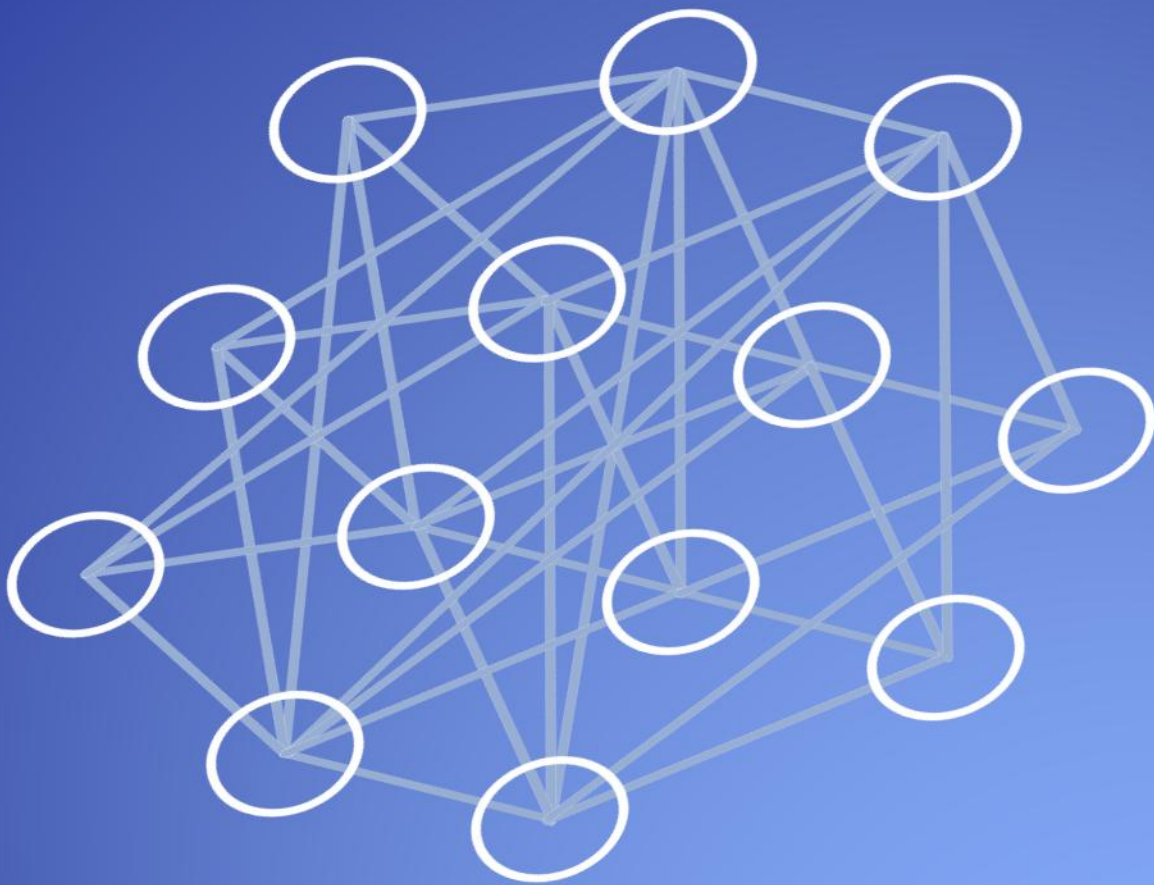


Factorial experimentation regarding Machine Learning in the context of Land Use/Land Cover classification



Emil Falk Knudsen
4th semester, Master in Land surveying
May 28th 2025



**AALBORG
UNIVERSITY**

Title page

The Technical Faculty of IT & Design
Surveying, Planning and Land management
Strandvejen 12-14
9000 Aalborg
<https://www.en.tech.aau.dk>



**AALBORG
UNIVERSITY**

STUDENT REPORT

Title:	Factorial experimentation with Convolutional Neural Networks in the context of Land Use/Land Cover classification
Project:	Master's thesis
Project period:	January 2025 – May 2025
Project group:	Group 12
Members:	Emil Falk Knudsen (20204841)
Supervisor:	Associate Professor Maike Schumacher
Physical page count:	72
Standard page count:	65,5
Symbol count:	157.196
Word count:	26.285

Abstract

This master's thesis serves as a follow-up to my 9th semester report, *Land Use & Land Cover Classification of Aalborg Municipality*. The 9th semester report explored the possibilities of generating a Land Use/Land Cover map for Denmark through Machine Learning (ML). The accuracies achieved in said report were lower than desired with an Overall Accuracy (Po) of 74% and a Cohen's kappa of 0.63. This was hypothesised to primarily be due to two reasons related to the development being executed in Google Earth Engine (GEE). The first reason is that GEE's stratified sampling function limited the amount of training and validation data that could be used. The second reason is that GEE was limited to non-Neural network models.

This thesis explores whether better results can be achieved using Neural Networks (NN) as opposed to GEE's Random Forest (RF) architecture. This is first explored through a literature study to determine the difference between RF and other decision tree ensemble methods compared to NN, which type of NN is most suited for Computer Vision tasks that require spatial awareness, and which factors are relevant to the accuracy and computational performance. Based on the literature study, which was consulted, this master's thesis is focused on the construction and implementation of Convolutional Neural Networks (CNN).

As a result of the literature study, a partial factorial experiment was set up and executed with inspiration from Design for Six Sigma's Design of Experiment methodology. The experiment included seven experimental factors, of which the first two were attempts to mitigate the impact of the relatively large background class, while the remaining factors regarded the CNN model's architecture, the type of data augmentation applied, the class weights applied in the loss functions, the Learning Rate and the number of epochs that the model is trained on. The implementation of these factors, the justification for the chosen data, and the general structure of the GEE and Python scripts written for the NN factorial experiment are described as well.

Upon having optimised the script for each factor, the accuracy measures of the CNN model are compared to the RF model from the previous report. In the comparison, it is found that the CNN model outperformed the RF model relatively by 15% for Po and 24% for kappa.

Preface

This master's thesis was developed throughout the spring semester of 2025 for the Surveying, Planning and Land Management master's programme. It serves as a direct follow-up to my 9th semester report. However, it takes a very different direction than the one outlined in the 9th semester report. This thesis takes a mostly technical, experimental approach to the development of Neural Network based Land Use/Land Cover maps with a goal of outperforming the RF model developed in the previous report. The experiment takes inspiration from Design for Six Sigma.

First and foremost, I would like to extend my thanks and gratitude to Associate Professor Maike Schumacher for her guidance as the supervisor of this master's thesis. Additionally, I would also like to thank Assistant Professor Shaoxing Mo and Professor Ehsan Forootan for showing interest in and contributing with ML scripts and articles that served as inspiration for this thesis and its ML scripts.

All figures included in the report are generated by the author of this master's thesis. The tools used for making these figures are PowerPoint, Excel, Google Earth Engine, and Python. When figures are inspired by a particular source, it is stated in the figure text along with the source of the figure which provided inspiration.

Reading guide

Referencing to sources

The chosen method of citing sources for this project is the Harvard method. This means that sources will be referred to in the following manner: [Name, Year, Pages]. An example hereof could be [Howard and Gugger, 2020, p. 298, 322-324]. Sources can occur in one of three ways. If placed in the middle of a sentence, it refers to that particular sentence. If it occurs at the end of a sentence before punctuation, it refers to the of the entire sentence. If it occurs at the end of a paragraph, it refers to all or the majority of the content in that paragraph. Further detail of the sources can be found in the bibliography.

Appendices will be attached separately in a folder along with the hand-in of this master's thesis. Appendices will be referred to as Appendix X, where X denotes the letter assigned to the given appendix. For appendices with multiple files, the specific file will be referred to by the name of the script iteration that it is based on. The naming convention for these files is thus FxLy, where F is short for factor, x denotes the factor that is experimented on, L is short for level, and y denotes the level which is tested. thus, an example could be Appendix C, F1L1.

Structure of thesis

The thesis can be summarised with the following structure:

- Context
- Pre-analysis
- Main analysis
- Summary conclusion & reflections
- Bibliography and abbreviations

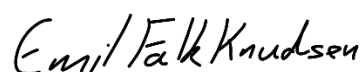
The **context** of the report is provided by Chapters 1 and 2. Chapter 1 specifically regards the political context as well as a brief summary of previous research pertaining to the topic of this thesis. This includes both my 9th semester report and reports from University of Copenhagen, Aarhus University, Roskilde University, and the Technical University of Denmark, which the 9th semester report is based on. Chapter 2 describes the general methodological approach of this thesis.

The **pre-analysis consists** of Chapters 3 and 4. Chapter 3 presents the initial wonder, which is explored further in Chapter 4. Chapter 4 explores the differences between Decision Tree Ensemble and Neural Network (NN) models, what distinguishes CNNs from other types of NN, and a selection of NN parameters which could be experimented with in this thesis.

The **main analysis** consists of Chapters 5-9. Chapter 5 presents the problem formulation and its sub-questions. Chapter 6 justifies the choice of data used for images, labels, and Region of Interest in the experiment of this master's thesis. Chapter 7 describes which parameters are chosen as experimental factors, which response variables will be used to measure the impact of said factors, and which parameters are treated as constants throughout the experiment. Chapter 8 describes the structure of the scripts developed for this thesis. Chapter 9 justifies the choice of optimal level for each factor in the experiment, reflects on the reliability of the experiment and compares the CNN model to the RF model.

The summary conclusion and reflection consist of Chapters 10 and 11. The findings of both the pre- and main analyses are summarised in Chapter 10. Here the initial wonder as well as the problem formulation and its sub-questions are answered to the extent that this thesis allows. Chapter 11 follows up with a reflection upon how the experiment could have been improved if the scope had been expanded.

The bibliography can be found in Chapter 12, where additional information regarding each source used in this thesis can be found. Additionally, chapter 13 has a definition of each abbreviation used in this thesis.



Emil Falk Knudsen

Table of Contents

1	Introduction: Context of master's thesis	9
1.1	Previous research: Analysis regarding political context	9
1.2	Previous research: Analysis regarding development of Random Forest model	10
2	Methodology	15
2.1	Research design choices	15
2.2	Use of Generative AI	18
3	Initial wonder	19
4	Literary study	20
4.1	Machine Learning	20
4.2	Neural Networks vs. Decision tree models: Why use CNN?	21
4.3	Convolutional Neural Networks	23
4.4	Neural Network factors	26
5	Problem formulation	33
6	Justification for choice of data	34
6.1	Region of interest	34
6.2	Input labels	34
6.3	Input images	35
7	Factors & response variables	38
7.1	Experimental factors	38
7.2	Choice of response variables	41
7.3	Constant factors' levels	44
8	Script Description	45
8.1	Data pre-processing	45
8.2	Machine Learning	49
9	Results	51
9.1	Factor 1: Black out background class	51
9.2	Factor 2: Exclude Background class in loss functions	52
9.3	Factor 3: Model Architecture	53
9.4	Factor 4: Data augmentation	54
9.5	Factor 5: Class weight adjustment in loss functions	54
9.6	Factor 6: Learning rate	55
9.7	Factor 7: Number of epochs	56
9.8	Reliability of results	57

9.9	Factor levels after experimentation.....	58
9.10	Comparing CNN output with RF output.....	59
10	Conclusion	61
10.1	Initial wonder	61
10.2	Problem formulation	61
11	Reflection: Potential improvements.....	64
11.1	Cross examination and full factorial experiment	64
11.2	Separate training and validation regions.....	64
11.3	Assembly of a <i>Land Use/Land Cover (LULC)</i> map	65
12	Bibliography	66
13	Abbreviations.....	70

1 Introduction: Context of master's thesis

Climate change impacts the world to an ever increasing extend, and does so in multiple different ways. This includes but not limited to an increase in the amount and intensity of natural disasters, loss of biodiversity, and a steadily rising sea level. The latter particularly impacts coast-near communities. With a large stretch of coastlines and relatively flat terrain, Denmark is a country that risks being severely affected by such changes. [United Nations] [Navigating 360, 2024, p. 4]

As pointed out in the article “Prioritering af Danmarks areal i fremtiden” (Prioritisation of the area of Denmark in the future), Denmark struggles with not having enough land area to accommodate all of its adopted spatial plans and goals, with the article stating that appr. 30-40% more land area being needed to accommodate these plans and goals [Arler et al., 2017, p. 3]. This issue seems to only be growing due to increasing political ambitions, such as “Den Grønne Trepert” (the Green Trinity) [Ministry of Food, Agriculture and Fisheries of Denmark, 2025] and the European Biodiversity Strategy [European Commission, 2025], as well as due to the above mentioned effects of climate change, with increasing sea levels causing the surface area of Denmark to shrink over time [Danmarks Meteorologiske Institut, N/A].

1.1 Previous research: Analysis regarding political context

Concurrently with this political shift toward more focus on the environment, several Danish universities have dedicated their research to how spatial planners and policy makers might accommodate the incongruence between Danish spatial plans and available land area. These universities include University of Copenhagen (UCph), Aarhus University (AU), Roskilde University (RUC) and the Technical University of Denmark (DTU). As a result, UCph and AU have put forth recommendations on where to establish protected nature areas with basis in preservation of Danish biodiversity. RUC have presented a mixed qualitative and quantitative approach to improve quantitative data analysis. Additionally, DTU, in collaboration with the think tank Navigating 360, have presented their findings on expected future impacts of storm surges in Denmark. This research was the basis for the pre-analysis of my 9th semester report, titled “Land Use & Land Cover Classification of Aalborg Municipality” [Knudsen, 2025]. [Petersen et al., 2024] [Ejrnæs et al., 2022] [Christensen and Eetvelde, 2024] [Navigating 360, 2024]

While variations can be found in the process through which UCph and AU developed their recommendations, they achieved remarkably similar results. As can be seen in their maps, a significant amount of the western Jutlandish coastline has been appointed as recommended for nature protected areas [Petersen et al., 2024, p. 8] [Ejrnæs et al., 2022, p. 17]. In DTUs research, it can be seen that coastal areas will be particularly impacted by storm surges, of which some are affected more than others [Navigating 360, 2024, p. 23]. [Knudsen, p. 7-9]

UCph, AU, DTU and RUC have all contributed with thorough and relevant research. However, there were still new and unique topics to explore [Knudsen, 2025, p. 10]. Thus, with inspiration from the above-mentioned research, a small series of semester reports was initiated at Aalborg university. The goal of this series was originally to determine if it was possible to develop a Spatial Decision Support System (SDSS) model for calculating the resistance toward implementing UCph's or AUs nature protection recommendations on a local scale based on

resistance factors such as land ownership, expected future hydrological changes and future Land Use/Land Cover (LULC) conditions across Denmark.

1.2 Previous research: Analysis regarding development of Random Forest model

As a result of the pre-analysis, the 9th semester report's main analysis was focused on the development of a collection of historical LULC maps for use in future LULC prediction. This collection of maps was supposed to be produced within the time range for which Landsat (LS) 8 images are available, that being from 2013 to present. These maps were developed using the Machine Learning (ML) capabilities within Google Earth Engines (GEE) Application Programming Interface (API). This enabled the option to keep the majority of the data pre- and post-processing, ML training, validation and utilisation in one environment. This environment also provided high level coding options and access to online processing power provided by Google. The access to online processing power ensured that the local machine used for programming did not become a bottle neck because of underpowered hardware for the application. [Knudsen, 2025]

1.2.1 Choice of input data

The final Random Forest (RF) model required three input datasets. These were the Region of Interest (ROI), images for training and validation, and labels for the same region.

The **ROI data** was requested to cover Aalborg Municipality. This data was derived from Denmark's Administrative geographic subdivision (DAGI) and was downloaded from Dataforsyningen. As the official administrative border registry for Denmark, this gave the clearest definition of the spatial location and extend of Aalborg Municipality. [Dataforsyningen, a]

Regarding **input images**, it was specifically chosen to use LS8 satellite images. This choice was in part made because of the fact that its images contain bands in both the visible range as well as the infra-red spectrum, including Near Infra-Red (NIR) and Thermal Infra-Red (TIR). Additionally, LS8 is still active, which means that new LULC classifications would be possible to produce at an annual basis in the future. Since LS9 measures within the same bands, it can function as a backup in case LS8's nominal mission should end [USGS, c].

For **input labels**, it was chosen to use GeoDenmark (GDK) data as a basis. GDK was chosen for two reasons. The first one is that this is the dataset which is basis for the INSPIRE LULC map for Denmark [Klimadatastyrelsen, N/A]. The second reason is the fact that it is human-made. This makes it a more authoritative source of comparison than other algorithm generated LULC maps of the ROI, such as the sentinel-based ESA WorldCover maps [ESA, 2025]. This statement is based on the assumption that human made maps are more accurate than ML made maps which are trained on human-made labels.

The following list of GDK themes were chosen:

- Bassin
- Bykerne (City center)
- Erhverv (Industry)
- Hede (Moorland)
- Højbebyggelse (Tall urban areas)

- LavBebyggelse (Low urban areas)
- Råstof (Raw Materials areas)
- RekreativtOmråde (Recreative areas)
- SandKlit (Sand/Dune)
- Skov (Forest)
- Sø (Lake)
- TekniskAnlægFlade (Technical Facility plane)
- TekniskAreal (Technical Areas)
- VådOmråde (Wetlands)

From these, the following classes were derived through experimentation:

- C0: Bassin
- C1: Hede
- C2: HøjBebyg & LavBebyg & Erhverv & Bykerne
- C3: Råstof
- C4: Rekreativt
- C5: SandKlit
- C6: Skov
- C7: Sø
- C8: Vådområde

Notable for the input data of the RF model is that there is no clear spatial distinction between training and validation regions. This is due to the lack of spatial awareness in the RF model and the relatively small amount of data used for the model, since the model was limited to Aalborg Municipality.

One class that is noticeably missing in the GDK dataset is agricultural fields. It was considered to include agricultural fields as a class in the RF model. However, because agricultural data is registered on an annual basis in Denmark, it was deemed unnecessary to dedicate effort to generate annual classifications of the agricultural areas of Denmark. Therefore, the RF model was specifically not trained to recognise agricultural fields. This meant that the model would incorrectly classify all agricultural land in Denmark. However, due to the annual availability of agricultural data, the LULC output from the RF was simply overwritten with agricultural data for the relevant year. This replaced all the incorrectly classified agricultural pixels and increased the accuracy of the model.

1.2.2 Model structure

The RF model along with most of the pre- and post-processing of data was performed in GEE. While multiple iterations of the RF model script existed and were included in the report, they were all structured similarly, with the script being split into multiple distinct parts, which each served a unique purpose for the script. These parts can be categorised as follows:

- Parts 1, 2 & 2.1: Import, define & preprocess RF input data
- Parts 3, 4, 5 & 5.1: Pixel counts & stratification
- Part 6: Classification & visualisation
- Part 7: Accuracy assessment & export of Confusion Matrix (CM)
- Parts 8-9: Agricultural data processing & overwrite
- Parts 10 & 10.1: Legends

Section 1.2.2 is primarily based on Chapter 7 of the 9th semester report. [Knudsen, 2025, p. 23-31]

1.2.2.1 Parts 1 & 2: Import, define & preprocess RF input data

Parts 1 and 2 imported data to allow the rest of the script parts to process the data. **Part 1** imported the 14 originally chosen GDK themes along with the Aalborg Municipality polygon from DAGI. Important to note is that all of the themes had clipped been to the ROI and dissolved to a single polygon pr. class before being uploaded as a GEE asset.

Part 1 is where the **first factor** in the factorial experiment, the **class list**, was tested. The class list is the different categories which the images will be categorised according to when generating the LULC map. This factor was aimed at determining a class list which reached a Choen's kappa (kappa) of 0,61 or higher while maintaining as many distinct classes as possible. This was done by combining and removing various GDK polygon themes. kappa is a measure which describes how accurate a model is when correcting for chance agreement, see Section 7.2. 0,61 was chosen as this is the lower limit for achieving a kappa which indicates a substantial agreement between what a model predicts and reality, as stated by Landis and Koch [Landis and Koch, 1977, p. 165].

Parts 2 and 2.1 dealt with the import and pre-processing of LS8 images and the calculation of various normalised difference indices derived from the LS8 data for future experimentation.

1.2.2.2 Parts 3, 4, 5 & 5.1: Pixel counts & stratification

Parts 3-5 and 5.1 dealt with the additional pre-processing of input data. This process was done using stratified sampling. These parts relate to the **second factor** in the experiment, which sought to find the **optimal distribution of training points** between classes. This factor was important as the relative distribution of points that represented each class in training was found to affect the accuracy of the model. The effect on accuracy appeared to particularly affect the classes which had the smallest surface area.

