TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Joosep Koort 211674IAPM

MACHINE LEARNING PREDICTION ALGORITHMS FOR USE IN A TACTICAL SITUATIONAL AWARENESS APPLICATION

Master's Thesis

Supervisor: Sven Nõmm
Phd

Co-supervisor: Adrian Nicholas Venables
Phd

Tallinn 2023

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Joosep Koort 211674IAPM

MASINÕPPE ENNUSTUSALGORITMID KASUTAMISEKS TAKTIKALISES OLUKORRAS

Magistritöö

Juhendaja: Sven Nõmm
Phd

Kaasjuhendaja: Adrian Nicholas Venables
Phd

Tallinn 2023

Author’s Declaration of Originality

I hereby certify that I am the sole author of this thesis. All the used materials, references to the literature and the work of others have been referred to. This thesis has not been presented for examination anywhere else.

Author: Joosep Koort

08.05.2023

Abstract

This thesis presents the pilot studies of the development of an AI-based military situational awareness system, which requires synthetic data as a first step in predicting future force dispositions. More precisely, at the tactical level of an military deployment we assume that only partial information about the positioning of adversary forces is available, and the goal is to predict the future positions of units. To achieve this, a synthetic data generator for the prediction system had to be created. The reason for this is that the development of AI-based systems requires sufficient amounts of training data. In this work, the possibility of generating purely synthetic data sets on the basis of doctrines and field manuals is discussed. For this purpose, we have explored the application of a statistical approach to data generation and the application of generative adversarial networks.

To demonstrate the applicability of the generated dataset, deep learning-based models were trained to estimate positioning of the adversary units. These utilised convolutional neural networks and conditional generative adversarial networks. The convolutional neural networks demonstrated that by showing only 10.9% percent of the known positions to the machine, the whole area can be predicted. Less than that, and the positioning of the whole area is not shown. Conditional generative adversarial networks were able to predict the entire area under examination by showing only 10.9% percent of the known positions. Less than that, and the direction of the structure may contain errors.

The thesis is written in English and is 63 pages long, including 8 chapters, 38 figures and 3 tables.

Annotatsioon

Masinõppe ennustusalgoritmid kasutamiseks taktikalises olukorras

Käesoleva magistritöö eesmärgiks oli luua olukorrateadlikkuse süsteem, mis vajab esmalt sünteetilist andmestikku, ennustamaks maaväe üksuste asukohti. Eelduseks on, et osaline teave üksuste asukoha kohta on teada ja eesmärgiks on ennustada üksuste asukohta tulevikus. Saavutamaks seda, on esmalt vaja luua masinõppemudeli jaoks sünteetiline andmestiku generaator. Peamine põhjus selleks oli, kuna tehisintellekti süsteemid vajavad piisavalt treeningandmeid. Selles töös käsitletakse sünteetilise andmestiku loomist statistilise lähenemise vaatepunktist.

Et loodud andmestiku rakendatavust tõestada, treeniti käesoleva töö raames lisaks süvaõppevõrgustikud, ennustamaks peidetud üksuste asukohad. Nendeks võrgustikeks olid konvolutsioonilised närvivõrgud ja tingimislikud generatiivsed võistlevad võrgud. Konvolutsioonilised närvivõrgumudelid demonstreerisid, et näidates 10.9% teadaolevate üksuste asukohti masinõppemudelile, on võimalik ennustada ülejäänud üksuste asukoha paiknemine tervel alal. Kui näidata vähem asukohti, siis ei ole võimalik tervet ala ennustada. Tingimislikud generatiivsed võistlevad võrgud suutsid ennustada terve ala asukoha paiknemise samuti 10.9% puhul, aga vähema puhul võis olla tuvastatud struktuuri suunas eksimus.

Lõputöö on kirjutatud inglise keeles ning sisaldab teksti 63 leheküljel, 8 peatükki, 38 joonist, 3 tabelit.

List of abbreviations and terms

AIS	Automatic identification system. Tracking device used on ships.
ANN	Artificial neural network. Network based on human brain connections. In simpler format.
CGAN	Conditional generative adversarial network. Type of a GAN.
CNN	Convolutional neural network. One type of artificial neural networks.
CPU	Central processing unit, the main component of computers. Slower than GPU.
Data	List of variables such as speed, time, weather conditions, position, terrain and so on for one specific event.
Dataset	List of data.
DEM	Digital elevation model represents earth as a map
Epoch	Defines a number on how many times to update the weights in the training process of a neural network.
GAN	Framework proposed in 2014 for estimating generative models.
Hyperparameter	Term used in machine learning, defines the machine learning process, for example the size of the neural network, size of the batch, activation function and so on.
GPS	Global positioning system, allows to locate the location of something.
GPU	Graphics processing unit, a faster type of processor, which reduces the training time of neural networks.
Heatmap	One subset of explainable artificial intelligence. Allows to visually represent or highlight areas of interest on a picture.
Model	Machine learning model that contains mathematical steps. Could also be said algorithm.
Quantization	Reducing the information. In our case it is used in the context of color reduction process.
ReLU	Rectified Linear Unit is an activation function used in deep neural networks.

RMSE	Root mean square error is a commonly used measure to evaluate predictions.
Situational awareness system	System that takes into consideration multiple variables and returns a result after calculation based on these variables. In this context the situational awareness system may be a CNN or cGAN combined together with the preprocessing and postprocessing of the data.
SVM	Support vector machine are supervised learning models for classification.
Synthetic data	Computer generated data with the help of a user, rather than data obtained from the real world.
UAV	Unmanned aerial vehicle. Aircraft without passengers or a pilot.
YAGO	Yet another knowledge base is an open source knowledge graph containing data from wikipedia.

