net network. This can be considered as a good F1-score, as the results obtained by the network show relatively accurate predictions. This shows that the U-Net is currently better at identifying field boundaries in comparison to the networks FCN-DK5(49%) and FCN-DK6(50%).

Yet, it is still difficult to ensure this score is a good representation of the network. As the two different classes are represented unequally in each of the patches, it might still be that the amount of black pixels is not weighed down properly, resulting in a less accurate F1-score.

In general, the F1-score is reliable, but it is still an abstract measure of the total accuracy of the network. Especially with having a value difference of 1% between both the FCN-DK5 and FCN-DK6 network, it makes it hard to draw conclusions about which is the better network purely on this score.

4.2 Quality of Training and Reference data

The input data set consists of 7 provinces. Each province has a different number of input tiles. Comparing the results of the accuracy assessment with the availability of training and reference data, we can conclude the following observations.

The imprecise predictions belong to provinces of Limburg and Gelderland. In the case of Limburg, only 5 tiles of training data are provided, whereas Gelderland just provides 3. When putting this into relation with Friesland (24 tiles) and Flevoland (19 tiles), one can suspect that the network's performance can be improved by providing more evenly distributed input data.

Furthermore, you can see different overall structures of the agricultural regions in a province. For instance, Flevoland and Friesland are both sparsely populated regions with a large proportion of agricultural land use. In these regions, the predictions are similar to the reference data. In comparison, some input tiles cover an area with exceptional land coverage. An example would be a tile of a satellite image that is in the coastal region and mostly covered by the sea. Also, input data of Zuid-Holland deviates from regular agricultural fields. This region is characterized by tulip fields which tend to be narrow and small.

To sum it up, the training data set is a decent representation of regular agricultural regions in densely populated countries of western Europe. Nonetheless, the training data set has potential for improvements. Additional images of unique and uncommon agricultural regions that stand in contrast to regular agricultural regions of the Netherlands can help to increase the quality of the training data.

It could also be an option to include thicker field boundaries as this improves the true positive matches with field boundaries. In this case, the reference data would have thicker lines which gives the network more room to identify field boundaries. This can be an option to improve the accuracy of the field boundary delineation using deep learning.

Finally, we would like to add one last remark to the quality of the input data. At the end of the assignment, we discovered that the tiles of Gelderland are rendered with different parameters than the rest of the input data. In particular, a single pixel in a Gelderland tile has a resolution of 20 meters, whereas all other tiles have a 10 meter resolution per pixel. Consequently, a tile of Gelderland covers the area of 16 x 16km instead of the usual 8 x 8km. Just the tiles of the province Gelderland are affected, which sums up to a total of only three tiles. Apart from the resolution, we do not expect distortions, since the selection of bands remains with band

2, 3, 4 and 8. We expect a lower accuracy for Gelderland tiles, since the resolution is lower than in other tiles.

4.3 Comparison of different networks

As can be seen in the results from section 3.3, all three networks show highly accurate results. Not only for data with solely field boundaries, but also for images that covers additional types of landscapes. For example, when looking at figure 3.12a in section 3.3.3, the image contains about 50% sea and about 30% field boundaries. Nonetheless, image 3.12c seems to not be affected by this and gives an overall accuracy of approximately 97%. The coastal landscape does not disturb the FCNs predictions of field boundaries. The results are still considerably accurate.

When looking at the F1 score for all three networks, U-Net3 differs by approximately 8% compared to the two FCN-DKs. One of the reasons is, that the U-Net network just consists of 3 layers. On the one hand, this means, training the network can be done efficiently on less complex features. More complex features can not be considered with only a few layers. On the other hand, less complex features are also trained faster. Thus, it can be an indicator that the FCN-DK5 and 6 network requires more training to optimize on the more complex features.

From this observation, it can be interpreted that the FCN-DK networks are trained less sufficiently compared to the U-Net. Furthermore, the result indicates a good quality for all three networks, especially the U-Net3. To conclude, the currently most precise network of this research is U-Net3.

5 Conclusion

The deep learning networks used in this project were capable of detecting the boundaries of diverse agricultural fields. The results are not perfect but there is potential for improvement with Sentinel-2 images beyond our limited dataset.

The final conclusion we came to is that U-Net3 is the network best suited for field boundary delineation. It provides accurate results with minimal training effort. However, looking at the results for FCN-DK6, this seems to be a better option if one is willing to re-train this data using a high amount of epochs (i.e. 2000). The advantages of FCN-DK6 are cheap training cost due to low computational demands while retain a large amount of layers. This means, also complex features in an image are considered and therefore, one can expect a better accuracy. This network was designed from day one to process satellite images, compared to the biomedical background of U-Net3. Besides, the time to train the U-Net3 network largely exceeds the time needed to train the FCN-DK6 network, making FCN-DK6 the more favourable option.