Because of an upper limit of 5000 points when running stratified sampling, this method resulted in the smallest classes based on surface area being severely underrepresented. Because these 5000 points had to be shared between training and validation data, the smallest class only received a single training point with no validation points. As such, other approaches were also tested. These approaches included an equal distribution of points and a compromise between the equal and area-based distributions. The compromise had points distributed between classes based on relative surface area, with classes below a certain threshold receiving additional points until the threshold was reached. The compromise with a threshold value of 100 points was found to give the best results [Knudsen, 2025, p. 37-38].

Part 3 merged the polygons from the chosen classes into a mask layer consisting of all labels. **Part 4** calculated pixel counts, from which it derived the relative class surface area distributions. **Part 5** did the stratifying sample, where the actual experimentation for the point distribution class took place. **Part 5.1** finished the pre-processing of data by splitting the nigh 5000 points between training and validation data. The split was made pr. Class to ensure that the distribution of points between classes in both the training and validation sets fit with the intended distribution for each approach and level of the factor.

1.2.2.3 *Parts 6 & 7: Classification & visualisation, accuracy assessment & export of CM*

With the training and validation data fully prepared, **part 6** trained the RF classification model on the data from 2021. In early iterations of the script, the model would only classify the ROI for 2021. Later iterations, including the final one, would classify LS8 images from 2013 through 2024 based on the 2021 training data. Regardless of the number of classifications, these would be displayed along with the LS8 image or images that were classified to allow for visual assessment of the RF model's performance.

Part 6 is also where the **third factor** in the factorial experiment was implemented. This factor had the goal of finding the approximate **amount of decision trees** required to no longer gain substantial increases in accuracy. This factor was included because it was requested by Mølbak that the RF model should be made publicly available and runnable on a website. The best performing number of trees was 175 in a range of 100-300 trees [Knudsen, 2025, p. 38-39].

Part 7 would assess the accuracy by generating a Confusion Matrix (CM) from the validation data. A CM is a matrix which describes how many pixels were correctly classified for each class and how many pixels were confused with each incorrect class, see Section 7.2. This confusion matrix was what all other accuracy assessment methods were derived and calculated from, see Section 7.2. In the assessment of the RF model's performance, emphasis was placed on the overall accuracy (Po) and kappa value. As described in Section 7.2, the Po describes the percentage of correctly classified pixels out of the total number of pixels classified, while kappa is a measure which hypothetically adjusts the Po based on expected chance agreement. Together, these two values give an easily interpretable description of how well the model performed. However, if particularly egregious issues were found when visually assessing the performance of the RF model during experimentation, this would take priority over the quantitative accuracy assessment methods, as can be seen regarding the point distribution factor [Knudsen, 2025, p. 36].

1.2.2.4 *Parts 8, 8.1 & 9: Agricultural data processing & overwrite*

While not included in the final iteration of the RF model script, these parts were included in previous iterations mentioned throughout the report. **Part 8** rasterised the agricultural data so that it had the same spatial sampling as the LS8 images. **Part 8.1** calculated the area of the agricultural data in preparation for the following accuracy assessments. **Part 9** then overwrote the Rf classification output so that it contained the correct classification for the agricultural areas of the ROI.

1.2.2.5 *Parts 10 & 10.1: Legends*

Parts 10 and 10.1 generated legends within GEE to ease interpretations of the resulting classifications. Part 10 generated the legend for the classes which were classified by the RF model while part 10.1 generated the legend for the agricultural data which the RF classification was overwritten with for improved accuracy. Similarly to parts 8-9, part 10.1 was not included in the final iteration of the script for RF based LULC classification.

1.2.3 *Results of RF model optimisation*

The result of the partial factorial experiment on the RF classification model was an overall accuracy of 74% and a kappa coefficient of 0,63 for the resulting 2021 LULC before the agricultural data overwrite. This increased to an overall accuracy of 90% and a kappa coefficient of 0,83 after overwriting the 2021 output with the agricultural data.

These results lie within the ranges of moderate and substantial strength of agreement respectively [Landis and Koch, 1977, p. 165]. Despite this, the resulting maps were deemed insufficient for use in future LULC prediction for the ROI. This was particularly due to the Producer's Accuracy (PA), see Section 9.10. PA calculates the Po, but for each individual class independent of other classes. lower PA values were found for classes which were underrepresented, including the hydrological classes Bassins and Lakes.

Furthermore, two untested factors were postulated to have held back the accuracy which could be achieved in the report. The first reason was the selection of deployable models in GEE, particularly that no Neural Network (NN) models were available. The other reason presented was the 5000-point restriction on the stratified sampling method. As such, this master's thesis will focus on the development of an NN for LULC classification instead of the nature protection implementation resistance SDSS that was proposed the author's 9th semester project report. [Knudsen, 2025, p. 44]

2 Methodology

In this chapter, the overall methodological approach of and considerations related to this master's thesis will be outlined. This is the foundation of both the report's contents and the experiments upon which this report is based. This chapter will specifically discuss the research philosophy, research type, research strategy, data collection methods and data analysis tools in said order. Following this, it will clarify the use of generative AI to make the programming of the master thesis' resulting script more efficient.

2.1 Research design choices

2.1.1 Research philosophy

As stated by Dr. Robert Galliers, there are two major philosophies from which to draw inspiration from when working with information systems. Those are the scientific/empirical approach, also known as the positivistic approach, and the interpretivistic approach. The former is based on a premise which emphasises objectivity and reproducibility, whereas the latter is based on a premise which states that multiple subjective interpretations of the same phenomenon can be correct, aligning more with a post-positivistic mentality. While neither is more correct than the other, it can be said with certainty that they each can be more suitable depending on the goal of research. [Galliers, 1991, p. 331]

With a goal of understanding and optimising Machine Learning (ML) in order to achieve the highest measurable accuracy, which can be described as a quantitative experimental approach, the research philosophy of this report lies well within the positivistic realm.

2.1.2 Research type

Another important choice to make regarding the approach taken to research is whether the research will be conducted inductively or deductively. Similarly to my 9th semester report, there are elements of both in this thesis. The overall goal of this report is to determine whether Neural Network (NN) models are more accurate than Random Forest (RF) models in the context of Land Use/Land Cover (LULC) classification. While this clearly demonstrates a deductive mindset, where it is sought to prove or disprove a theory, the approach taken in this report to reach such a conclusion takes on a more exploratory, data-driven character, in which no assumptions or hypotheses are made before the execution of experiments. As such, this report's research can be described as primarily consisting of an inductive research type.

2.1.3 Research strategy

The primary focus of this report being to explore whether an NN ML model can achieve a greater accuracy than that of the previously developed RF model. Therefore, the research strategy can be viewed as a two-pronged approach, consisting of literature research and scientific experimentation.

My previous semester report already has covered the political context and relevance of this kind of research, see Section 1.1. Therefore, the **literature research** of this project will primarily be aimed at developing a greater understanding of the technical and practical aspects of implementing AI. Focus will particularly be aimed at understanding AI in the context of geoinformatics (GeoAI), Computer Vision and Semantic Segmentation. Thus, the determining criteria for a source's reliability has been narrowed down compared to the previous report of

this series. The political status was considered reasonable justification for use of sources in certain contexts previously. This has been narrowed down to primarily regard the academic and developer status of the authors of sources used in this thesis. This means that a source will be considered reliable if it either was written by researchers with a relevant background or by a person or company who contributed to developing the topic of research, be that python libraries, cloud services, hardware components, etc.

To ensure the highest quality of sources, the technique snow balling has been used to find the original source of certain information in order to mitigate the risk of misunderstandings caused by repeated reformulation of the same information from source to source.

As for the **scientific experimentation** in the main analysis, a significant amount of inspiration was taken from the book “Design for Six Sigma: A roadmap for product development” by Yang and El-Haik [Yang and El-Haik, 2003]. In chapter 12 of this book [Yang and El-Haik, 2003, p. 367-406], Yang and El-Haik present the theory of Design of Experiment (DOE), which aims to describe the relation between the outcome of a given process and factors which are part of the process as the following function:

$$y = f(x_1, x_2, \dots, x_n) + \varepsilon \quad [1]$$

Of which y is the outcome of the process, also known as the response variable, $f(x_1, x_2, \dots, x_n)$ describes the relation between the controllable variables and ε is the experimental error. ε is also known as the experimental variation, which is defined as the sum of the influence from uncontrollable factors, those being z_1, z_2, \dots, z_n .

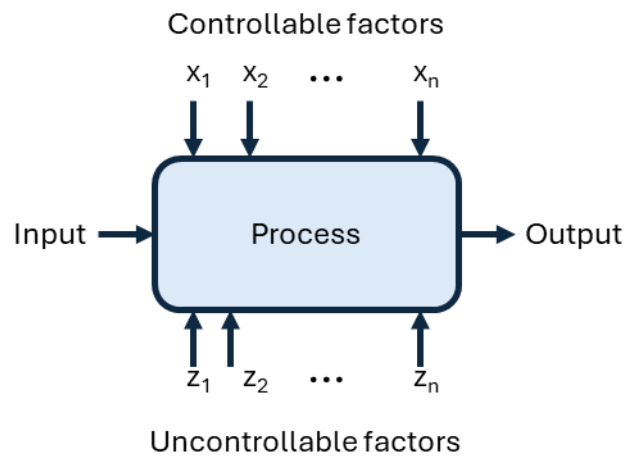


Figure 1: An illustration of the relation between the input, out, and controllable as well as uncontrollable factors as proposed in DOE [Yang and El-Haik, 2003, p. 367]

When basing an experiment on DOE, Yang and El-Haik propose a 7-step process for the planning and analysis of the experiment, which goes as follows.

Step 1 is the **project definition**, in which the objective of the project is identified and defined along with the scope of said project.

Step 2 is the **choice of response variable**. This is the output of the function and should provide meaningful, continuous and accurate insight into the effect of various factors.

Step 3 regards the **choice of not only the experimental factors, but ideally also the levels and ranges** which will be tested for each factor. Levels refer to the particular values that will

be tested for a given factor while the range refers to the difference between the two levels that are the furthest apart in cases where the levels of a factor can be described in a quantitatively determined order. Note that due to the interplay between the contents of step 2 and 3, these two steps are often executed concurrently.

Step 4 is the **choice of experimental design**. This step is dependent on the scope of the project, as this determines whether there will be the needed resources to perform a full factorial experiment. These resources can include time, money, and manpower. Depending on the resources available, there may not be enough for a full factorial experiment. If there are not enough resources for a full factorial experiment, the alternative is a partial factorial experiment. If a partial factorial experiment is chosen, certain factors will have to be left out of the experiment.

With the planning in place, step 5 concerns the actual **execution of the experiment**. This step also involves considerations such as how to ensure reproducibility in the experiments, only changing one factor at a time while maintaining the remaining factors as constants and recording all results for future analyses and documentation.

Step 6 regards the **analysis of gathered data**. Depending on the choice of experimental design and the objective of the project, the execution of this step can vary. Typically, the analysis will include statistical analyses of the data. Goals with the analysis can include:

- Identification of significant vs. insignificant factor effects and interactions
- Ranking of relative importance of factor effects and interactions
- Empirical mathematical model of relationship between response variable and factors
- Identification of optimal factor levels based on project objective

With the planning, execution and analysis done, a conclusion can be reached and recommendations made. This is where step 7, **conclusions and recommendations**, comes in. Here it is either found that there is enough information in the analysis to make a decisive recommendation or that there is need for further experimentation. In order to determine the reliability and reproducibility of the experiment, it is also recommended to perform control runs of the experiment at this stage to ensure recommendations are well founded

2.1.4 Data collection

As part of ensuring the highest degree of validity and reliability from the factorial experiment, it is necessary to use appropriate quantitative data with the highest quality possible. To ensure this, data has only been downloaded from authoritative sources, those being Google Earth Engine (GEE) for satellite data, Datafordeler.dk and Dataforsyningen.dk (Denmark's primary official geodata distribution sites) for GeoDenmark and Administrative Borders data and LandbrugsGIS (The Danish Agricultural Agency's official geodata distribution site) for agricultural geodata.

2.1.5 Data analysis tools

As an extension to the positivistic research philosophy of this thesis, the analytical approach of this thesis is that of a quantitative approach. This is because collections of big data will be compared through interpretable quantitative measures such as the Overall Accuracy (Po), Cohen's kappa and PA.

The factorial experiment and the evaluation hereof were facilitated by the use of the following software. GEE was utilised for the cleaning and extraction of image data for the model. QGIS was used to reduce the complexity of data that was the foundation of the labels. Python was used for the generation of label data, the preparation and augmentation of the image and label data for use in ML, as well as the execution and accuracy assessment of the experiment. The Python codes were primarily written in Google Colab and executed in Jupyter Lab. A Python library which is particularly important to mention would be PyTorch, as it is the basis of the ML parts of the scripts.

2.2 Use of Generative AI

In order to increase the amount of time available for literature studies and experimentation with ML, permission was granted to use Generative AI (Gen AI) to aid in the development of scripts. The way that Gen AI was used is comparable to asking a teacher or expert on python, GEE, or ML for help with writing the scripts. All of the implemented code from Gen AI has been proofread to ensure it functions as intended. The Gen AI models used to aid in the writing and troubleshooting of the ML scripts which contributed to the analysis of this master's thesis were ChatGPT's Code Pilot and ChatGPT 4o models.

Gen AI has not been used to produce text or other content for the report than the above-described scripting assistance.

Pre-analysis

3 Initial wonder

This report will take a technical, explorative approach to testing whether it is possible to achieve a higher accuracy with a Neural Network (NN) based model as opposed to decision tree-based models (e.g. Random Forest). In order to explore this hypothesis, it is necessary to develop an understand of state-of-the-art AI. Hence, the following initial wonder was formulated:

“What is considered state of the art for Machine Learning (ML) and semantic segmentation and how can it be implemented in the context of Land Use/Land Cover (LULC) map generation?”

4 Literary study

As stated by Guérin et.al., There are large collections of geospatial data which have been collected over time. These Big Data collections can be challenging to make sense of for humans. As a result, AI has been adopted in the realm of Geoinformatics with success. [Guo et.al., 2020, p. 358]

The following chapter will explore how Machine Learning (ML) works as well as how it has been implemented in Geoinformatics through the field of GeoAI. This chapter will start with a broader focus on ML before narrowing in on Neural Network (NN) models and its subcategory, Convolutional Neural Networks (CNN).

4.1 Machine Learning

This project focuses on the implementation and comparison of ML models. Therefore, it is important to have a clear understanding of ML. While ML can be considered a type of program, it differs from the traditional understanding of what a program is. This is because the purpose of ML is to automate the optimisation of performance. [Howard and Gugger, 2020, p. 20-22]

A program in the traditional sense will process an input and give a result, as can be seen on Figure 2. This program has been optimised in advance by a person. Where ML differs is that ML has a training phase in which it is capable of optimising itself. An ML model has not only an input in the form of input data, but also parameters. Parameters as a term is sometimes treated as interchangeable with the term model weights, though model parameters usually also cover other parameters than just weights. These parameters impact the results of the model, also known as the predictions. When predictions have been generated, they are then compared to the actual known value of the data which the model is trained on. The known values are commonly referred to as labels or ground truth data. The comparison between the model results and the ground truth data describes how well the model performed. This is also known as the loss. the weights can then be updated based on the loss. This iterative process is repeated until a result of satisfying quality has been achieved. What determines whether the quality is satisfying depends on the goals of the specific project. It should be noted that once training is complete, an ML model will function more like a typical program. [Howard and Gugger, 2020, p. 20-25]

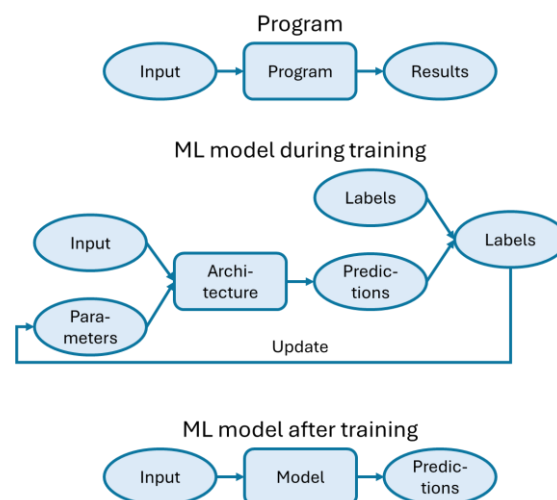


Figure 2 Simplified visualisation of the way that Programs vs ML models function. [Howard and Gugger, 2020, p. 20-25]

4.2 Neural Networks vs. Decision tree models: Why use CNN?

The specifics of how the architecture section functions varies depending on the type of ML model that is deployed. While this report does not contain an exhaustive list, notable types of ML models would be decision tree based and NN based ML models.

4.2.1 Decision tree-based models

The decision tree category of ML models includes models such as Random Forest (RF) and Gradient Tree boost (GTB). These are both examples of ensemble decision tree models. Ensembling in the context of machine learning means using multiple models with the goal of achieving a higher accuracy than previously possible. GTB uses boosting, which is a type of ensembling where decision trees are trained sequentially. This means that the performance of one decision tree affects the following decision tree. Alternatively, RF executes ensembling through bagging. This means that a collection of decision trees is trained in parallel and independently, after which the average output of all the trees becomes the model's final output. The main reason for choosing one over the other is that boosting can achieve more accurate results than bagging when model parameters are tweaked correctly, but boosting is more prone to overfitting than bagging. Since overfitting causes model to be worse at generalising [Smith, 2018, p. 5], this is undesired. This is the reasoning for the choice to use RF during the 9th semester report [Knudsen, 2025, p. 27-28]. [Howard and Gugger, 2020, p. 298, 322-324]

However, decision tree-based models are not spatially aware. This means that the model classifies each pixel of an image independent of the surrounding pixels. This was postulated to be a major contributor to the confusion that was found between the urban class (C2 for RF model) and several other classes. [Knudsen, 2025, p. 44]

When the spatial context is taken away, it can be difficult to determine whether an area with lots of trees is part of a forest or a garden. Similarly, it can be difficult to tell whether a pixel with a reflectance corresponding to that of a building belongs to a dense urban area or an urban sprawl area when the model cannot see if there is a garden in the neighbouring pixel. This is reflected in the Confusion Matrix (CM) of the most accurate RF model, as shown in Figure 3. It can be interpreted from the colour coding of the entries that the largest degree of confusion for classes 0 and 3 through 8 occurs with class 2, the urban class. This means that when the pixels which belong to classes 0 and 3 through 8 are misclassified, they are most commonly misclassified as class 2. Furthermore, it can be seen that pixels belonging to class 0 are more likely to be misclassified as class 2 than to be correctly classified.

		0	1	2	3	4	5	6	7	8
Bassin	0	5	1	6	0	0	0	5	1	2
Hede	1	0	50	4	0	0	0	42	0	3
Høj & Lav Bebyg & Erhverv & Bykerne	2	0	1	295	0	0	3	62	0	8
Råstof	3	0	0	5	8	0	0	4	0	2
Rekreativt	4	0	0	5	0	11	0	3	0	0
SandKlit	5	0	0	6	0	0	16	0	0	0
Skov	6	0	4	38	0	0	0	319	1	12
Sø	7	1	0	2	0	0	0	2	19	1
Vådområde	8	0	3	24	0	0	1	15	2	48

Figure 3 The confusion matrix for the run of the RF model with the optimal level for each factor. The diagonal, which indicates correct estimations, has been highlighted in grey. Yellow, orange and red highlight varying degrees of confusion.

Due to their lack of spatial awareness, decision trees were postulated to not be the most suitable models for dealing with tasks where spatial context is important. Hence the goal of this master's thesis to explore which types of models would be more suitable for tasks which require spatial awareness.

4.2.2 Neural Networks

NN is a subcategory of ML which was proposed by McCulloch and Pitts in 1943. NN has seen a significant increase in popularity in recent years. NN differ from other types of ML models because they attempt to imitate the neural structure found in human brains [Howard and Gugger, 2020, p. 5 and 20].

A neural network consists of multiple layers of neurons. These layers can be categorised into the input layer, the hidden layer(s) and the output layer. Relating these terms back to Figure 2, the input layer corresponds to the input, the output layers correspond to the predictions, and the hidden layers are the architecture. The hidden layers are what distinguishes NN from other ML model categories, such as RF and GTB. The number of hidden layers is also what determines whether an NN is a Deep NN (DNN) or not, with any NN containing more than one hidden layer being considered a DNN. [Kufel et.al., 2023, p. 8-11]

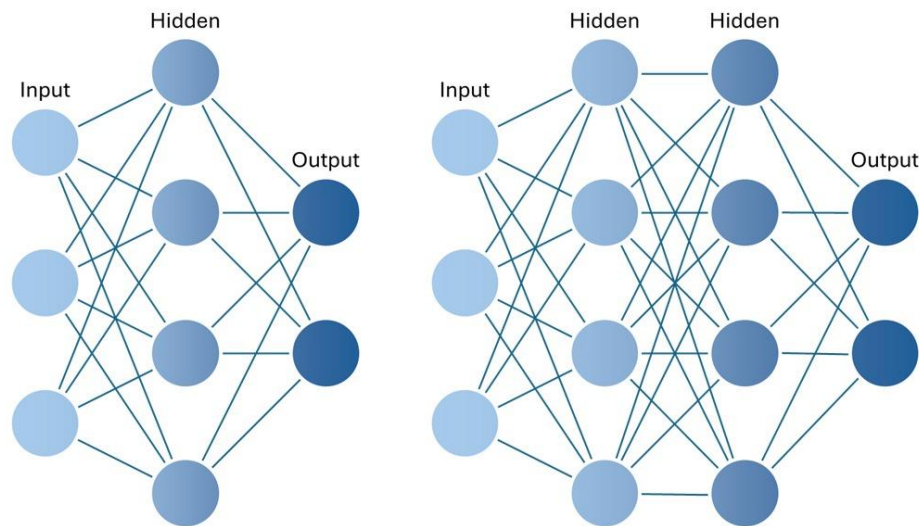


Figure 4 examples of an NN that cannot be considered deep (left) and an NN that is considered a deep NN

In 2021, Minaee et.al. executed a survey where they compared the reported performances of 100+ NN models based on training with commonly used image segmentation training datasets. These kinds of common datasets used for comparisons are also known as benchmark datasets. According to the definitions presented by Minaee et.al., there are three types of image segmentation, those being semantic, instance and panoptic segmentation. Semantic segmentation is the task of defining which class each pixel belongs to while instance segmentation is the task of defining which unique object each pixel belongs to. Panoptic segmentation is a combination of semantic and instance segmentation. [Minaee et.al., 2021, P. 1-2]

The task of Land Use/Land Cover (LULC) map generation through ML falls within the category of semantic segmentation. Multiple different types of NN architectures can be applied in this context. To compare different semantic segmentation models, Minaee et.al. Used the Jaccard Index, which is also known as the Mean Intersection over Unit Index (mIoU). This

describes the average performance of the model across all classes. Among the best performing models, the dilated CNN architecture family DeepLab can be found. [Minaee et.al., 2021, p. 7, 12]

The fact that the DeepLab architecture family generally performed well fits well with Howard and Gugger stating that CNN is the current state of the art for computer vision tasks [Howard and Gugger, 2020, p. 298, 322-324]. Therefore, from here on out, this master's thesis will focus on CNNs.

4.3 Convolutional Neural Networks

What sets CNNs apart from other types of neural networks are, as the name suggests, convolutions. These make CNNs particularly suited for image recognition. This is because it has reduced computational costs due to the reduced number of parameters required to train and run CNNs compared to NNs. By reducing the computational costs of processing images, CNNs are capable of handling larger and more complex image related tasks than traditional NNs. Furthermore, the reduced number of parameters also results in a less complex model, which reduces the risk of overfitting. [O'Shea and Nash, 2015, p. 2-3]

While variations of CNNs which incorporate other types of layers exist, many CNNs are based on the following types of layers [O'Shea and Nash, 2015, p. 4-5]:

- Input layer
- Convolutional layers
- Pooling layers
- Fully-connected layers
- Output layer

The **input layer** simply contains a number of neurons equivalent to the number of pixels in a single image from the training dataset. The number of pixels is defined by multiplying the height, width and number of channels in an image. Hence, it is necessary in most CNNs that all images have the exact same dimensions. E.g., if the training dataset is the MNIST benchmark data, which contains black/white (single channel) images with a resolution of 28 x 28 pixels, the input layer will have $28 * 28 = 784$ neurons. If the training dataset consists of equivalently sized images with RGB colours (3 channels), the input layer will have $28 * 28 * 3 = 2352$ neurons. Similarly, if the training dataset contains images with RGBD or RGBA (4 channels), the number of input neurons would increase to 3136. Similarly, the dataset contains 4 channelled images with a resolution of 64 x 64, the needed input neurons would be 16384. In a traditional fully-connected NN, each of the neurons in the following layer would have N weights, with N equating to the number of neurons in the input layer. However, the needed number of weights pr. neuron is greatly reduced by the following layer types. [O'Shea and Nash, 2015, p. 2-5]

Convolutional layers are layers which apply one or multiple kernels, also known as filters, to each individual pixel of an image to generate a new output from the previous image. This is achieved by calculating the scalar product of the kernel and each individual pixel. The derived output will highlight particular features. This feature highlighting output is commonly called the activation map. An example where the kernel produces an activation map which highlights left and right sides of an object respectively can be seen below in Figure 5. As the model's

accuracy improves due to training, these kernels will become better at detecting relevant features. [C O'Shea and Nash, 2015, p. 5-6]

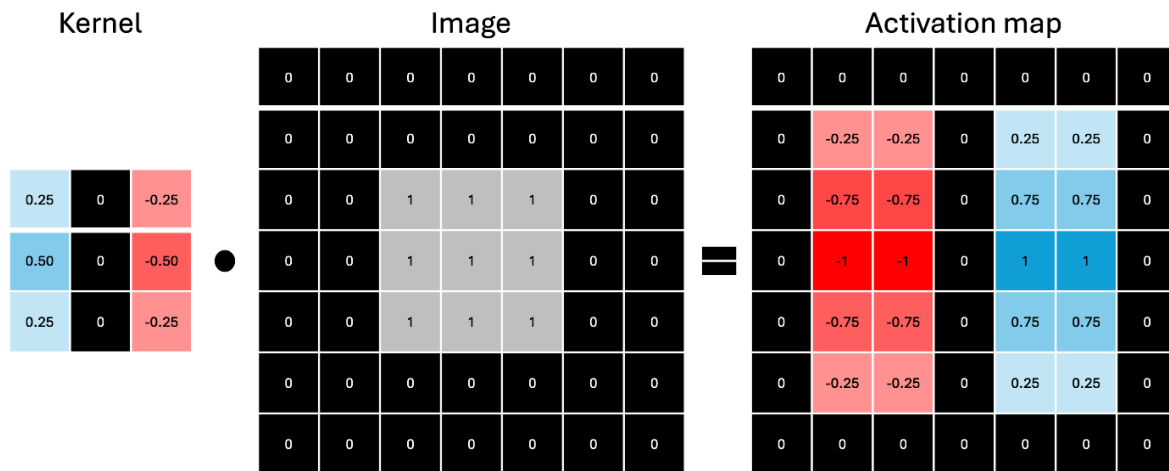


Figure 5 An example of the scalar product between a kernel and an image along with the derived activation map. Note that in order to calculate the value pixels along the edge, the image was padded with 0s. Padding is not visualised in the example.

Parameters in convolutional layers are based on the kernel. While a kernel's dimensions usually only cover a small portion of the width and height of the images at a time, it typically will cover all channels of an image at the same time. This means that, if a kernel has a height and width of 3 and the image contains 4 channels, the number of weights associated with this kernel's dimensions would be $3 * 3 * 4 = 36$. The height, width and depth of the kernel is also known as the receptive field. The number of weights for a neuron in a convolutional layer is independent of the size of the input image. While a significant number of neurons will still be needed, the reduction of weights pr. Neuron compared to fully-connected NNs still saves significantly on processing power pr. neuron. [O'Shea and Nash, 2015, p. 6-7]

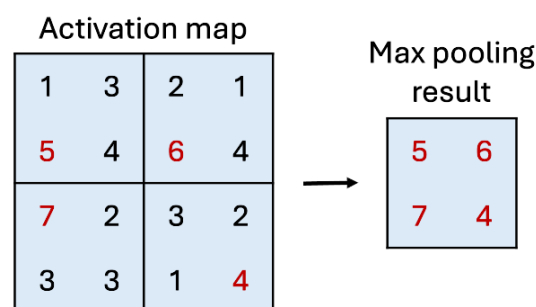


Figure 6 An example of max pooling, where each group of 2×2 pixels has been reduced to a single pixel with a value corresponding to the max value for each 2×2 grouping.

Pooling layers are typically placed right after one or multiple convolutional layers. The purpose of pooling is to reduce the size of the activation maps. This allows for more efficient computing and allows the model to more easily discover new patterns in the data. Multiple types of pooling exist, including max pooling, average pooling [Gholamalizadeh and Khosravi, 2020, p. 3] and overlapping pooling. A common pooling method is max pooling with the dimensions for the pooling kernel being 2×2 , meaning that the activation map is split into groups of 2×2 pixels which are all reduced to the biggest value within that given 2×2 group, as illustrated in Figure 6. This allows the model to learn from an increasingly bigger area of the input image without increasing the kernels' receptive fields [Minaee et.al., 2021, p. 2]. By

pooling groups of 2×2 , the image is reduced to 25% of its original size. When pooling with dimensions 2×2 , it is important ensure the height and width of the data images is divisible by 2. [O'Shea and Nash, p. 8]

The following layer types are used for classification based on the activation maps. For simple object recognition, where the goal is simply to know what the image contains, but not where the content is located on the image, it is common to use **fully-connected layers**, as described by O'Shea and Nash. Fully-connected layers resemble those of a standard NN, where the input initially is the final output of previous convolutional and pooling layers. [O'Shea and Nash, p. 4-5]

However, because the goal of this master's thesis, as pointed out in Section 4.2.2, is semantic segmentation, other types of layers are more recommendable. To understand which other types of layers can be recommended for the classification based on the derived activation maps, it is necessary to understand the terms encoder and decoder. CNN model architectures can generally be split into two different sections, the encoder section and the decoder section. The encoder section is the part of the script that is responsible for converting the input data into activation maps. It does so by extracting features from the data which the model determines significant for the distinction of relevant classes. The decoder section is the part that is responsible for deriving a classification from the activation maps. While the Encoder of a CNN generally will consist of a combination of convolutional and pooling layers this is not necessarily the case for the decoder depending on the requirements posed by the task at hand. If the goal is simply to determine what is on an image, a decoder which resembles a standard fully-connected NN will be sufficient. [Minaee et.al., 2021, p. 2-3]

However, for more complex tasks such as various types of image segmentation, it is beneficial to include layers similar to those seen in the encoder. Fully-convolutional architectures are model architectures which contain no fully-connected layers. Instead they use a combination of convolutional layers and skip connections in the decoder to upscale the activation maps generated by the encoder with the higher resolution information of activation maps from previous layers. Skip connections are a method sometimes implemented in more complex models to mitigate the complexity's impact on the effectiveness of the given model's training [Lundby et.al., 2023, p. 1]. Note that while the abbreviation FCN is both used for Fully-Connected and Fully-Convolutional Networks in various articles based on context, FCN will refer specifically to Fully-Convolutional Networks throughout this master's thesis. [Minaee et.al., 2021, p. 3]

Since the development of CNNs, many attempts have been made to further improve image processing by modifying CNN architectures to create new architectures. Attempts include but are not limited to architectures such as Recurrent CNNs (R-CNN) and dilated CNN. of these, the DeepLab architecture family falls into the latter category. Dilated CNNs, also known as atrous CNNs, are similar to regular FCNs except that the convolutional layers hold an extra hyperparameter called the Dilation Rate (DR). The DR describes how close or far apart the entries of a kernel are. The number of pixels between the entries can be described by DR, as seen in Figure 7. The dilation rate allows the kernel to have a bigger receptive field while maintaining the number of weights it has. As an example, a 3×3 kernel with $DR = 2$ has the receptive field of a 5×5 kernel with $DR = 1$ and a 3×3 kernel with $DR = 3$ has the same receptive field of a 7×7 kernel with $DR = 1$. [Minaee et.al., 2021, p. 6-7]

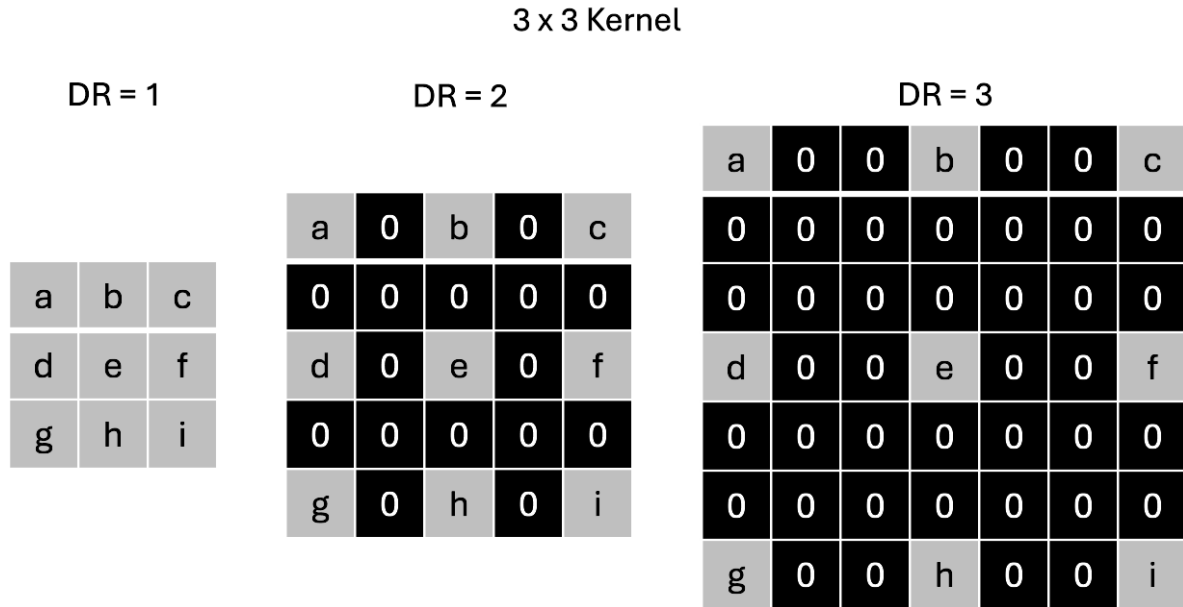


Figure 7 Examples of dilated convolution kernels. The example from left to right demonstrate the impact of increasing the Dilation Rate (DL).

The use of dilated convolutional layers has been implemented in architectures like DeepLabv2 to mitigate the impact of downscaling data with pooling layers. Furthermore, DeepLabv2's implementation of dilated convolutional layers is also reported to improve the ability to localise object boundaries. These features have since been further improved in the following DeepLabv3 and DeepLabv3+ architectures. [Minaee et.al., 2021, p. 7]

4.4 Neural Network factors

When developing an NN model of any kind, there are multiple considerations to be made regarding the construction of it. While certain factors, such as the model architecture or CNN specific ones like the kernel size and DR have already been mentioned, this section will function as a general overview of relevant factors. The purpose of this chapter is to support the execution of steps 3 and 4 in Design of Experiment (DOE) by covering relevant factors from which the experimental factors can be chosen. In this master's thesis, factors will be split into four categories, those being:

- Architecture
- Training
- Regularisation
- Miscellaneous

Note that the following list of factors is not exhaustive. Instead, this should be seen as the first step toward choosing the controllable factors which will be included in this master's thesis' factorial experiment. Some of these factors are typically referred to as hyperparameters but will be referred to as factors in this master's thesis to adhere to the terminology of Design for Six Sigma.

4.4.1 Architectural factors

As stated in section 4.3, when applying CNNs in a semantic segmentation, it makes sense to view the architecture as an encoder section and a decoder section. The encoder converts the

input of the CNN to activation maps while the decoder develops an output based on the activation maps. Both the encoder and decoder can vary based on the number and type of layers they have.

While DeepLabv3/DeepLabv3+ is sometimes discussed as an entire architecture in and of itself [Minaee et.al., 2021, p. 7], it often is built upon an encoder such as one of the ResNet CNN architectures [PyTorch, a] [Melinda et.al, 2024, p. 443-444]. The article which introduced the DeepLabv3 model even compares the DeepLab architectures performance based on whether it is coupled with ResNet-50 or ResNet-101 as it's backbone [Chen et.al., 2017, p. 5-6].

In the comparison performed by Chen et.al., it was found that DeepLabv3 would perform better on the benchmark data when using the ResNet-101 backbone. This proved that the choice of backbone will impact the performance of the DeepLabv3 architecture. Furthermore, their comparison to other architectures, including an FCN and the DeepLabv2 architecture, showed that DeepLabv3 outperformed both. This shows that both the choice of encoder and decoder is of important to the performance of a semantic segmentation model. [Chen et.al., 2017, p. 5-7],

4.4.2 Training factors

No matter the chosen architecture, there are several factors relevant to the training of the NN model. The training relevant factors include, but are not limited to, the following:

- Loss function
 - o Class weights
 - o Class exclusion
- Optimiser
- Learning rate
- Epochs
- Batch size

The way that a model learns is in part defined by the loss function and in part by the optimiser. However, in order to find the most suitable loss function, it is necessary to understand what the intended use of the model is.

ML models can serve different purposes depending on the nature of the task at hand. ML models are commonly defined as either a regression or classification model. **Regression models** are models that attempt to predict a numeric value such as temperature or coordinates. Regression models can also be described as making **continuous predictions**. Regression models are not to be confused with models based on linear regression. **Classification models** differ by assigning a class or category, such as dog or cat. Classification models can also be described as making **discrete predictions**. The goal of this master's thesis is to assign discrete labels to pixels for which the coordinates are already known. Therefore, this thesis will be focusing on classification ML models. [Howard and Gugger, 2020, p. 28]

The way that an ML model is updated depends somewhat on whether it is dealing with a regression or classification task as well as what type of classification task. However, the fundamentals of the process are the same for each. The process starts with calculating the difference between the predicted results of the models and the labels. This is also known as **the loss** [Howard and Gugger, 2020, p. 24]. For single label classification, Howard and Gugger recommend using the Cross Entropy Loss function. For multi-label classification, they

recommend Binary Cross Entropy with Logits Loss. The data used in relation to this master's thesis only contains one label pr. pixel, such as either forest **or** wetland, not forest **and** wetland. Therefore, the type of classification utilised is single label classification, thus making Cross Entropy Loss the most suitable type of loss. The loss function is used to find the negative gradient vector, which is used by the optimizer to update the model's weights [Howard and Gugger, 2020, p. 162-163]. [Howard and Gugger, 2020, p. 237]

Loss functions can be further modified to contain weights for each of the label classes. For specifically Cross Entropy Loss in PyTorch, it is possible to assign a weight tensor in which the weight of each class is specified. Furthermore, by using the command `ignore_index` in PyTorch, it is possible to entirely ignore a class in the loss function so that the specified class is not optimised for. [PyTorch, b]

As with loss functions, there exists different **optimizer** options. The simplest optimisers merely apply Stochastic Gradient Descent (SGD) to find the most optimal adjustment of weights for the model to move in the direction where the errors should lower the most efficiently. However, to improve the efficiency of optimisers, some optimisers employ a method called momentum. **Momentum** allows for previous training updates to impact the adjustment of weights. This helps the model to avoid getting stuck in smaller valleys found along the way toward a more significant local minimum. The degree of influence from momentum is adjustable. A larger influence on the optimiser from momentum can cause the model to ignore changes in the direction of the gradient. Therefore, it is important to use a reasonable compromise. Another way of improving an optimiser is by implementing RMSProp. **RMSProp** is an alternative to SGD where the Learning rate varies. In order to understand what this means, it is first necessary to understand what the learning rate is. [Howard and Gugger, 2020, p. 471-477]

The **Learning Rate** (LR) describes the size of the step that is taken in the direction of the negative gradient vector. There are multiple methods for designing a model's LR. One way of categorising LR functions is proposed by Wu et.al. and contains three categories. These categories are fixed, decaying and cyclic LR functions. Of these, fixed is considered to be the simplest to implement, but also the slowest and most inefficient of the three. When implementing a **fixed LR**, the LR is constant throughout training. This is the traditional approach to LR. The size of an LR can impact the way that an ML model learns, and thus how high of an accuracy can be achieved. A smaller value increases the chance of getting closer to a local minimum. However, small LR values can be disadvantageous because there is no guarantee that the found local minimum is the global or a significant minimum. Smaller LR values also have a harder time breaking away from a local minimum. Furthermore, a smaller LR also means that it generally requires more steps to approach a minimum. In contrast, larger LR values are more likely to break away from local minima to find either the global or a lower local minimum. Larger values also move toward minima in fewer epochs. However, they risk moving past a minimum, while still being stuck in the valley of that minimum. This can result in the loss oscillating as the LR is struggling to find the actual minimum value. As such, Wu et.al. recommend choosing a lower LR value when the loss function yield worse or oscillating results, whereas larger values are recommended when the loss functions yield consistently better results. The methods within the latter categories were developed to reach a compromise between the advantages and disadvantages of smaller and larger values. **Decaying LR** functions attempt to do so by initially using larger values to quickly approximate the position of a significant local minimum. The LR then gradually decreases in size to allow for a closer

approximation of that local minimum. **Cyclical LR** values do not only decrease the size of the LR value, but also periodically increases it in a cyclical pattern. [Wu et.al., p. 1-3]

RMSProp further modifies the LR by raising the LR for weights that have had a corresponding entry that is close to 0 in the negative gradient vector over a given number of updates as this indicates that the LR is not big enough to improve the weight significantly. Conversely, RMSProp lowers the LR for weights with an entry in the negative gradient vector that has had more drastic changes over recent updates. [Howard and Gugger, 2020, p. 477-478]

A common optimiser which combines SGD with RMSProp and momentum is Adam. These traits have made the Adam optimiser the standard optimiser for ML in the FastAI python library. [Howard and Gugger, 2020, p. 479]

An **epoch** is defined by the model having trained on all training data and validated on all validation data once. The number of epochs chosen when training an NN will in practice depend on two parameters; How much time is available for training and when the model achieves the highest accuracy possible. To understand the second half of this statement better, it is necessary to understand the concept of under- and overfitting. [Howard and Gugger, 2020, p. 32]

Under- and overfitting are two scenarios in which a model is not optimised properly, but for different reasons. When underfitting occurs, this is a result of the model not having been trained enough. Underfitting can be mitigated by increasing the number of epochs. Overfitting occurs when the model has gone through too many epochs, resulting in the NN model no longer generalising well. This is a result of the model learning too many features from the training data, including features unique to that dataset. Overfitting is more complex to mitigate. To know whether a model is underfit, overfit or optimally fit to the training data, it is necessary to look at the results of the loss functions. [Smith, 2018, p. 4-5]

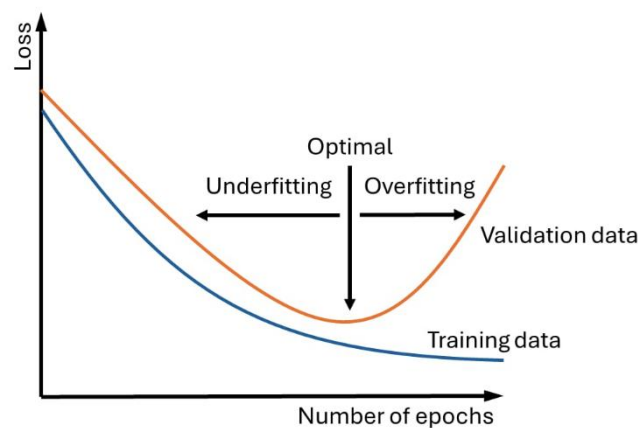


Figure 8 A visualisation of how to determine whether a model is under- or overfit based on the validation loss. Figure is inspired by Smith [Smith, 2018, p. 3].