As stated, an optimal training was not possible for FCN-DK6 due to resource constraints, resulting in room for improvement. On the one hand, the point where the validation accuracy starts dropping whilst the training accuracy still rises was not reached, even after 600 epochs. On the other hand, one can observe a slight overfit in U-Net3 after 200 epochs (compare figure 3.7). This observation was discussed in section 3.2. This means that FCN-DK6 can be further optimized before overfitting occurs, whereas no similar improvement can be applied to U-Net3. Unfortunately the limit of Google Colab was already reached. This behaviour can be further investigated in further research.

Another topic for further inquiries is comparing the efficacy of other types of networks like SegNet and Vnet in predicting field boundaries. It is wise to adjust these networks and tweaking FCN-DK6 to better suit this task. This would allow for a more definitive judgement on which network type is actually best for the job. Adding agricultural fields from different parts of the world can show if these networks can find universal rules for boundaries or not. Finally, adding in more classes to identify not only boundaries but also use cases will make more practical applications possible.

One use of this technology is administrative aid to governments in keeping their land maps up to date, especially in places were division of land is informally acquired. For research purposes, classifying land use types can reveal the potential of a nations food production. Furthermore, infrastructure planning can be supported by the resulting network of this assignment to minimize the footprint on agriculture.

References

[1] Tensorflow community. *Tensorflow*. URL: https://www.tensorflow.org/. (accessed: 21.01.2020).

[2] Francois Chollet. *Keras*. URL: https://keras.io/. (accessed: 21.01.2020).

[3] Numpy council. *Numpy*. URL: https://numpy.org/. (accessed: 21.01.2020).

[4] Serco. *Copernicus Open Access Hub (2014)*. URL: https://scihub.copernicus.eu/dhus/#/home#%5C%2Fhome. (accessed: 21.01.2020).

[5] Ministry of economics and environment. *Dataset: Agrarisch Areaal Nederland (AAN)*. URL: https://www.pdok.nl/introductie/-/article/agrarisch-areaal-nederland-aan-. (accessed: 21.01.2020).

[6] Kadaster. *Dataset: Bestuurlijke Grenzen*. URL: https://www.pdok.nl/introductie/-/article/bestuurlijke-grenzen. (accessed: 21.01.2020).

[7] Esri Nederland. *ArcGIS-platform*. URL: https://www.esri.nl/nl-nl/producten/arcgis-platform/home. (accessed: 21.01.2020).

[8] J.E. Trost. "Statistically nonrepresentative stratified sampling: A sampling technique for qualitative studies." In: *Qual Sociol* 9.10 (1986), pp. 54–57. DOI: https://doi.org/10.1007/BF00988249.

[9] Jason Brownlee. *How to Control the Stability of Training Neural Networks With the Batch Size*. URL: https://machinelearningmastery.com/how-to-control-the-speed-and-stability-of-training-neural-networks-with-gradient-descent-batch-size/. (accessed: 20.01.2020).

[10] Jason Brownlee. *Understand the Impact of Learning Rate on Neural Network Performance*. URL: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/. (accessed: 20.01.2020).

[11] David Mack. *How to pick the best learning rate for your machine learning project*. URL: https://medium.com/octavian-ai/which-optimizer-and-learning-rate-should-i-use-for-deep-learning-5acb418f9b2. (accessed: 20.01.2020).

[12] C. Persello and A. Stein. "Deep Fully Convolutional Networks for the Detection of Informal Settlements in VHR Images". In: *IEEE Geoscience and Remote Sensing Letters* 14.12 (2017), pp. 2325–2329. DOI: 10.1109/LGRS.2017.2763738.

[13] O. Ronneberger, P. Fischer, and T. Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. URL: https://arxiv.org/pdf/1505.04597.pdf. (accessed: 20.01.2020).

[14] A. Nowaczynski. *Deep learning for satellite imagery via image segmentation*. URL: https://deepsense.ai/deep-learning-for-satellite-imagery-via-image-segmentation/. (accessed: 20.01.2020).

[15] karolzak. *keras-unet 0.1.2*. URL: https://pypi.org/project/keras-unet/. (accessed: 20.01.2020).

[16] Tensorflow community. *tf.keras.optimizers.Adam*. URL: https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam. (accessed: 21.01.2020).