When plotting the loss for validation data, underfitting can be detected by the loss continuing to decrease, showing that it has not found a local minimum. Overfitting can be detected by the validation loss increasing over multiple epochs. The optimal degree of fitting to the training data lies where the local minimum for the validation loss lies, as this is where the model has learned as much as it can from the training data without compromising the model's ability to

generalise. Several solutions for overfitting exist. Some of the methods which can mitigate overfitting include batch size and learning rate (LR) [Smith, 2018, p. 3-5, 7].

Batch size is a factor that in part results from computational limitations. The size of ML dataset is commonly beyond the size that regular computers can handle when training an NN. Therefore, it is often necessary to split the datasets into smaller batches, so that the computer does not run out of memory when attempting to train the model. This is done using batch size, which aids in determining how many batches the training and validation datasets will be split into. This is done by specifying how many images may, at most, be within a batch. Each of these batches will be used individually to optimise the weights of the model through the process of SGD. When choosing the batch size, it is a compromise. Larger batch sizes will result in shorter run times due to more efficient use of hardware. Findings made by Smith also suggest that larger batch sizes in combination with larger LRs tend to yield higher validation accuracies. Conversely, smaller batch sizes can function as a form of regularisation, which will improve the model's ability to generalise. [Smith, 2018, p. 7-8]

4.4.3 Regularisation factors

While loss functions can help with detecting under- and overfitting, it does little to solve the issue. The act of mitigating overfitting is called regularization. Santos and Papa performed a study of regularization methods. In this study they discuss three general categories of regularization for CNN models, those being: [Santos and Papa, 2022, p. 3-4]

- Data augmentation
- Label regularization
- Internal structure change

Data augmentation is a collection of regularization methods which are specifically applied to the input image data. The advantage of data augmentation is that it improves the model's ability to generalise without requiring the production of more labelled data. This is because data augmentation artificially creates more labelled data through the modification of existing labelled data [Howard and Gugger, 2020, p. 60]. This category includes methods such as Cutout, where random regions of data in an image are removed during training, RandomErasing, where the removed regions are replaced with other data, and Mixup, where separate images are blended together. These are all data augmentation methods. However, while some of these methods do not require a change in the label tile associated with the image tile, other methods do. In their mention of data augmentation, Howard and Gugger, mention methods such as rotation, flipping, warping, brightness change and contrast change [Howard and Gugger, 2020, p. 74]. [Santos and Papa, 2022, p. 8-10]

Methods such as brightness and contrast change mostly do not require augmentation of the labels. However, Mixup, rotation and flipping do when working with image segmentation specifically, as these methods change either the content or location of pixels in the image. Therefore, while data augmentation specifically targets the input images, it does not always do so exclusively. Often times, data augmentation methods are accompanied by label regularisation. **Label regularisation** is the act of augmenting label data. One such method would be label smoothing. Label smoothing aims to mitigate not just overfitting, but also model overconfidence. Though label smoothing can be applied by itself, it can also be applied in relation to data augmentation methods such as Mixup. An example could be two images are

overlapped into one image, where each pixel's value is determined as 60% of image 1 and 40% of image 2. Label smoothing would reflect this by averaging out the label confidence so that the label would say the pixels of the image are 60% likely to belong to the class of picture 1 and 40% likely to belong to the class of picture 2. Label smoothing in its original sense is utilised throughout the entire training phase. However, at a certain point in training, the improvement caused by Label Smoothing is prone to wearing off. A variant which attempts to improve the effect of Label Smoothing is Two-Stage Label Smoothing (TSLA). TSLA implements label smoothing in early stages of training before disabling it in later stages. [Santos and Papa, 2022, p. 14-15]

Internal structure changes is a collection of methods that impact the weights and kernels during training without augmenting input images or labels. This category is split into two sub-categories, those being Dropout based methods vs. Other methods. The majority of the methods mentioned in this category fall within the Dropout category. Dropout is a regularization method which, in its most basic variant, removes random neurons from the NN during training. This keeps the model from becoming overly reliant on these neurons. However, removing neurons randomly does not ensure the most relevant neurons are removed. Thus, several variants of the original dropout method have been proposed to improve the effect of dropout. These variants include, but are not limited to, methods such as MaxDropout, where the neurons that are most likely to be triggered are removed, and DropBlock, which is a CNN specific method that essentially performs a cutout on the activation maps. [Santos and Papa, 2022, p. 12-13]

4.4.4 Miscellaneous factors

This category covers miscellaneous methods through which a model can be optimized in relation to different parameters. The factors covered in this section are:

- Early stopping
- Mixed precision learning
- Seeds

Loss functions aid in determining when overfitting occurs and regularisation aids in postponing overfitting. **Early stopping** saves time in the training phase by stopping the training phase when overfitting occurs. It does so by stopping the training phase when improvements have not been made over a given number of epochs and then saving the model from the epoch with the lowest validation loss as the optimal iteration of that given model [Scikit learn, N/A]. Because the lowest validation loss and highest validation accuracy rarely occur in the same epoch, with the highest accuracy usually occurring a few epochs after the lowest validation loss, Howard and Gugger recommend running a model twice, once where you have enough epochs for overfitting to occur and then once again with a specified number of epoch equating to when the previous run of the model began overfitting [Howard and Gugger 2020, p. 213]. However, early stopping is still worthy of consideration for this thesis, as early stopping effectively cuts the collective run time in half at bare minimum.

Another method which can decrease the required run time and computational requirements is **Automatic Mixed Precision (AMP)**. AMP is a tool which allows for the mixed use of different float data formats (Float32 vs. Float16) during the training of the model. This is relevant because certain processes are significantly faster when run with Float16 as opposed to the standard float32 format. However, it is worth noting that this is a Graphics Processing Unit

(GPU) specific method, that not all GPUs are capable of utilising this method, and the impact of the method varies for GPUs which are capable of using it depending on the specific GPU in question. AMP is particularly effective when the GPU in question contains tensor cores. [PyTorch, c]

The GPU used for this thesis is an Nvidia RTX 5080, which is built with the Blackwell GPU architecture [Nvidia, 2025, p. 6]. The Blackwell architecture is not specifically mentioned in the PyTorch documentation on AMP. However, Nvidia has stated in a report on the Blackwell architecture that not only does it contain tensor cores which support the Float16 and Float32 data formats, but that the Blackwell Architecture also further builds upon the Turing and Ampere architectures, which are both mentioned in the PyTorch documentation on AMP [Nvidia, 2025, p. 7-8, 15]. [PyTorch, c]

Seeds are also relevant, though not for optimisation, but rather for reproducibility. In order for the results of this thesis to be valid, some degree of reproducibility for the experiment is necessary. Perfect reproducibility is not guaranteed in PyTorch. However, some of the subprocesses in a PyTorch ML script can be reproducible when utilising the same version of PyTorch on the same device while utilising the same components, particularly the Central Processing Unit (CPU) and GPU. This is done using seeds, which is a given string of digits that will be used as the starting point of any otherwise pseudo-random process. When reproducibility is desired in PyTorch, it is recommended to implement a global seed at the beginning of the script. However, it should be noted that the implementation of reproducibility is likely to slow down the training of ML scripts. [PyTorch, d]

5 Problem formulation

Based on the findings of chapter 4 regarding semantic segmentation with Neural Network (NN), it was determined that this master's thesis will focus on the following problem formulation:

“Is it possible to achieve a higher accuracy of Land Use/Land Cover (LULC) classification with Convolutional Neural Networks (CNN) than previously found with Random Forest (RF)?”

To determine the sub-questions, it is worth considering the potential goals of a Design of Experiment (DOE) analysis. The goal of an analysis can be to determine or rank the significance of factors, determine the specific relation between factors and the response variable, or identify the optimal level of each factor, see Section 2.1.3. With the goal of achieving a higher accuracy, the direct goal of this thesis can also be described as attempting to find the optimal levels for each factor. Hence, the following sub-questions were formulated:

- Which level for each of the experimental factors is optimal for achieving the highest accuracy possible?
- What is the significance of the interaction between the model and its input data?
- To what extent can the NN model outperform the RF model?

Main analysis

6 Justification for choice of data

This chapter expands on the choices which are stated in the data collection strategy, see Section 2.1.3. The chapter will justify the choice of data for the Region of Interest (ROI), images and labels.

6.1 Region of interest

Before downloading any data, it is necessary to determine the ROI for the project. For this report, it was chosen to work with all of Denmark. The reason for using Denmark is that this was the eventual goal of the 9th semester report. While the initial goal was to develop a model for Aalborg Municipality and then upscale the model to a national scale, it has been chosen to go straight to the national scale when developing the Neural Network (NN) model. This is due to the following three reasons.

The first reason is that it was found through the second experiment in the previous semester report that the distribution of class representation in the training data impacted the final classification. If this model should be applicable on a national scale, it seems more logical to train it on said scale. The second reason is that, as stated in chapter 4.3, Convolutional Neural Network (CNN) models do not take points as input data, but instead images. As such, it is necessary with a bigger area to achieve an equal sample size before implementation of data augmentation. The third reason is based on upscaling issues that were encountered last semester. This issue pertained to the temporal expansion of classification. While the use of Surface Reflectance (SR) Landsat (LS) 8 data provided no major issues for Aalborg when working with the year 2021, it was found to be significantly impacted by cloud cover and other noise for other years. A similar issue could be expected to be encountered for the spatial expansion from municipal to national scale. Thus, in order to ensure the model would work at a national scale, the training and validation data was derived from all of Denmark.

In order to ensure that the data is reliable, the ROI has been downloaded from Datafordeler.dk, which is the authoritative geodata distribution site in Denmark. The ROI was more specifically downloaded as part of the DAGI dataset

6.2 Input labels

Since this thesis deals with the comparison of performance between Random Forest (RF) and NN image segmentation, it is important to compare the model's performances on comparable data. Hence, this largely dictates the choice of label data for this thesis. Thus, the same GeoDanmark (GDK) themes will be combined into the same classes that were derived from the factorial experiment in the author's 9th semester report [Knudsen, 2025, p. 32]. These themes and classes can also be found in Section 1.2.1, though the assigned number to each class is transposed for this thesis' factorial experiment. This means that Bassins will be 1 class for this thesis' experiment instead of class 0 and so on.

The choice of GDK data is derived from the fact that the Danish INSPIRE Land Use/Land Cover (LULC) map is based on GDK and that the data is human made as opposed to Machine Learning (ML) made, see Section 1.2.1. However, because this thesis deals with NN instead of

RF models, certain changes must be made for this dataset to be used in the new context. Because RF models are not spatially aware, they do not need to see the context of the pixels they are trained on. Therefore, the training data consists of individual pixels as opposed to entire images. This meant that the shape of the polygons from which training data was derived was inconsequential.

In contrast, what makes NNs more suitable for spatial image segmentation is the spatial awareness, which comes from the model seeing pixels in the context of entire images. Therefore, the shape of the GDK polygons now matters because they do not perfectly fit with the shape of the square training images which the NN will be trained upon. Therefore, it was initially believed to be necessary to include agricultural data. However, as discussed in depth in Section 7.1, it was found that entire classes could be ignored in the loss function, thus making the agricultural data irrelevant for the experiment. As such, only GDK data is used for label data.

6.3 Input images

While the exact same label datasets could be transferred from the RF experiment to the NN experiment of this thesis, the same cannot be said for the input images that are to be classified. Similarly to the RF experiment, LS8 satellite images will be utilised.

However, two issues occurred as a result of upscaling to a national scale for the NN experiments of this thesis. Firstly, the images to be classified should preferably be collected from the same period as the labels. The monthly timing of collection of the orthophotos that are basis for GDK data is just before leaf spring [Knudsen, 2025, p. 24]. However, the collection years of GDK is less straight forward to determine.

When GDK data is collected, it is done in one of two ways, those being total updates and designated updates. Total updates are where an entire municipality is classified into GDK classes, whereas designated updates are done for smaller areas pr. request from Municipalities. In addition, GDK data has not been saved historically. This means that it is not possible to access previous total updates or the latest total update before designated updates were made. While all municipalities are total updated for GDK at least once every 5 years, this is not done simultaneously for all municipalities. [Knudsen, 2025, p. 24]

This means that, ideally, every municipality's year of total update should be used to filter which year the satellite data should be derived from for each municipality, resulting in a spliced together satellite image for all of Denmark. Ignoring the issue of collection years not being publicly registered for municipalities, this method could result in significant amounts of noise in the training phase. Because of the images' decreased temporal homogeny, the image data of each municipality could vary significantly more in luminosity, soil humidity, and cloud coverage and shadows, which would make it harder for the model to recognise particular features in the dataset.

Furthermore, the issue of cloud cover and shadows persisted despite the recommended switch from SR images to Top of Atmosphere (TOA) images granting greater flexibility regarding cloud cover and shadow removal [Knudsen, 2025, p. 42] [GEE, a]. The choice to switch from SR to TOA was made so that greater control of the cloud removal could be achieved. However, by expanding the spatial scope, the chance of any particular part of the Region of Interest (ROI) being affected by significant cloud cover or shadows was increased. Upon visualising the LS8

TOA images for the ROI in 2021 within the months March through April (the months leading up to leaf spring) without cloud removal and with newer data prioritised, a significant area of the ROI was found to be severely affected by cloud coverage. Attempts to remove cloud coverage within this specified time frame did not achieve desirable results across all of the ROI, as seen in Figure 9. Further attempts to achieve a greater coverage was initially attempted by expanding the month range. Coverage was only found to be acceptable nation-wide when using the time span March through June. However, because June is well past leaf spring in Denmark, the temporal priority was flipped to prioritise older data. However, this meant that the highest prioritised data would be from start of March, which is significantly before leaf spring. In an attempt to mitigate this issue, the month range was reduced to April through June. However, this also resulted in undesirable coverage.

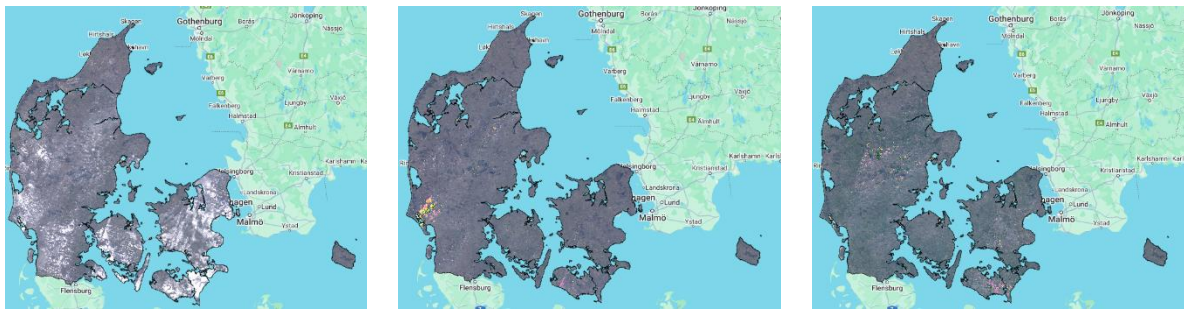


Figure 9 Visualisations of LS8 image data for ROI. Images showcase the result of generating a mosaic from LS8 TOA data from 2021 in the months March through April with newer data being higher priority without cloud and shadow removal (left) and with cloud and shadow removal (middle) as well as from April through June with older data prioritised (right).

Furthermore, these attempted solutions did not account for the variety of years in which different municipalities had been total updated. Thus, a choice was made based on the following assumption: Data noise and loss caused by weather occurrences, soil moisture, cloud cover and shadows, etc. would impact the quality of the NN experiment more than the changes in land use from year to year would. With basis in this hypothesis, it was decided to hold on to the initial month range of March through April with newer data being prioritised. Instead, the quality of different years around the time in which municipalities have last been total updated would be experimented with. This gives a collection of years that spans from 2021 through 2025 with the option to use slightly earlier years if this initial range does not provide desirable result. Based on qualitative comparisons of years 2020-2025, 2020 was chosen as the dataset with the best coverage within the ideal month range.

A major difference between SR and TOA is the spectral available. While SR only has 7 bands which range from ultra blue to short wave infra-red (SWIR), TOA has an additional four bands in the infra-red (IR) and thermal infra-red (TIR) range [GEE, b] [GEE, c]. During the RF experiment, SR bands 2-7 were used [Knudsen, 2025, p. 24]. Similarly, it was chosen to use the same bands for the NN experiment, though band 8 was included as well.

Band 8 distinguishes itself from other bands because it is panchromatic. As such, band 8 covers the range of multiple other bands, including all of band 3 and 4 as well as part of band 2. This allows band 8 to provide a different and more detailed visual description of the measured area as band 8's resolution is double that of the other bands included in the experiments. However, this added detail will be lost on the model due to the model needing all input to be of the same resolution. [KILDE USGS, b]

Therefore, band 8 is included as an additional layer of description of the ROI with the intention of giving the NN model a broader foundation of data to work with. It does not add any information with higher resolution, however. Had the scope of the project been bigger, this could have been yet another factor to experiment with.

The exclusion of band 9 is based on what band 9 is used to detect. Band 9 is primarily used to detect cirrus clouds [KILDE USGS, b]. Because cloud removal has been implemented, this was deemed as unnecessary and potential confusion for the model. Bands 10 and 11 were excluded as well. The primary reason is that bands 10 and 11 have a significantly lower resolution at 100m x 100m pr. pixel as opposed to the 30m x 30m or 15m x 15m pr. pixel spatial sampling of other bands. This means that, in order to utilise these bands, the other bands would have to be upscaled to 100m x 100m pr. Pixel. Not only would this reduce the quality of the NN generated LULC output, but it would also not be possible to cleanly upscale the other bands to 100m x 100m pr. Pixel.

7 Factors & response variables

This chapter will discuss steps 1-4 from the Design of Experiment (DOE) model presented in Section 2.1.3. However, they will be presented out of order.

Step 1 includes the identification and definition of the goal and scope of the project. The goal is defined in the problem formulation, see Chapter 5. The scope is yet to be defined and will thus be defined here. Because this is a master's thesis written by a single student, there are limits as to how much can be achieved before the deadline for the project. Given that Machine Learning (ML) is an expansive field, there are more factors which could be experimented upon than this project has resources for. These factors include but are not limited to the ones described in Section 4.4. Hence, the choice of overall experimental design was apparent before work on steps 2 and 3 was initiated.

For step 4, it was chosen to go with a partial factorial experiment. This means that not all controllable factors will be experimented upon. Furthermore, certain chosen factors have the capacity to contain many relevant levels during experimentation, such as factors 5 and 6. Therefore, it was chosen to not test all permutations of chosen factors and each of their levels, as this would have been too time requiring compared to the expected learning gained from attempting such an experiment. Hence each factor will be optimised one at a time.

As for steps 2 and 3, these have been executed somewhat concurrently, as is commonly done, see Section 2.1.3. However, because this is a software focused project, there were no physical restraints on available options for response variables. As such, the chosen factors have been the primary cause for choice of response variables instead of the available resources for response variables being a limit on which factors can be experimented on. Therefore, step 3 will be presented in Section 7.1 before Step 2 in Section 7.2, as the choice of experiment factors will be utilised when justifying the choice of response variable.

7.1 Experimental factors

For the DOE of this project, the following controllable factors were chosen as experiment factors:

1. Black out of background class
2. Inclusion/exclusion of background class in loss functions
3. Model architecture
4. Data augmentation
5. Weight adjustment of underrepresented classes in loss functions
6. Size of learning rate value
7. Number of epochs in training phase

As mentioned at the beginning of Chapter 7, these factors will be optimised one at a time. However, the order in which they are optimised for is not arbitrary. This is because the factors likely are dependant. This means that level A for factor X might be optimal when factor Y has level B, but not when factor Y has level C. While the optimal approach to determining the optimal setting for each factor would have been to test all permutations of the factor levels for all the factors, this is too resource intensive for the project scope. Therefore, the next best solution was deemed to be to determine a pragmatic order of factor optimisation. The thought process behind the order is based on two criteria. The first criterion is whether a factor is

considered a technical fix as opposed to an actual variable in the experiment. Technical fixes will be tested first so that the actual variable factors can be tested under optimal conditions. The second criterion is how fundamental a factor is considered to be for the construction of an Neural Network (NN) model.

Factors 1 and 2 fall within the realm of technical challenges which were to be worked around. Because NNs use rectangular images as input data, as opposed to single points like Random Forest (RF) models, a choice had to be made. This choice regarded whether the data should be reduced to only contain tiles which purely consist of desired classes or whether there was allowed to be background classes. These background classes would be registered agricultural fields and larger water bodies respectively. Larger water bodies include the surrounding sea and Fjords of Denmark. These water bodies were not included in the Google Earth Engine (GEE) export and thus consist of completely “blackened out” pixels, meaning that the value for each band of each pixel within this class consists purely of 0s. This resulted in completely black pixels for the larger background water bodies when visualised.

Sand Dunes is an underrepresented class, see Table 11, Section 9.10, which shares a significant amount of border with background water bodies connected to the seas around Denmark. Therefore, it was chosen to include background classes so as to not offset the class representation in a manner which negatively impacted an underrepresented class. Hence, it was chosen to only remove tiles which purely would consist of background classes. However, when developing the label data, it was found that Python ignored the Agricultural data due to a supposed error in the geometry of the agricultural data. Because of the method with which the labels were developed, see Section 8.1.2.1, this created a single background class which consisted of two very distinct types of image data, that being the agricultural fields and the larger water bodies. The agricultural areas consisted of the actual measured values from LandSat (LS) 8, and the larger water bodies consisted of blacked out pixels. Furthermore, because these two classes were combined into one big class as a result of the non-functional geometry, it now made for one even bigger background class as its area now is the combined areas of the larger water bodies and the agricultural fields. The stark variation of the class contents and the increased size raised the risk of the background causing confusion in relation to the other classes in classification outputs.

The first attempt to mitigate this risk was **factor 1**, the blacking out of all background data. This factor was developed and chosen based on the hypothesis that the more homogeneous the content of the background class was and the less similar it was to any other classes, the lower was the risk of confusion between the background class and other classes. As such, it was chosen to make the background appear less like classes which could easily be confused with agricultural areas, such as moorland, wetlands and forest. This was done by blacking out all image pixels within class 0.

However, factor 1 only addressed the visual similarities of the background class, not the accumulative size of background water bodies and agricultural fields combined, which still would cause the NN model to underrepresent other classes. Therefore, **factor 2** addresses this issue by removing the background class entirely from the loss functions, thus ensuring the model does not focus on learning to classify the background. This can be done because the background consists of agricultural fields, which can be overwritten with the agricultural dataset post classification in GIS and because the blacked-out waterbodies can be removed

again by clipping the output of the model to the Region of Interest (ROI) polygon. This factor also makes the inability to process the agricultural data in Python irrelevant. It should be noted that because factor 2 removes the background class, it practically nullifies factor 1. The reason that factor 1 is implemented as well is that it is considered less intrusive. Therefore, Factor 1 was considered worthy of exploring the impact of before implementing Factor 2.

The remaining factors focus on actual ML parameters as opposed to technical challenges. As such, the order of factors from factor 3 and onward will be determined by how fundamental each factor is considered to be for the ML model. The two most fundamental of the remaining factors would be the model architecture and the data augmentation. With basis in Figure 2, the architecture can be described as the foundation of the model while data augmentation is an expansion and manipulation of the input to artificially boost and diversify the input. While both orders of these two factors are justifiable, it was chosen to experiment with model architecture first to mitigate the impact of data manipulation on the choice of model architecture. Therefore, model architecture is factor 3.

Factor 3 regards the architecture of the model. To streamline the experimentation process for this factor, it was chosen to use the torchvision.models library. From this library, four models were chosen. These models include:

- fcn_resnet50
- fcn_resnet101
- deeplabv3_resnet50
- deeplabv3_resnet101

This allows for comparing the impact of both the type of encoder and the size of decoder on the accuracy. The DeepLab architectures have been chosen based on the arguments presented in Section 4.3, while the Fully-Convolutional Network (FCN) architectures were chosen to compare the DeepLabv3 architectures to a simpler segmentation Convolutional Neural Network (CNN), see Appendix E.

Data augmentation is **factor 4**. There are several of the regularization methods described in Section 4.4.3 which could be relevant to implement in this factor. However, the scope of this thesis does require a narrowing of this factor. As such this factor will focus on data augmentation and only include three augmentation methods. The included methods are overlap, mirroring and rotation. In this project, overlap and offset will be used interchangeably. The reason for including these methods specifically is that these methods augment the spatial location of the pixels in the input data while not affecting the reflectance or proportions of the input data like stretching warping, brightness, and contrast changes do.

It is important to note that the regular, non-augmented dataset always should be included, and no version of the dataset may appear twice. Therefore, there are up to 8 distinct levels which can be implemented in this experiment. However, these levels have been reduced to 5, with those being:

- The regular dataset by itself
- The regular dataset combined with the offset dataset
- The regular dataset combined with the mirrored dataset
- The regular dataset combined with the rotated dataset
- The regular dataset combined with all three augmented datasets

Through the first four combinations, the impact of each individual method is measured while only executing the bare minimum of runs. This allows for assessing whether any of the methods negatively impacted the accuracy, so that the given method could be excluded from the final combination if need be.

Because the class representation is important and sought preserved to the extent that it is possible, the impact of augmentation on class representation has been mitigated by developing a separate script which augments the entire collection of training data tiles in the exact same way, ensuring that class balance is impacted minimally when generating the offset dataset and preserved entirely when mirroring and rotating, see Section 8.1.2.

Factor 5 regards the weights of individual classes in the loss function weights. After having removed class 0 with factor 2, there was still a significant imbalance between the remaining classes. As this was one of the major shortcomings of my 9th semester project's RF model, this factor was implemented to raise the accuracy of underrepresented classes. An initial minimum goal of 70% in producer's accuracy for all classes was put in place, which was adjusted as the experiment progressed. Because this experiment has 9 unique, continuous variables that can be adjusted, this factor could support an entire full factorial experiment of its own. To accommodate the scope of this master's thesis, initial weights were set at 1 for all 9 relevant classes, after which individual weights were adjusted for underrepresented classes. This resulted in 8 distinct levels for the experiment.

Factor 6 is the Learning Rate (LR). As mentioned in Section 4.4.2, there are multiple different methods for approaching the choice of LR, which can be placed into the categories fixed, decaying and cyclic learning rates. For each of these methods, there are different sub-factors to experiment with. To not exceed the scope of this project, the simplest method to implement has been chosen, which is the fixed LR. As such, this factor will focus on the particular LR value rather than the methods available.

Factor 7 regards the number of epochs. With all other experimental factors optimised, the final iteration of the model can be run at a significantly higher epoch count and without early stopping to see when the model truly overfits. This will help to determine the approximate number of epochs that the model should be trained at. Since early stop is technically implemented but practically disengaged, the best iteration will still be saved [PyTorch, e]. Note that, due to the way that the script is written, epochs will be named from 0 and up. However, the number of specified epochs will match the max number of epochs that can be run. This can be seen for iteration F7L1, where early stop is disabled and 100 epochs are run, which are named Epochs 0-99, see Appendix D, F7L1.

7.2 Choice of response variables

In order to quantitatively determine whether various levels for each factor improve the accuracy of the NN model or not, it is necessary to have an appropriate response variable. This response variable is also what allows for determining how well the NN model does in comparison to the RF model. Combined with the description in Section 2.1.3, that gives the following criteria for response variables:

- Must be relevant to the factors experimented upon
- Must be a continuous scale
- Must allow for comparisons to 9th semester RF factorial experiment

A collection of relevant response variables has been described by Stehman. According to Stehman, there are three distinct categories of accuracy assessment measures for thematic classification models, such as a semantic segmentation model. Each of these categories have their own level of detail. These categories can be described as matrix, vector and single value summaries of the accuracy of an ML model, presented in order of decreasing level of detail. [Stehman, 1997, p. 77-78]

In the matrix summary category, Stehman mentions a single method. That method is the confusion matrix (CM), also known as the error matrix. This is the most detailed method of accuracy assessment. Because of how detailed it is, Stehman recommends to always include the CM when discussing the accuracy of a model. Therefore, all CMs for this master's thesis' experiments are included in human-readable format through the script printouts in Appendix D. CMs are constructed by plotting the number of times that the model correctly classified pixels and the number of times that the model incorrectly predicted pixels to belong to each of the incorrect classes. This is demonstrated in Figure 10, where correct classifications are denoted by T for True Prediction and can be found along the diagonal entries of the CM while the incorrect classifications are denoted by F for False Prediction and consist of all the non-diagonal entries in the matrix. [Stehman, 1997, p. 77-78]

		Predictions				Legend
		1	2	...	N	
Ground Truth	1	T1	F2	...	FN	True prediction for given class
	2	F1	T2	...	FN	False prediction for given class
	
	N	F1	F2	...	TN	

Figure 10 A visualisation of how CMs are constructed.

However, the high degree of detail in CMs can make them more complex to interpret. Thus, it is customary to use summaries with a lower degree of detail for ease of interpretability. However, there are multiple accuracy measures to choose from depending both on the desired level of detail and the intended use of the summary. The use of summarising accuracy measures can have one of two purposes. These can be to either report the final accuracy of a fully developed map or to compare accuracies of different maps to determine which is more accurate. The factorial experiment of this report places the intended use of accuracy measures in this report within the second use case. [Stehman, 1997, p. 77-78]

Had the intended use been to represent the final accuracy of a map, it would have been sufficient to report the Overall Accuracy (P_o) by itself. This accuracy measure is calculated for the validation data by dividing the amount of correctly classified pixels by all of the classified pixels. This can also be described as the sum of the CMs diagonal entries divided by the sum of all entries in the CM. However, because the goal is to determine the most accurate classification out of multiple classifications, Stehman recommends choosing multiple accuracy assessment methods. Of these, at least one should compensate for chance agreement. Compensating for chance agreement means that the accuracy assessment must account for how likely it is that the model is to correctly classify pixels by chance instead of due to proper training of the model. Among the chance compensated accuracy measures presented by

Stehman, it has been chosen for this thesis to use Cohen's kappa coefficient to compliment the P_o . [Stehman, 1997, p. 77-80]

The way that kappa compensates for chance agreement is by normalising P_o with a hypothetical estimation of how likely the model is to correctly classify a pixel by chance, P_e , as seen in Function 2.

$$kappa = (P_o - P_e) / (1 - P_e) \quad [2]$$

Where P_e is calculated as seen in Function 3.

$$P_e = (1/N^2) * \sum n_{i+} * n_{+i} \quad [3]$$

The choice of kappa is based on multiple reasons. The fact that it compensates for chance means that it contains relevant information about the impact that adjusting factor levels has upon the accuracy of the model beyond the information provided by the P_o . Furthermore, it compensates for chance in a manner which is not based on assumptions of a model's accuracy. This makes for a more objective comparison method compared to E.g. tau, which is calculated similarly to kappa, by normalising P_o , but is biased. tau is biased because it is chance adjusted by pre-determined assumptions of the model's accuracy, called a-priori. This bias prevents tau from functioning as a direct comparison. Finally, kappa and P_o are also the accuracy measures used to evaluate the RF model from the author's 9th semester project report along with P_o [Knudsen, 2025, p. 19]. [Stehman, 1997, p. 80-81]

kappa and P_o are suitable for factors 1 through 4, 6, and 7, as these all aim to improve the general accuracy of the model with no class specific goal. One challenge with using kappa and P_o is that they are single value measures and thus encompass all classes. This means that it also includes the background class. Because the model is not optimised for the background class, it will drag down the accuracy measures. To accommodate the removal of the background class, P_o and kappa will be calculated twice for every experiment, once with the background class included and once with the background class excluded. Because the background class is considered insignificant to this project, the deciding values for determining the optimal level for relevant factors will be the kappa and P_o which ignore the background class.

However, while single value summaries are the least complex to interpret between the three accuracy measure categories and allows for comparison to the RF model, it does not contain information on the performance of individual classes. Therefore, it is not a suitable response variable for all factors. In order to meet the relevance criterion for response variables regarding factor 5, the class weight factor, it is necessary to use a vector summary as the response variable. Vector summary measures provide a degree of detail and complexity of interpretation that lies between those of single value and matrix summary measures. Among the ones presented by Stehman are User's Accuracy (UA) and PA. These are calculated similarly to the P_o , but for the individual classes. **UA** describes how likely a prediction is to be correct by dividing the sum of correct predictions by the sum of predictions made for the given class. This can also be described as dividing the diagonal entry for a given class by the column sum for the corresponding class. The **PA** describes how well the model can recognise a given class by dividing the sum of correct predictions by the sum of ground truth samples for the given class. This can also be described as dividing the diagonal entry for a given class by the row sum for the corresponding class. As the goal for factor 5 is to improve how well the model recognises

relevant classes, PA has been chosen as the response variable for this factor. [Stehman, 1997, p. 79]

One last measure to be added is the loss curves. As mentioned in this section 4.4.2, this measure discloses if the model is under- or overfitting. In the literary research of this project, no scientific papers were found which commented on the absolute values of the loss functions. Therefore, assessment of the loss values will be based on the improvement from epoch to epoch. When evaluating the loss functions' outputs, the primary focus will lie on the validation loss graph. While some oscillation is to be expected, the validation loss should mainly be declining. [Smith, 2018, p. 2-5]

7.3 Constant factors' levels

The remaining factors described in Section 4.4 did not become experimental factors and thus will be treated as constant factors throughout the entirety of the factorial experiment. Therefore, their values and arguments for the chosen values will be disclosed in this section.

While the weights of the model's loss functions is a factor, the specific **choice of loss function** is not a factor in the experiment. This is because, as stated in Section 4.4.2, Howard and Gugger recommend using the Cross Entropy loss function for single label classification [Howard and Gugger, 2020, p. 237], which is the category of ML that this thesis' CNN experimentation falls under. As for the **optimiser**, the choice is also based on recommendations from Howard and Gugger. They recommend using the Adam optimiser [Howard and Gugger, 2020, p. 479].

To improve the reproducibility of the experiments, it was chosen to use a **global seed** for all pseudo-random functions in the code which affected the accuracy performance of the model. Note that the seed thus was not used for visualisation of data. The pseudo-randomness in these cases was desirable as it allowed for trouble shooting. The seed was set to 42.

The choice of **batch size** was more nuanced. As stated in Section 4.4.2, Smith has presented pros and cons to both smaller and larger batch sizes. Larger batch sizes were stated to be conducive to higher accuracies and faster runtime while smaller batch sizes improve the generalisation of models [Smith, 2018, p. 8]. The data augmentation factor already focused on improving generalisation and the attempt to improve reproducibility through the implementation of seeds slows down the training process. Therefore, it was chosen to implement a larger batch size within the range of what could be run by the utilised hardware. Upon testing the max batch size of the pc, it was found that the Graphics Processing Unit (GPU) could handle a batch size of 64 images but would run out of Video Random Access Memory (VRAM) at a batch size of 128. Thus, the batch size was set to 64.

As for regularisation methods, no **label augmentation** was implemented. **Internal structure change** was implemented in all four architectures presented in factor 3. Each architecture has a dropout rate in the decoder. The FCN decoder has a dropout chance of 10% while the DeepLabv3 decoder has a dropout chance of 50%, see Appendix E, p. 9, 15, 26, and 35. Because the dropout is related to the decoder, the dropout rate is unaffected by the change of encoder.

Due to time constraints, **AMP** was not implemented, though it would have been beneficial to do so if the time frame of this thesis had been longer.

8 Script Description

In this chapter, the structure and content of the scripts and data processing will be described and justified. It is important to note the distinction between images and masks/labels. Images specifically refer to the LandSat (LS) 8 data while masks and labels refer to the GeoDanmark (GDK) data.

8.1 Data pre-processing

A lesson learned in the previous report was that the quality of the Neural Networks (NN) input data is arguably as important as the model itself and often more labour intensive to develop. As such, before the NN can be run, it is necessary to develop input data of as high a quality as possible. Therefore, the above-mentioned data has been processed as described below.

8.1.1 Google Earth Engine

In order to ensure that the chosen LS8 images are of a satisfying quality and from a reliable source, they have been downloaded from Google Earth Engine (GEE). However, the Top of Atmosphere (TOA) data cannot simply be downloaded directly as stored in GEE. Beyond filtering according to the year and months specified in Section 6.3, it is also necessary to compensate for cloud cover and shadows. Furthermore, more bands requires more Video Random Access Memory (VRAM), which limited the amount of training data that could be used for training a model and increased training time. Because of this and the arguments presented in Section 6.3, the GEE script used for collecting the needed LS8 data was written to only download bands 2-8 For Denmark within the specified time range. This was achieved with the following structure, as can be seen in Appendix A:

First step in the script is to define and visualise the Region of Interest (ROI). The table that defines ROI is an imported GEE asset and thus cannot be seen in the script. The imported asset consists of a shapefile from the DAGI dataset, see Section 6.1.

Following this, the cloud mask function for the LS8 TOA images is defined, in which it uses the QA_PIXEL band to mask out low-quality measurements as a result of cloud cover, cloud shadow and snow cover. Next up, the LS8 data is loaded with a series of filters, including the ROI and the specified temporal range. The cloud mask is then applied before the images are sorted so that newer images with the least amount of cloud cover are prioritised for the following composite. Clipping the LS8 images to the ROI ensures that major bodies of water surrounding the ROI are removed. This ensures that the model will be trained on the classification of land cover specifically.

This composite is then visualised to allow for a qualitative assessment of the cloud removal, after which bands 2-8 are singled out for extraction before exporting all the bands as part of a single geotiff image. Due to the size of the ROI, the LS8 data is exported as two separate files.

8.1.2 Python/Jupyter Lab script

After the input images have been pre-processed in and downloaded from GEE, it is time to generate the label data and convert both the images and labels into a format which is digestible to the NN model. The CNN architectures used in this experiment need consistently sized images. To achieve this, a collection of matching image and label tiles must be generated. In order to do this, a python script was developed. This script works by first generating a label tiff

from the GDK data, then preparing the image label data for tiling, and determining valid tiles before tiling the input image and mask identically and exporting the desired tiles.

8.1.2.1 Structure of script

The process for this thesis' data pre-processing in python is structured as follows

1. Imports
2. Prepare input data
3. Tiling of regular and offset images
4. Training/validation split
5. Augmentation of tiles
6. Conversion of tiles from Tiff to Tensor format
7. Cleaning and normalisation of tensor tiles
8. Setting image values to 0 for Class 0

To streamline the processing, several features which could have been performed with GIS were performed with Python. This also improved the runtime for data pre-processing significantly. A feature of Python which sets it apart from work with typical GIS is the use of libraries. Python libraries are collections of commands that allow for simpler coding, and without the installation and importing of appropriate libraries, python scripts will not work. Therefore, after having set up a virtual environment with the appropriate libraries, **step 1** was to import the needed libraries into the script. Notable is that because of the use of geotiff files, it was found that imageio could not be used. Instead, the rasterio library was used to handle geotiff files.

Step 2 was split into three separate cells, one for the initial merging of the image data, one for the development of mask/label data, and one for the calculation of class representation. The merging of image data is relevant because of the way that the image data was exported from GEE. As mentioned in Section 8.1.1, the LS8 images of the ROI were too big to be exported from GEE as a single file. Therefore, the two files had to be merged before further pre-processing could take place.

The label data was developed by generating a raster file with the same dimensions as the merged images but with a single channel consisting of entries with the value 0. This raster was then intended to be overlayed with the GDK themes and the agricultural theme. It should be noted that in order to reduce the runtime and computing costs, the GDK and agricultural data files were dissolved in GIS ahead of being included in the script.

Because the method used was overlay, the order in which each theme was overlayed matter. This is because themes that were included later in the overlay process would overwrite themes that were included earlier in the overlay process. For most themes, this was not a significant issue as they have been designed to not overlap each other, such as the various urban classes and forests. However, certain themes were found to have been digitized within the boundaries of other themes. Bassins were particularly prone to be digitised within the boundaries of other themes. Therefore, hydrological classes were placed as the last themes to be included in the overlay process.

Upon attempting to execute the overlay, it was found that the agricultural data could not be included in the overlay due to a fault in the file used. Attempts to fix the agricultural files and redevelop the dissolved agricultural theme did not solve this issue. This is the reason for the solution presented in factor 1 being implemented. Attempts were made to develop the label file

in QGIS but would repeatedly fail or crash after 20-40 hours of runtime. Therefore, the label data was developed without agricultural data.

After the overlay, the representation of each class is calculated. The output of this cell is the total pixel count for each class as well as how much of the total label data consists of each class. This relative calculation is run with both the inclusion and exclusion of class 0. Due to the above-mentioned issue with the agricultural data, it is not included in these calculations.

Step 3 consists of two separate cells. Each of these cells generate their own set of tiles from the prepared image and label files. The set of tiles from the first cell will from here on out be referred to as the regular or standard dataset, while the set from the second cell will be referred to as the offset dataset. The regular dataset is generated by first calculating the number of valid tiles that can be generated. This is done because the function used for tiling cannot tile images which have dimensions that are not exact divisible by the calculated number of whole tiles. when the number of valid tiles has been calculated, the excess of the image is cropped out. Because the function starts counting from the top left corner of the image, this means that the excess number of pixels will be cropped on the southern and eastern borders of the image and mask data, see Figure 11.

With the dimensions of the image and mask data adjusted, it can be tiled. The regular tiles have the naming convention “i_j”, where i denotes the tile’s location along the North-South axis relative to the other tiles, while j denotes the tile’s location along the West-East axis relative to the other tiles. Both i and j start at 0 in the North-Western corner of the tiff data. After tiling, the script removes all unwanted tiles by checking all tiles’ mask tile entry values and removing all tiles that only consist of class 0 (No data). These are removed because class 0 is irrelevant for the NN experiment, see Section 7.1. By removing this class, it is less represented, making the model less likely to classify pixels as class 0.

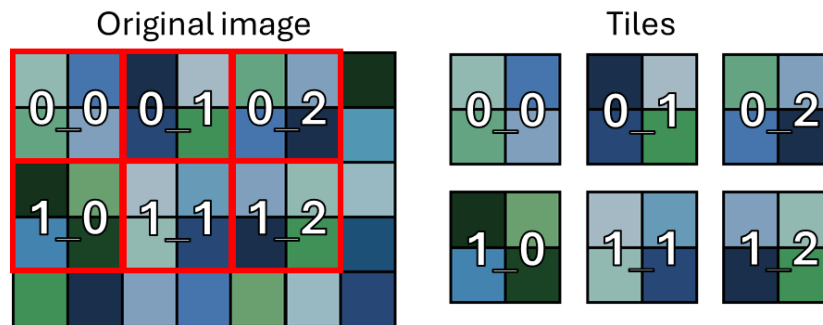


Figure 11: Example of tiling process illustrated on randomly generated 3D matrix.

A similar process is executed for the offset dataset, except the tiling process is moved South and East by half of the tile width and height. This tile set has the naming convention “oi_j”, where o stands for offset while i and j denote the relative location of each tile. This results in a 25% overlap between any given offset tile and the original tiles that they overlap with. This significant overlap is justified by the choice to train the model on a dataset with a distribution of classes based on the actual amount of surface area of each class within the ROI. By overlapping the regular and offset data like this, the model is trained on almost the entire ROI twice.

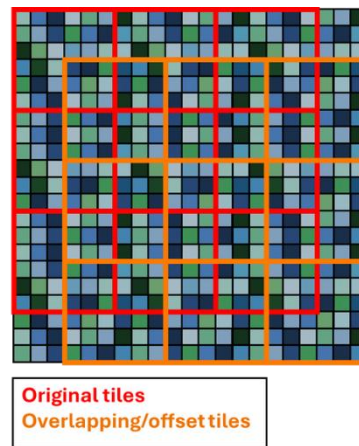


Figure 12 Conceptual visualisation of the overlap between regular and offset tiles

Step 4 splits the data into training and validation data. This step is placed before further augmentation to avoid mirrored or rotated data getting included among the validation data. The dataset is split so that it comes as close as possible to an 80/20 split with 80% training data and 20% validation data.

Denmark is a relatively small ROI compared to other countries. This is especially true when excluding the agricultural Marker dataset, which took up a significant amount of surface area in Denmark for 2020. This is why **step 5** executes data augmentation performed on the image and mask files throughout the following two cells. The first cell does so by mirroring all of the tiles, including the regular and offset tiles. Mirrored tiles receive the prefix “M”. The second cell does so by rotating all four tile set permutations by 90°, 180°, and 270° relative to the original orientation. Tiles receive the prefix “r1_” if rotated by 90°, “r2_” if rotated by 180°, and “r3_” if rotated by 270°. An example of a mirrored, offset tile rotated by 90° would thus be called “r1_Moi_j”. Augmenting the data through offsetting, mirroring and rotating the tiles increase the total number of training tiles from 2876 in the regular tile set to 46568, which is approximately larger by a factor of 16. All of these files are in geotiff format however, which is not storage or VRAM efficient.

Upon attempting to train a simple CNN model on a laptop with only 4GB of VRAM, it was found that the model could not train on such a large number of images. As such, **step 6** was added to convert the tile sets to a more VRAM efficient format. This saved VRAM by inputting data that was already in tensor format instead of forcing the script to spend VRAM during training to convert TIFF data to tensors during the training phase.

Step 7 then cleans up the data. Initial attempts to train the model on the tensor files failed. The received error indicated that this was due to Not a Number (NaN) or negative values in the dataset. Furthermore, it is also recommendable to normalise data before training a model on the data [KILDE FASTAI, p. 241].

Therefore, 3 cells were added. The first cell shows the global minimum and maximum values as well as the total NaN count and dimensions of both the image and label tile tensors. Since no NaN values were found but max values for the image tiles was above 1, all the data was normalised by the global max value. The third cell once again checks the global minimum and maximum values as well as the total NaN count and dimensions of both the image and label

tile tensors, but this time it used the normalised image data to ensure that the data cleaning and normalisation had been done correctly.

Step 8 is a response to the issue of not being able to overwrite the label raster with agricultural data. To allow for experimentation with factor 1, a new set of folders were created in which all entries within class 0 in the image tiles were set to zero.

8.2 Machine Learning

Similarly to the data pre-processing script written in python, the Machine Learning (ML) script can be split into multiple steps consisting of one or more cells. The structure is outlined as follows:

1. Imports
2. Loading of data
3. Define NN model
4. Training/validation loop and visualisation
5. Accuracy assessments

Step 1 is the import of relevant libraries. One particularly notable library is Pytorch. Throughout this master's thesis, two ML libraries were considered and tested. These two are TensorFlow (TF) [TensorFlow, N/A] and PyTorch [PyTorch, e]. Initial attempts at developing ML scripts for the NN experiment were made with TF. However, these attempts were mostly fruitless due to issues with TFs interoperability issues with CUDA. CUDA is a toolkit which allows for acceleration of ML using Graphics Processing Units (GPU) [Nvidia, N/A].

Throughout multiple attempts to set up TF with CUDA, it was found that installing TF in an environment would result in CUDA missing a crucial file. This made it impossible to run ML scripts with GPU acceleration. Upon switching to PyTorch, it was found that CUDA was consistently installed correctly with no crucial files missing. Hence, the ML script and all of its iterations presented in this script are based on PyTorch.

The imports cell is also where methods to improve the reproducibility of the NN experiment were implemented. This includes the global seed discussed in Section 7.3.

Step 2 is the loading of data. This carries two meanings, as the data is both loaded into the script, but it is also in this step that the dataset class is defined. The first cell in this step is the loading of data into the script. First, the directory to the training and validation data are defined for both image and label tile tensors. It is through the definition of the image tile directories that factor 1 is implemented, as changing this directory allows for the switching between images where class 0 has been blacked out or not.

To ensure that the data is usable by the model, a series of tests are executed at this stage. Firstly, the cell checks that all images have a matching label tile and vice versa. This is done both for the training and validation data. Following that, the global min and max entry values as well as the global NaN count for both the training and validation data are found. If the min value is 0 or positive, the max value 1 or lower and the NaN count 0, then the data is suitable for ML training.

The following cell has also been added for trouble shooting. This cell visualises 3 randomly chosen validation image tiles and their corresponding label tiles. This allows for a qualitative

Script Description

assessment of whether the correct dataset is being used with respect to factor 1 and if it has been properly processed before training starts. The third cell in step two defines the dataset class, which helps the NN model to efficiently go through the training and validation data during the training process.

Step 3 is the definition of the NN model. This is where factors 2, 3, 5, 6 and part of factor 7 are implemented. The step consists of a single cell, which starts by specifying the model should be run on the computer's GPU. Following that is an attempt to define the initial weight parameters of the model to further improve reproducibility.

The next part of the cell defines a collection of important controlled and experimental factors in the experiment, including factor 7's number of epochs and factor 6's Learning Rate (LR), before using the dataset loader to define the training and validation data. The next factor to be implemented in the script is factor 3, the model architecture, as the model is defined. In extension, the initial weights are also implemented here along with the optimiser being defined. Factor 5 is implemented next as the `class_weight` tensor is defined before being utilised in the loss function, where factor 2 is also implemented through the `in_` or `exclusion` of `ignore_index=0`.

While it is technically not part of the definition of the NN model, a set of functions are defined so they can save 3 images' actual and predicted labels throughout the training to visualise the progression of the model's training in step 4. While the three images are initially chosen at random, the script will continue to track the same three images throughout an entire training run. As the final part of step 3, two lines for visualisation are added. One can visualise the structure of manually defined models whereas the other can visualise the structure of `torchvision.models` models. However, both have been commented out during the execution of the NN ML scripts to avoid risking that the Jupyter Lab frontend crashing because it runs out of memory. To see the structures of the four models implemented in factor 3, see Appendix E.

9 Results

In this chapter, the results of each factor and its levels will be discussed. This is followed by a section on the reflections related to the lack of complete reproducibility in PyTorch. Afterward, the highest accuracy achieved in the factorial Neural Network (NN) experiment is compared to the results of the Random Forest (RF) model developed in the factorial RF experiment. As stated in Chapter 7, each factor is optimised for one at a time. As such, it is important to note the baseline parameters:

- Factor 1: Black out background class
 - o No
- Factor 2: Exclude background class in loss functions
 - o No
- Factor 3: Model architecture
 - o Encoder: ResNet-50
 - o Decoder: FCN (Fully-Convolutional Network)
- Factor 4: Data augmentation
 - o None
- Factor 5: Class weights in loss function
 - o C0: 1; C1: 1; C2: 1; C3: 1; C4: 1; C5: 1; C6: 1; C7: 1; C8: 1; C9: 1
- Factor 6: Learning Rate
 - o 0.001
- Factor 7: Number of epochs and early stop
 - o Maximum number of epochs: 50
 - o Early stopping
 - Patience: 5
 - Minimum expected improvement over span of patience: 0.001

For each new factor in the experiment, the optimal setting from previously tested factors will be used to ensure that the optimal level for the new factor will be determined based on how well it fits with the chosen levels for previous factors

9.1 Factor 1: Black out background class

The first factor experiment contains two levels. those are whether or not the background class (class 0) has been blacked out to make class 0 more distinct compared to other classes. The iteration for which class 0 has not been blacked out was dubbed F1L0 while the iteration in which class 0 has been blacked out is dubbed F1L1

As mentioned in Section 7.2, the Overall Accuracy (Po) and Cohen's kappa are both calculated with class 0 in- and excluded. This is done for both F1L0 and F1L1. The typical way of calculating Po and kappa is to include all classes. However, the exclusion of class 0 allows for the gauging of the actual performance of the model on the classes that are relevant for this thesis' experiment.

By blacking out class 0, the Po and kappa was improved both when including and excluding class 0 in the calculation of the accuracy measures.

Results

Accuracy measure	Class 0 Included		Class 0 Excluded	
Experiment	F1L0 (No black out)	F1L1 (Class 0 blacked out)	F1L0 (No black out)	F1L1 (Class 0 blacked out)
Po	0.8458	0.8713	0.4857	0.5832
kappa	0.5731	0.6680	0.3546	0.4270

Table 1 The results of experimentation with factor 1. To demonstrate the impact that factor 1 has both for the relevant classes and in general, the accuracy measures are shown with both class 0 (background) included (left) and excluded (right). Notably, this factor's optimal setting improves the measures in both cases.

As stated in Section 7.1, the background class is not within the focus of this master's thesis. Therefore, the choice of optimal level for the factors will be determined from the accuracy measures which exclude class 0. When ignoring class 0, the Po is raised by 9%-points and kappa is raised by 8%-points from F1L0 to F1L1. Therefore, F1L1 is considered the optimal iteration for Factor 1. However, looking at the relative CMs for each level, it is clear that while improvements have been made, the background class still causes significant confusion. Hence, it was deemed necessary to remove the background class altogether through the implementation of factor 2.

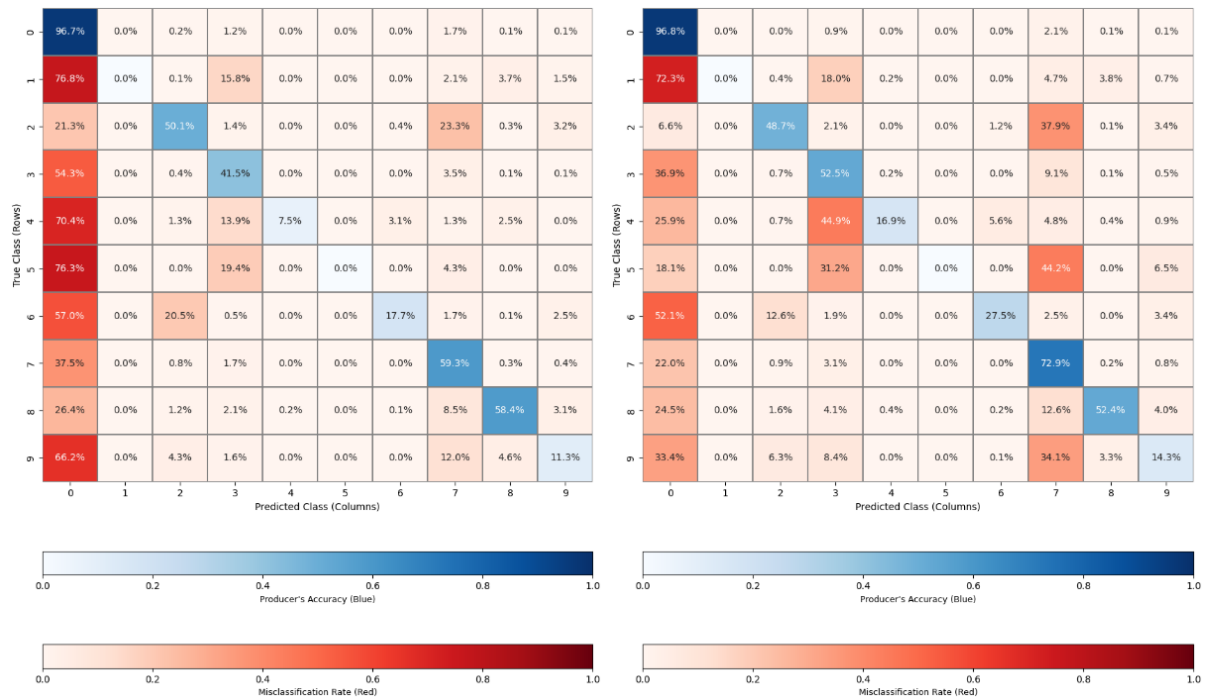


Figure 13 The relative CMs for experiments F1L0 (left) and F1L1 (right). While the relative number of misclassifications where pixels are classified as class 0 (background class) has significantly decreased, Class 0 is still the biggest cause for misclassification among the classes, see Appendix D.

9.2 Factor 2: Exclude Background class in loss functions

The second factor also contains two iterations. The first is dubbed F2L0 and still includes class 0 in the training and validation loss functions. The second, F2L1, ignores class 0 in the loss functions. Similarly to factor 1, the accuracy measures have been calculated with class 0 in- and excluded for both F2L0 and F2L1.

Ignoring class 0 should allow the model to focus on the other classes and thus achieve a greater accuracy overall when ignoring class 0 at the expense of class 0's accuracy.

Accuracy measure	Class 0 Included		Class 0 Excluded	
Experiment	F2L0 (Class 0 not ignored)	F2L1 (Class 0 ignored in loss)	F2L0 (Class 0 not ignored)	F2L1 (Class 0 ignored in loss)
Po	0.8713	0.1924	0.5832	0.7654
kappa	0.6680	0.1305	0.4270	0.6422

Table 2 The results of experimentation with factor 2. To demonstrate the impact that factor 2 has both for the relevant classes and in general, the accuracy measures are shown with both class 0 (background) included (left) and excluded (right).
Notably, the improvement of the relevant classes is at the expense of class 0.

As can be seen from Table 2, ignoring the background class results in a Po %-point increase of 19% and a 21%-point increase for kappa when excluding the background class in the accuracy measures. This accuracy is at the expense of class 0, as can be seen by how much the accuracy measures decrease when including class 0 in the measures. However, because Class 0 is irrelevant to this master's thesis, see Section 7.1, F2L1 is deemed the optimal iteration for factor 2.

At this stage, the accuracy of the Convolutional Neural Network (CNN) model has surpassed that of the RF model [Knudsen, 2025, p. 39]. The following factors will thus be aimed at exploring how high accuracy measures can be achieved with the CNN model.

9.3 Factor 3: Model Architecture

Factor 3 regards the model architecture. For this factor, the architecture has been broken down into two parts, that being the encoder/backbone and the decoder. For each of them, two different levels will be tested. This results in 4 unique levels for this factor. F3L0 tests the accuracy performance of the fcn_resnet50 architecture, F3L1 the deeplabv3_resnet50 architecture, F3L2 the deeplabv3_resnet101 architecture, and F3L3 the fcn_resnet101 architecture. All models are derived from the torchvision.models python library.

Experiment	F3L0 (fcn_resnet50)	F3L1 (deeplabv3_resnet50)	F3L2 (deeplabv3_resnet101)	F3L3 (fcn_resnet101)
Po	0.7654	0.7708	0.7613	0.7817
kappa	0.6422	0.6431	0.6248	0.6612

Table 3 The results of experimentation with factor 3. Due to the irrelevance of class 0, it has been excluded in these accuracy measure calculations.

Interestingly, the most complex architecture (F3L2) has the lowest accuracy measures, with the least complex model (F3L0) being narrowly ahead. While their Pos are close enough that the difference could be due to chance, there is a 2%-point gap in kappa. This indicates that F3L0 is slightly better at least at this stage in the factorial experiment. F3L0 has lower Po and kappa than F3L1, though the difference for both values is so low that the difference could be due to chance for both accuracy measures. However, F3L3 is 1-1.5%-point and close to 2%-points ahead of both F3L0 and F3L1 in Po and kappa respectively. Therefore, F3L3 is chosen as the optimal iteration for factor 3. However, it should be noted that the differences are narrow for all four. Had the order in which factors are optimised for been different in this partial factorial experiment, the order in which each architectures performed might have been different.

9.4 Factor 4: Data augmentation

For factor 4, there was a total of 8 possible levels to test. This was reduced to 5 levels by only testing one combination of augmentation methods, that being all methods combined. The five levels are F4L0 with no augmentation, F4L1 with offset tiles added to the regular dataset, F4L2 with mirrored tiles added to the regular dataset, F4L3 with rotated tiles added to the regular dataset and F4L4 where all three augmentation methods were added to the regular dataset.

Experiment	F4L0 (No augmentation)	F4L1(Off-set)	F4L2 (mirrored)	F4L3 (Rotated)	F4L4 (All methods)
Po	0.7817	0.8415	0.8036	0.8226	0.8484
kappa	0.6612	0.7584	0.6930	0.7286	0.7623

Table 4 The results of experimentation with factor 4. Due to the irrelevance of class 0, it has been excluded in these accuracy measure calculations.

As can be seen from Table 4, all methods improved the accuracy of the model, with the highest accuracy being achieved when applying all methods. Therefore, F4L4 is chosen as the optimal iteration.

9.5 Factor 5: Class weight adjustment in loss functions

The initial goal for this factor was to achieve a minimum of 70% Producer's Accuracy (PA) across all classes except class 0. To reduce the number of runs needed, multiple classes' weights would be adjusted from one run to another. In doing so, the weight of classes which were adjusted based on their proximity to the goal of 70% accuracy, where classes well below would be raised the most. Conversely, classes which were closer to or approximately had met the goal were raised less. Initially, classes which were well above target were not adjusted.



Figure 14 Relative CM for iteration F5L2, see Appendix D.

However, significant amounts of confusion regarding classes 1 and 5 was caused by class 3. This continued through iteration F5L2, despite the weight of classes 1 and 5 having been raised

to 15 and 10 respectively. It was undesired to raise the weights of classes 1 and 5 too significantly compared to classes that did not cause significant confusion for them. Therefore, the weight of classes 3 and, eventually, 7 were lowered throughout iteration F5L3-F5L6. This along with the continued raising of the weights of underrepresented classes had a growing negative impact on other classes.

Experiment	F5L0	F5L1	F5L2	F5L3	F5L4	F5L5	F5L6	F5L7
Weights	C0: 0 C1: 1 C2: 1 C3: 1 C4: 1 C5: 1 C6: 1 C7: 1 C8: 1 C9: 1	C0: 0 C1: 7 C2: 4 C3: 1 C4: 4 C5: 5 C6: 1 C7: 1 C8: 2 C9: 3	C0: 0 C1: 15 C2: 4 C3: 1 C4: 4 C5: 10 C6: 1 C7: 1 C8: 3 C9: 3	C0: 0 C1: 30 C2: 4 C3: 0.8 C4: 4 C5: 15 C6: 2 C7: 1 C8: 4 C9: 5	C0: 0 C1: 50 C2: 4 C3: 0.6 C4: 5 C5: 20 C6: 2 C7: 1 C8: 5 C9: 5	C0: 0 C1: 50 C2: 4 C3: 0.4 C4: 5 C5: 20 C6: 2 C7: 1 C8: 4 C9: 5	C0: 0 C1: 50 C2: 4 C3: 0.2 C4: 5 C5: 20 C6: 2 C7: 0.5 C8: 4 C9: 5	C0: 0 C1: 200 C2: 4 C3: 0.2 C4: 5 C5: 20 C6: 2 C7: 0.5 C8: 4 C9: 5
Po	0.8484	0.8536	0.8446	0.8532	0.8042	0.8311	0.7981	0.7714
kappa	0.7623	0.7795	0.7645	0.7806	0.7134	0.7445	0.7060	0.6640

Table 5 The results of experimentation with factor 5. Due to the irrelevance of class 0, it has been excluded in these accuracy measure calculations.

By iteration F5L6, class 5 had reached the 70% Producer's Accuracy (PA) goal while the Po and kappa of the model had fallen by 5% and 6% respectively compared to iteration F5L0. With class 1 only having achieved a PA of 37%, a large raise in weight was made to see if it was realistically possible to raise the accuracy of class 1 to the 70% target. By setting the weight of class 1 to 200, a factor 1000 larger than the weight of class 3, a PA of 57% was reached for class 1 at the expense of other classes.

As such, it was chosen to use kappa and Po as response variables instead of PA. Upon analysing the Po and kappa values achieved for factor 5, Iterations F5L1 and F5L3 were deemed the best, though close enough in performance that the difference was negligible. As such, the choice was made by looking at which of these two highest performing models had fewest classes with a PA below the initial 70% goal, excluding class 0. F5L1 has three classes below the target while F5L3 only has two below the target. Hence, iteration F5L3 is chosen as the optimal level for this factor among the tested levels.

9.6 Factor 6: Learning rate

For this factor, there is an endless possibility of levels. To set a reasonable range for finding a local optimal value for the factor, the model was initially run with the Learning Rate (LR) set to the baseline value multiplied and divided by a factor 10 respectively.

Both F6L1 and F6L2 resulted in lower accuracies than the baseline F6L0. This indicates that a local optimal value has been found in the vicinity of F6L0. Thus, it was chosen to test an LR value between the baseline value of F6L0 and F6L1-2 respectively, resulting in F6L3-4. F6L3 and F6L4 both performed better than the extremity levels but worse than the baseline.

Results

Experiment	F6L0 (LR = 1e-3)	F6L1 (LR = 1e-4)	F6L2 (LR = 1e-2)	F6L3 (LR = 5e-4)	F6L4 (LR = 5e-3)	F6L5 (LR = 2e-3)	F6L6 (LR = 15e-4)
Po	0.8532	0.7922	0.7693	0.8172	0.8390	0.8277	0.8287
kappa	0.7806	0.6950	0.6694	0.7257	0.7611	0.7459	0.7471

Table 6 The results of experimentation with factor 6. Due to the irrelevance of class 0, it has been excluded in these accuracy measure calculations.

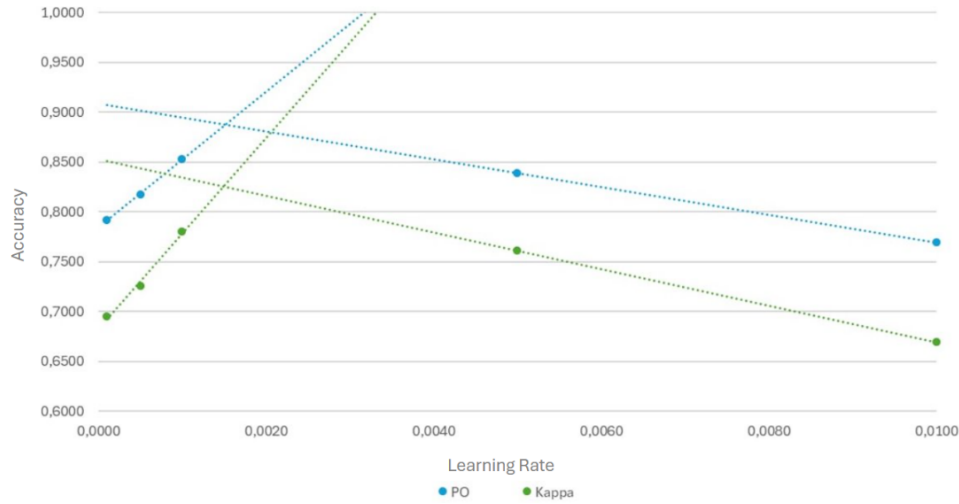


Figure 15 A plotting of the Overall Accuracy (blue) and kappa (green) for F6L0-4 in relation to the corresponding LR levels. Linear regressions have been made for groupings of points to aid in pin-pointing a potential optimal level.

F6L4 was at this stage the closest to F6L0 in accuracy for both Po and kappa. Thus, it was hypothesised that the optimal level might lie in the range between the LR values of F6L0 and F6L4. To further pinpoint where a potential optimal value might be, the accuracies of F6L0-4 were plotted in a graph. A relatively steep increase in accuracy can be seen for the three lower levels in the case of both accuracy measures. While the sample size is rather small, this appears to roughly fit with a linear regression. As such, in an attempt to approximate where this potential optimal point might be, linear regressions were made for both the grouping of three lowest levels and the grouping consisting of the remaining two levels. The crossing point for both Po and kappa lies approximately at $LR = 15e-4$.

As a result, two more levels were tested. These levels were F6L5 with $LR = 2e-3$ and F6L6 with $LR = 15e-4$. $2e-3$ was chosen for two reasons. Because of the small sample rate, the crossing point method was inaccurate. Furthermore, the crossing point is significantly closer to F6L0 than F6L4. This could be in part be because there is more samples in the grouping which F6L0 belongs to. Thus, $2e-3$ serves as a compromise between the crossing point and the median LR value of the two highest performing levels thus far. However, both F6L5 and F6L6 were less accurate than F6L0 and F6L4. Thus, F6L0 was chosen as the optimal iteration for factor 6.

9.7 Factor 7: Number of epochs

With all other experiment factors optimised for, the goal of the last factor was to explore if a higher accuracy could be achieved by practically disabling early stopping and increasing the number of epochs. Therefore, the max number of epochs was raised from 50 to 100 and the patience set to 100. If the validation loss was found to oscillate significantly or have a

decreasing tendency within these 100 epochs, the model would be tested again with more epochs.

As can be seen from Figure 16, the increased number of epochs did not result in an improved validation loss. The epoch with the lowest validation loss in both cases is epoch 14, which equates to the 15th epoch that was run in both iterations.

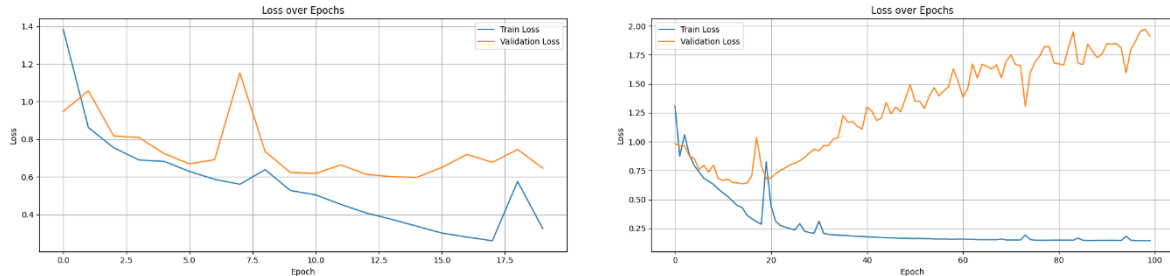


Figure 16 Visualisation of the loss curves for F7L0 (left) and F7L1 (right). In both cases, the lowest validation loss occurred at epoch 14 (equating to the 15th epoch).

Given the general increasing tendency of the validation loss curve for iteration F7L1, it was concluded that increasing the number of epochs would not result in a lower validation loss. Thus, no more levels were tested for factor 7. Despite the optimal epoch being the same for both iterations, the accuracy was not the same for each iteration. The Po and kappa for iteration F7L1 is approximately 2%-points and 3%-points lower than for iteration F7L0. This indicates that, despite the implementation of seeds, there is still a noticeable degree of pseudo-randomness within the code. This fits with what was discussed in Section 4.4.4.

Experiment	F7L0	F7L1
Epoch with lowest validation loss	Epoch 14 (15 th epoch)	Epoch 14 (15 th epoch)
Po	0.8532	0.8341
kappa	0.7806	0.7529

Table 7 The results of experimentation with factor 7. Due to the irrelevance of class 0, it has been excluded in these accuracy measure calculations.

As a result, Iteration F7L0 is deemed to be the optimal iteration of this factor, not due to the factor level, but due to chance. Thus, F7L0 is concluded to be the most accurate iteration achieved within the factorial experiment of this master's thesis.

9.8 Reliability of results

As stated in Section 4.4.4, complete reproducibility is not guaranteed in PyTorch. Therefore, it is necessary to reflect on the significance this has on the reliability of the specific choice of optimal factor levels.

The incongruence seen in Factor 7 seems to suggest there is a sampling error in the experiment. Because it is directly stated in PyTorch's documentation on seeds that there is not a guarantee of reproducibility, see Section 4.4.4, it is assumed that this is the source of error which causes the script's results to not be perfectly reproducible. The ideal solution to this would be to run each iteration of the script multiple times and either take the highest or average accuracy values. However, due to how time consuming it would be to repeat each iteration's training process multiple times, it is outside of the scope of this project.

Results

Therefore, the reliability of the factor experiments will be determined based on the observations of Factor 7. Factor 7 only compares two runs. Therefore, the sample size is too small for calculating a reasonable standard deviation. However, by comparing the best performing iteration from each factor to their second-best performing iteration, it is found that factors 3-6 all have a lower difference in kappa than factor 7, see Table 8. This would suggest that all of these factors would benefit from having multiple reruns to mitigate the impact of PyTorch's non-deterministic design.

Notably, the technical fixes implemented in factors 1 and 2 have increased the accuracy significantly more than the difference found for factor 7. This would suggest that the chosen optimal level for these two factors is reliable. However, it is uncertain if their impact comes from them belonging to a different type of factor than the other factors or because they came first. However, the significant difference in improvement for factors 1 and 2 compared to the remaining factors and the fact that the entries of Table 8 are not steadily decreasing indicates that the order of factors is less significant than whether the factor was aimed at fixing technical issues with the input data or not.

Accuracy measure	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7
Difference	8	21	2	0.4	0.1	2	3

Table 8 The %-point difference between the highest and second highest kappa achieved in each experiment.

Despite the less reliable basis for comparison of optimal level choice, the choice of optimal factors will be retained based on the data that was collected. Furthermore, this challenge does not impact the comparison to the RF model, as the Accuracy achieved with the CNN model is still reliable as an absolute measure of accuracy based on the chosen factor levels. This is regardless of whether the chosen factor levels are the optimal ones or not.

It is also important to note that Table 8 only shows that more data would be needed to definitively state whether some factors' two highest performing levels are the optimal settings. Table 9 shows that most factors had a significant impact on the accuracy of the model. By calculating the difference between the highest and lowest kappa values for each level, it shows that factor 7 had the lowest difference. All other factors except factor 3 have a relatively significantly bigger difference than that of factor 7. This shows that the majority of chosen experimental factors likely have an actual impact on the model, but that some factor levels are closer to each other in performance than other levels. This warrants more reruns for these factors to truly determine the optimal level for these factors.

Accuracy measure	Factor 1	Factor 2	Factor 3	Factor 4	Factor 5	Factor 6	Factor 7
Difference	8	21	4	10	12	11	3

Table 9 The %-point difference between the highest and lowest kappa achieved in each experiment.

9.9 Factor levels after experimentation

As a result of the partial factorial experiment, the final factor levels are as follows:

- Factor 1: Black out background class
 - o Yes
- Factor 2: Exclude background class in loss functions

- Yes
- Factor 3: Model architecture
 - Encoder: ResNet-101
 - Decoder: FCN (Fully-Convolutional Network)
- Factor 4: Data augmentation
 - Offset
 - Mirroring
 - Rotation
- Factor 5: Class weights in loss function
 - C0: 0; C1: 30; C2: 4; C3: 0.8; C4: 4; C5: 15; C6: 2; C7: 1; C8: 4; C9: 5
- Factor 6: Learning Rate
 - 0.001
- Factor 7: Number of epochs and early stop
 - Maximum number of epochs: 100
 - Early stopping
 - Patience: 5
 - Minimum expected improvement over span of patience: 0.001

9.10 Comparing CNN output with RF output

With the factorial experiment of this thesis completed, it was possible to compare the accuracies of the resulting CNN model to that of the RF model which was previously developed. Judging by the single value measures seen in Table 10, The CNN model is a significant improvement over the RF model. With a Po of 85%, the CNN model outperforms the RF model by an absolute amount of 11% points, equating to a relative improvement of 15%. The most significant single value measure improvement is kappa. This is not only because it compensates for chance agreement, which gives deeper insight in the performance model. It is also because the CNN model, with a kappa of 0.78, outperforms the RF model absolutely by 0.15. This equates to a 24% relative improvement.

Accuracy measure (Excluding class 0)	CNN model	RF model	Absolute improvement	Relative improvement
Po	85%	74%	11%	15%
kappa	0.78	0.63	0.15	24%

Table 10 The single value accuracy measures for the best iteration of both the CNN and RF model. Absolute and relative improvement show how much better the CNN model performed than the RF model.

To get a better sense of how the CNN model outperforms the RF model, the PA was compared as well, see Table 11. Note that due to the inclusion of a background class, the class labels of the CNN model are offset compared to the original RF model's classes. For the PA of each model, it was found that the CNN model outperforms the RF model to various degrees in all but 2 classes. Those two classes are Class 1 (Bassins), which achieved an accuracy of equivalent to that of the RF model at 25%, and class 8 (Lakes), which decreased in accuracy by 3%. This is an interesting coincidence. However, there is no significant reason to draw conclusions of correlation. As discussed in Section 9.5, it was possible to raise the accuracy of class 1 to 57%, but due to the relatively small representation of class 1, this was at the expense of other classes, hence why class 1 was not further prioritised. As for class 8, the accuracy

Results

could have been raised higher as well if assigned a higher weight, but because of the initial goal for factor 5 being that all classes had a PA of at least 70, class 8 was not further prioritised. Therefore, it seems reasonable to suggest the hydrological classes have accuracies that are equal to or worse than those of the RF model because of the choices made in Factor 5, not because the hydrological classes are inherently harder to classify. However, all other classes' PAs were improved to various extents compared to the performance of the RF model.

The classes with the lowest positive PA improvements are classes 5 (recreative areas) and 7 (Forest) with improvements of 7% and 1% respectively. The remaining classes all had two-digit relative improvements, with class 3 (urban areas) and 6 (sand/dune) both improving by 13%, classes 2 (moorland) and 9 (wetlands) improving by at least 60% and class 4 (raw materials) improving by 81%. Notable is that for classes with a relative representation of approximately 0.5% and above, the PA for the CNN model was above the initial 70% goal. This means that the CNN model had 7 out of 9 classes achieve the 70% goal, compared to only 4 classes achieving a PA of 70% or higher in the case of the RF model.

Class (CNN)	Relative representation (for CNN)	CNN model PA	RF model PA	Absolute improvement	Relative improvement
0	N/A	0%	N/A	N/A	N/A
1 (0 for RF)	0.06%	25%	25%	0%-points	0%
2 (1 for RF)	6.69%	80%	50%	30% -points	60%
3 (2 for RF)	29.97%	89%	79%	10% -points	13%
4 (3 for RF)	0.46%	76%	42%	34% -points	81%
5 (4 for RF)	0.03%	61%	57%	4% -points	7%
6 (5 for RF)	0.82%	81%	72%	9% -points	13%
7 (6 for RF)	48.60%	86%	85%	1% -points	1%
8 (7 for RF)	5.35%	74%	76%	-2% -points	-3%
9 (8 for RF)	8.00%	82%	51%	31% -points	61%

Table 11 The PA for the best iteration of both the CNN and RF model. Absolute and relative improvement show how much better the CNN model performed than the RF model. Note that the class names are based on those used for the CNN. Class 0 (background) is excluded due to the RF model not having a background class

10 Conclusion

10.1 Initial wonder

This thesis came to be as a result of the underwhelming Random Forest (RF) model produced in my 9th semester project report. Through a desire to achieve better results than those managed with the RF model, an initial wonder was posed:

“What is considered state of the art for Machine Learning (ML) and semantic segmentation and how can it be implemented in the context of Land Use/Land Cover (LULC) map generation?”

In order to answer this question, various literature was consulted. This formed the foundation for the literature study presented in Chapter 4. Through this literature study, a general consensus was found which points to Neural Networks (NN) being more suitable for Computer vision than traditional methods such as Decision Tree Ensembles. Furthermore, literature from 2020 and onward points to Convolutional Neural Networks (CNN) and variations thereof as being the most suitable NN models for achieving high accuracy measures when working with Computer Vision. This is because the convolutional and pooling layers are specifically designed to deal with image processing in a way that reduces the required processing power and complexity of NN models. This can both decrease the risk of overfitting and/or free up processing power for other processes during training.

The literature study has also aided in the understanding and selection of factors to experiment with. The categories of factors explored throughout the report include factors that relate to the architecture, training phase, and regularisation of a CNN model as well as other miscellaneous factors. While the majority of these factors were related to improve the accuracy measures of a CNN model, the miscellaneous category in particular regarded factors which focused more significantly on the technical optimisation of NN models.

10.2 Problem formulation

Based on the knowledge that was gained from the pre-analysis, a problem formulation was developed:

“Is it possible to achieve a higher accuracy of LULC classification with CNN than previously found with RF?”

With the following sub questions:

10.2.1 Which level for each of the experimental factors is optimal for achieving the highest accuracy possible?

The non-deterministic nature of PyTorch does make the results less reliable than desired. From the results of these experiments, the following optimal settings were found, though reruns would be required to confirm the results.

In the case of the technical fixes, it was found that the inclusion of both factors 1 and 2 had a positive impact. The significant differences between the accuracies of each level for each factor indicates the choice of optimal factor is more reliable. Thus, the optimal setting for factor 1 was found to be when black-out of the background class was included. For factor 2 the optimal setting was the exclusion of the background class in loss functions.

Conclusion

The optimal choice for the remaining factors is less clear though as the two top-performing levels for these factors were relatively close in performance. However, it should be noted that the range in accuracy for most factors is significant enough to demonstrate that the choice of level for these factors is of importance. This is demonstrated through factors 4-6 all having an accuracy range of 10-12%-points for Choen's kappa (kappa) from the lowest to highest performing setting in this experiment. Thus, the uncertainty does not lie in whether these factors are relevant or not to the experiment, but in the proximity of accuracy measures for the top-performing level for Factors 3-7.

Factor 3 regarded the choice of CNN architecture. The optimal architecture among the four tested was found to be the `fcn_resnet101`. It was found that both the most and least complex model were both outperformed. Whether this is due to the other model having an ideal of complexity to avoid overfitting, the order in which the factors are optimised for, the lack of reproducibility, or a combination hereof is unclear.

Factor 4 regards data augmentation, for which it was found that the highest accuracies were achieved when implementing all of the included augmentation methods.

Factor 5 regarded the weighting of each class in the loss functions. While a majority of classes could achieve a Producer's Accuracy (PA) of 70%, two classes were unable to achieve this goal without negatively impacting other classes too significantly. These were class 1, Bassins, with a PA of 25% and class 5, Recreative Areas, with a PA of 61%. Notably, these classes had a relative representation in the dataset of 0.06% and 0.03% respectively. The least represented class to achieve a 70% PA is class 4, Raw Materials, with a relative representation of 0.46%.

Factor 6 regards the choice of Learning Rate (LR). Despite being the factor with the second most levels tested, the optimal level turned out to be the initial level, that being an LR of $1e-3$. Attempts were made to predict potentially higher performing levels after the runs with the first 5 levels, yet the initial level remained the best performing.

Factor 7 was an attempt to see if the model could eventually reach a lower validation loss if allowed to run for enough epochs. Not only did it prove to not be the case, but it also highlighted the non-deterministic nature of PyTorch. It did so because the lowest validation loss was achieved at the exact same epoch as when early stopping had been implemented yet reached a lower accuracy when early stopping was disengaged. Because the highest accuracy was achieved when Early stopping was engaged, this was the run which was chosen as the optimal iteration, but this conclusion is not based on the implementation of early stopping, as this did not impact the accuracy

10.2.2 What is the significance of the interaction between the model and its input data?

The scope of this thesis' factorial experiment is too small to reach a clear and definitive answer. However, Factors 1 and 2 show that the quality of input data and the extent to which the data and model have been tailored for each other are of significant importance.

Factor 1 showed the impact of increased data quality the cleaning the input images. This was done by blacking out all data within the background class. This improved the Po by 10%-points and the kappa by 8%-points.

The most significant impact came from Factor 2, which improved the accuracy of the CNN model with 18%-points for Po and 21%-points for kappa. It does so by adapting the loss functions to the unique properties of the input data.

As such, the results of this factorial experiment showed that the optimisation of the interaction between the input data and CNN model equated to a high 30%-point improvement. This would suggest that the adaption of the model and the data for optimal interaction is significant.

10.2.3 To what extent can the NN model outperform the RF model?

As seen in Section 9.10, the absolute single value measure improvements of the CNN model over the RF model are 11%-points for Po and 0.15 for kappa. In relative terms, this translates to a 15% improvement in Po and 24% improvement. Furthermore, it was found that all but two classes had improved when comparing the CNN model based on the vector accuracy measure PA. For the PA, it was found that all classes had improved to some extent except the two hydrological classes. However, there was found no reason to believe that hydrological fields are particularly hard to recognise. Instead, the reason likely stems from these classes not being prioritised further during the experimentation with factor 5. Furthermore, all classes with a relative data representation higher than 0.5% reached a PA value of at least 70% in the case of the CNN. This equates to 7 out of 9 classes achieving a PA of 70% or higher compared to just 4 classes for the RF model.

10.2.4 Is it possible to achieve a higher accuracy of LULC classification with CNN than previously found with RF?

With basis in the answers found for the three sub-questions, the answer to this problem formulation is not only yes, but also that the difference achieved is significant.

11 Reflection: Potential improvements

While the goal of this master's thesis was achieved, there is still room for improvement. This section will cover various improvements that could have been made if the scope of this project was bigger.

11.1 Cross examination and full factorial experiment

It is proven that the accuracy of the Convolutional Neural Network (CNN) model achieved through this thesis' factorial CNN experiment is higher than that of the previous report's Random Forest (RF) experiment. However, it is not proven that the choice of optimal levels for each factor is the actual optimal levels, as discussed in Section 9.8. Furthermore, the exact impact of each factor is not known either. This is due to a mix of PyTorch not having full reproducibility and each level only being tested for once.

The ideal solution to this issue would have been to test all permutations of the factors and their levels. This method would have determined the actual optimal levels for each factor when the factors interact. However, this is well outside the scope of this project due to how time consuming such a process would be, especially considering that each permutation would have to be run multiple times to account for the lack of perfect reproducibility.

A less time requiring approach could have been to repeat the optimisation process but reversing the order in which the factors were optimised for. This won't determine the optimal level like testing all permutations would. However, it can improve the reliability of the findings of this thesis if it results in the same levels being chosen. Conversely, if other levels are found to be optimal when reversing the order, it proves that further experimentation would be needed. Either way, the reliability of this thesis would be determined while significantly reducing the required expansion of the scope. By testing the factors in a different order, it also helps determining whether the order of optimisation affects the impact that each factor has on the accuracy of the CNN model.

11.2 Separate training and validation regions

Initial planning of this project included a thorough approach to splitting training and validation data into appropriate regions. However, because of the long runtimes when processing data in GIS, the training/validation split was chosen to be a random split process. However, this means that there is no designated training vs. validation regions. This might make the model appear better at generalising than it actually is. It also means that each class' relative representation for the entire dataset might not be reflected in the training and validation sets respectively.

An approach to mitigate this could be to split the entire Region of Interest (ROI) into a group of sub-regions, for which the relative distributions of classes were calculated. The 20% of sub-regions that come closest to matching the relative class distribution when added together would be made the validation region. This would then leave the other 80% as the training regions.

For this method, it would be necessary to consider the ideal sub-region size. The larger the size of sub-regions, the more separated training regions will be. However, it would also make the resulting distribution less likely to have a class representation that is close to the total class distribution. Conversely, smaller sub-regions will give a greater number of permutations to test. This would increase the chances of there being a combination of sub-regions with a class

representation that better represents the total class distribution. However, this will also likely be at the cost of separation between training and validation regions. Additionally, the more regions there are, the more permutations there are to test, and thus the computational time or hardware requirements will be bigger.

To decrease computational requirements, it could be sensible to take inspiration from early stopping. This could be done by setting a value for how close the class representation of a given combination of sub-regions must be to the total class distribution. This could potentially be expanded by assigning specific limits based on the degree to which each class is under- or overrepresented in the total class distribution. One approach to this could be that classes with less representation in the total data can deviate less than bigger classes to ensure a reasonable representation of smaller classes in both the training and validation data.

11.3 Assembly of a *Land Use/Land Cover (LULC)* map

The development of a complete map is not necessary for this thesis. However, if this model were to be used in practice, it would be necessary to generate entire maps of a larger size than any individual tile. This is the reason for why the naming convention is based on location instead of assigning a single number. The naming convention of i_j would theoretically allow for the merging of tiles into a coherent map. However, because of the conversion from geotiff to tensor, the tiles lose their coordinates. This means that the resulting map won't be georeferenced and thus can't be used in any GIS application. To accommodate this issue, the naming convention of the merged tiles can also be used to transfer the coordinates of the original geotiff tiles to the merged output tiles. This would result in a merged, georeferenced raster file ready for use in GIS applications.

12 Bibliography

Alzubaidi et.al., 2021. Alzubaidi, L., Zhang, J., Humaidi, A.J. et al. *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*. J Big Data 8, 53. <https://doi.org/10.1186/s40537-021-00444-8>, 2021. Accessed: 2025-05-07.

Arler et al., 2017. Finn Arler, Michael Søgaard Jørgensen and Esben Munk Sørensen. *Prioritering af Danmarks areal i fremtiden*. ISBN: 978-87-91614-67-5, N/A. Fonden Teknologirådet, 2017.

Chen et.al., 2017. Liang-Chieh Chen, George Papandreou, Florian Schroff, Hartwig Adam. *Rethinking Atrous Convolution for Semantic Image Segmentation*. <https://arxiv.org/pdf/1706.05587>, 2017. Accessed: 2025-05-02.

Christensen and Eetvelde, 2024. A. A. Christensen and V. Van Eetvelde. *Decision making in complex land systems*. <https://link.springer.com/article/10.1007/s10980-024-01822-2>, 2024.

Danmarks Meteorologiske Institut, N/A. Danmarks Meteorologiske Institut. *Havet stiger omkring Danmark*. <https://www.dmi.dk/klima-atlas/om-klima-atlas/havetstigeromkringdanmark>. Accessed: 2025-03-26.

Dataforsyningen, a. Dataforsyningen. *Kommuneinddeling*. <https://dataforsyningen.dk/data/3901>. Accessed: 2025-05-07.

Ejrnæs et al., 2022. Rasmus Ejrnæs, Jesper Bladt and Camilla Fløjgaard. *Potentialet for at reservere 30 % af landarealet til beskyttede og strengt beskyttede områder i Danmark*. ISBN: 978-87-7156-706-9. Aarhus Universitet, DCE – Nationalt Center for Miljø og Energi, 2022

ESA, 2025. European Space Agency. *Data*. <https://esa-worldcover.org/en/data-access>, 2025. Accessed: 2025-05-16

European Commission. European Commission. *Biodiversity strategy for 2030*. https://environment.ec.europa.eu/strategy/biodiversity-strategy-2030_en, 2025. Accessed: 2025-03-26.

Galliers, 1991. R. Galliers. *Choosing Appropriate Information Systems Research Approaches: A Revised Taxonomy*. The Information Research Arena of the 90s. 155–173, 1991.

GEE, a. Earth Engine Catalog. *Landsat 8: OLI/TIRS*. <https://developers.google.com/earth-engine/datasets/catalog/landsat-8/>, a. Accessed: 2025-05-06.

GEE, b. Earth Engine Catalog. *USGS Landsat 8 Level 2, Collection 2, Tier 1*. https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C02_T1_L2#bands, b. Accessed: 2025-05-13.

GEE, c. Earth Engine Catalog. *USGS Landsat 8 Collection 2 Tier 1 TOA Reflectance*. https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC08_C02_T1_TOA#bands, c. Accessed: 2025-05-13.

- Gholamalinezhad and Khosravi, 2020.** Hossein Gholamalinezhad and Hossein Khosravi. *Pooling Methods in Deep Neural Networks, a Review*. <https://arxiv.org/abs/2009.07485>, 2025. Accessed: 2025-05-07.
- Guo et.al., 2020.** H. Guo, M. Goodchild and A. Annoni. *Manual of Digital Earth*. ISBN: 978-981-32-9915-3. Springer Nature, 2020.
- Howard and Gugger, 2020.** Jeremy Howard and Sylvian Gugger. *Deep Learning for Coders with fastai & PyTorch: AI Applications without a PhD*. ISBN: 978-1-492-04552-6. O'Reilly, 2020.
- Klimadatastyrelsen, N/A.** Klimadatastyrelsen. *Inspire - Arealanvendelse*. <https://dataforsyningen.dk/data/996>, N/A. Accessed: 2025-03-31.
- Knudsen, 2025.** E. Knudsen. *Land Use & Land Cover Classification of Aalborg Municipality*. https://kdbk-aub.primo.exlibrisgroup.com/permalink/45KBDK_AUB/n411aj/alma9921966558205762, 2025.
- Kufel et.al., 2023.** J. Kufel, K. Bargieł-Łączek, S. Kocot, M. Koźlik, W. Bartnikowska, M. Janik, Ł. Czogalik, P. Dudek, M. Magiera, A. Lis, I. Paszkiewicz, Z. Nawrat M. Cebula and K. Gruszczyńska. *What Is Machine Learning, Artificial Neural Networks and Deep Learning? —Examples of Practical Applications in Medicine*. <https://doi.org/10.3390/diagnostics13152582>, 2023.
- Landis and Koch, 1977.** J. Richard Landis and Gary G. Koch. *The Measurement of Observer Agreement for Categorical Data*. <https://www.jstor.org/stable/2529310>, 1977. Accessed: 2025-03-31.
- Lundby et.al, 2023.** Erlend Torje Berg Lundby, Haakon Robinson, Adil Rasheed, Ivar Johan Halvorsen, and Jan Tommy Gravdahl. *Sparse neural networks with skip-connections for identification of aluminum electrolysis cell*. https://ntnuopen.ntnu.no/ntnu-xmloi/bitstream/handle/11250/3124870/sysid_sparse_skip_connections.pdf?sequence=1, 2023. Accessed: 2025-05-02.
- Melinda et.al, 2024.** Melinda Melinda, Hurriyatul Aqif, Junidar, Maulisa Oktiana, Nurlida Basir, Afdhal Afdhal and Zulfan Zainal. *Image Segmentation Performance using Deeplabv3+ with Resnet-50 on Autism Facial Classification*. <https://pdfs.semanticscholar.org/b3d0/b8b4db1e3f943fb1ac734e828121720645b3.pdf>, 2024. Accessed: 2025-05-02.
- Minaee et.al., 2021.** Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. *Image Segmentation Using Deep Learning: A Survey*. <https://www.porikli.com/mysite/pdfs/porikli%202021%20-%20Image%20segmentation%20using%20deep%20learning%20-%20A%20survey.pdf>, 2021. Accessed: 2025-05-16.
- Ministry of Food, Agriculture and Fisheries of Denmark.** Ministry of Food, Agriculture and Fisheries of Denmark. *Grøn trepart*. <https://fvm.dk/arbejdsomraader/landbrug/groen-trepart>, 2025. Accessed: 2025-03-18.

Bibliography

- Navigating 360, 2024.** Navigating 360. *Når stormfloderne rammer*. https://www.navigating360.dk/files/ugd/06ce87_ff1072c08d3743788cdb50f8edc6a436.pdf, 2024. Accessed: 2025-03-18.
- Nvidia, 2025.** Nvidia. *NVIDIA RTX Blackwell GPU architecture: Built for Neural Rendering*. <https://images.nvidia.com/aem-dam/Solutions/geforce/blackwell/nvidia-rtx-blackwell-gpu-architecture.pdf>, 2025. Accessed: 2025-05-11.
- Nvidia, N/A.** Nvidia. *CUDA Toolkit*. <https://developer.nvidia.com/cuda-toolkit>, N/A. Accessed: 2025-05-07.
- O'Shea and Nash, 2015.** Keiron O'Shea and Ryan Nash. *An Introduction to Convolutional Neural Networks*. <https://arxiv.org/pdf/1511.08458>, 2015. Accessed: 2025-05-01.
- Petersen et al., 2024.** Anders Højgård Petersen, Berit Hasler, Thomas Laage-Thomsen, Mette Termansen and Carsten Rahbek. *Mere, bedre og større natur i Danmark. Hvor, hvordan og hvor meget?* ISBN: 978-87-972724-1-1. Center for Makroøkologi, Evolution og Klima, 2024.
- PyTorch, a.** PyTorch. *Torchvision.models*. <https://docs.pytorch.org/vision/0.9/models.html#semantic-segmentation>, a. Accessed: 2025-05-02.
- PyTorch, b.** PyTorch. *CrossEntropyLoss*. <https://docs.pytorch.org/docs/stable/generated/torch.nn.CrossEntropyLoss.html>, b. Accessed: 2025-05-06.
- PyTorch, c.** PyTorch. *Automatic Mixed Precision*. https://docs.pytorch.org/tutorials/recipes/recipes/amp_recipe.html, c. Accessed: 2025-05-11.
- PyTorch, d.** PyTorch. *Reproducibility*. <https://docs.pytorch.org/docs/stable/notes/randomness.html>, d. Accessed: 2025-05-11.
- PyTorch, e.** PyTorch. *EarlyStopping*. https://docs.pytorch.org/ignite/generated/ignite.handlers.early_stopping.EarlyStopping.html, e. Accessed: 2025-05-21.
- Santos and Papa, 2022.** Claudio Santos and João Papa. *Avoiding Overfitting: A Survey on Regularization Methods for Convolutional Neural Networks*. <https://arxiv.org/pdf/2201.03299>, 2022. Accessed: 2025-05-08.
- Scikit learn, N/A.** Scikit learn. *Early stopping in Gradient Boosting*. https://scikit-learn.org/stable/auto_examples/ensemble/plot_gradient_boosting_early_stopping.html, N/A. Accessed: 2025-05-11.
- Smith, 2018.** Leslie N. Smith. *A disciplined approach to neural network hyperparameters: Part 1 – Learning rate, batch size, momentum, and weight decay*. <https://arxiv.org/pdf/1803.09820>, 2018. Accessed: 2025-04-22.
- Stehman, 1997.** Stephen V. Stehman. *Selecting and interpreting measures of thematic classification accuracy*. <https://www.sciencedirect.com/science/article/pii/S0034425797000837>, 1997. Accessed: 2025-03-31.

TensorFlow, N/A. TensorFlow. *Install TensorFlow 2*. <https://www.tensorflow.org/install>, N/A. Accessed: 2025-05-07.

United Nations. United Nations. *Causes and Effects of Climate Change*: <https://www.un.org/en/climatechange/science/causes-effects-climate-change>. Accessed: 2025-03-18.

USGS, a. USGS. *LandSat 7*. <https://www.usgs.gov/landsat-missions/landsat-7>, a. Accessed: 2025-03-31.

USGS, b. USGS. *LandSat 8*. <https://www.usgs.gov/landsat-missions/landsat-8>, b. Accessed: 2025-03-31.

USGS, c. USGS. *LandSat 9*. <https://www.usgs.gov/landsat-missions/landsat-9>, c. Accessed: 2025-03-31.

Wu et.al., 2019. Yanzhao Wu, Ling Liu, Juhyun Bae, Ka-Ho Chow, Arun Iyengar, Calton Pu, Wenqi Wei, Lei Yu, Qi Zhang. *Demystifying Learning Rate Policies for High Accuracy Training of Deep Neural Networks*. <https://arxiv.org/pdf/1908.06477>, 2019. Accessed: 2025-04-21.

Yang and El-Haik, 2003. Kai Yang and Basem El-Haik. *Design for Six Sigma: A Roadmap for Product Development*. DOI: 10.1036/0071435999. McGraw-Hill, 2003.

13 Abbreviations

AAU =	Aalborg University
AI =	Artificial Intelligence
AMP =	Automatic Mixed Precision
API =	Application Programming Interface
AU =	Aarhus Univbarsity
C[x] =	Class [x]
CM =	Confusion Matrix
CNN =	Convolutional Neural Network
CPU =	Central processing Unit
DAGI =	Denmarks Administrative geographic subdivision
DNN =	Deep Neural Network
DOE =	Design of Experiment
DR =	Dilation Rate
DTU =	The Technical University of Denmark
FCN =	Fully Convolutional Network (Not Fully-Connected Network)
GDK =	GeoDanmark
GEE =	Google Earth Engine
Gen AI =	Generative Artificial Intelligence
GPU =	Graphics Processing Unit
GTB =	Gradient Tree Boost
IR =	Infra-Red
Kappa =	Cohen's kappa coefficient
LR =	Learning Rate
LS =	LandSat
LULC =	Land Use/Land Cover
mIoU =	Mean Intersection Over Unit Index
ML =	Machine Learning
NaN =	Not a Number
NIR =	Near Infra-Red

NN =	Neural Network
PA =	Producer's Accuracy
Po =	overall Accuray
RAM =	Random Access Memory
R-CNN =	Recurrent Convolutional neural Network
RF =	Random Forest
RGBA =	Red-Green-Blue-Alpha
RGBD =	Red-Green-Blue-Depth
ROI =	Region of Interest
RUC =	Roskilde University
SDSS =	Spatial Decision Support System
SGD =	Stochastic Gradient descent
SR =	Surface Reflectance
SWIR =	Short-Wave Infra-Red
TF =	TensorFlow
TIR =	Thermal Infra-Red
TOA =	Top of Atmosphere
TSLA =	Two-Stage Label Smoothing
UA =	User's Accuracy
UCph =	University of Copenhagen
VRAM =	Video Random Access Memory