Table of contents

1 Introduction **11**

1.1 General background 11

1.2 Objectives of the thesis 11

2 Literature review **13**

2.1 Situational awareness 13

2.1.1 Situational awareness datasets 13

2.1.2 Situational awareness systems 13

2.2 Machine learning 16

2.2.1 Machine learning algorithms 16

2.2.2 Machine learning process 17

2.3 Synthetic data generation 17

2.3.1 SMOTE and ADASYN 17

2.3.2 MCAR, MAR and MNAR 18

2.4 Data augmentation 18

2.4.1 Data augmentation techniques 19

2.5 Chosen techniques for this thesis 19

3 Problem statement **21**

3.1 Research questions 21

3.2 Research problems 21

4 Background **22**

5 Methodology **24**

5.1 Distributions 24

5.1.1 Normal distribution 24

5.1.2 Dirichlet distribution 25

5.1.3 Copulas 26

5.2 Used programming libraries 27

5.2.1 OpenCV 27

5.2.2 Scipy 28

5.2.3 Scikit-learn 28

5.2.4 Pandas and Numpy 28

5.2.5 Plotly 28

5.2.6 Keras 29

5.3 General work 29
5.4 Pre- and postprocessing of the generated data 30
5.4.1 Picture analysis and -processing 31

6 Results **34**
6.1 Validation of the results using decision trees 34
6.2 Validation of the results using CNNs 36
6.3 Validation of the results using vision transformers 54
6.4 Validation of the results using GANs 56

7 Discussion **64**

8 Conclusion **65**

References **66**

Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis **71**

Appendix 2 – cGAN generator **72**

Appendix 3 – cGAN discriminator **73**

Appendix 4 – The formulas **74**

List of Figures

1 3D Digital elevation model. 14
2 2D Digital elevation model. 15

3 Attack positioning. 25
4 Defence positioning. 26
5 Points at an angle. 27

6 Part of a generated decision tree 35
7 Confusion matrix of a decision tree. 36
8 General idea on how to predict. 37
9 Original images and their masked counterparts. 37
10 CNN classifier architecture. 38
11 Prediction accuracy during classifier training 39
12 Architecture of chosen encoder-decoder network. 40
13 Training loss on encoder-decoder network. 41
14 Result of a prediction. 42
15 Prediction placed onto terrain. 43
16 Results of predictions on multiple images with 8 squares visible. 44
17 Results of predictions on multiple images with 7 squares visible. 45
18 Results of predictions on multiple images with 6 squares visible. 46
19 Results of predictions on multiple images with 5 squares visible. 47
20 Results of predictions on multiple images with 4 squares visible. 48
21 Results of predictions on multiple images with 3 squares visible. 49
22 Results of predictions on multiple images with 2 squares visible. 50
23 Results of predictions on multiple images with 1 square visible. 51
24 Using one prediction result as an input to the next prediction 52
25 Root mean square error score on CNN for each shown square. 53
26 Structural similarity index measure score on CNN for each shown square. 54
27 Masked image modeling autoencoder pretraining 55
28 Masked image modeling autoencoder pretraining with smaller patch sizes 55
29 Training loss on autoencoder during pretraining. 56
30 cGAN training process 57
31 cGAN model prediction results after last epoch and 8 squares visible . . . 58
32 cGAN model prediction results after last epoch and 1 square visible . . . 59
33 cGAN model prediction results after 5 epochs and 8 squares visible 60
34 Root mean square error score on cGAN for each shown square. 61

35 Structural similarity index measure score on CGAN for each shown square. 62
36 Visual Information Fidelity score on CGAN for each shown square. . . . 63

37 cGAN model generator architecture. 72
38 cGAN model discriminator architecture. 73

List of Tables

1 The generated data in tabular form 30

2 Decision tree metrics. 36

3 Data after postprocessing. 43

1. Introduction

In this section, the general background for the research is provided. This constitutes an example of why quality data is needed for Artificial intelligence applications and how it is generated, and an outline of the main objectives of the thesis.

1.1 General background

Situational awareness systems are systems that process large amounts of data, and because of this they require lots of precise data to make correct decisions. This is crucial in medical, security or military applications where decisions could cost lives. These so called 'digital twins', that simulate the situation in real life are as necessary. the whole world cannot be simulated yet. Machine learning algorithms can simulate part of it, but in general require large amounts of data to train the prediction model. This training data is usually obtained as a result of day-to-day processes, but there is an issue if there is only data available from a short period of time or the data is incomplete. In this case the data has to be generated artificially and added to the known data. The generated data has to be of sufficient quality, and in the context of our case, represent the structure of land forces formation.

As a result of this thesis, different methods of data generation are explored to create a final method. These take into consideration the final requirement, which will be a prediction of the position of land forces, where insufficient real world data is unavailable.

1.2 Objectives of the thesis

The main objective of this research was to investigate ways of data generation and how the missing data required to train a ML algorithm could be created. The aim was to create a structure of units representing the positional data applicable to real world situations in the form of synthetic data.

The second objective of this research was to examine a range of machine learning techniques. This generated data could be used to predict the positioning of land forces, with the help of machine learning models such as convolutional neural networks (CNN) or conditional generative adversarial networks (CGAN).

Multiple machine learning models were tested, because it was not known how well each would perform.

But before training of the model, the generated data had to be converted to a suitable image format that the machine learning models could use. Additionally, because the machine learning models had to predict an output, they required initial data of the known locations of a known unit. To simulate a situation like this, the model had to be shown a subset of the original data. One way to do this was to mask parts of the images which was tested to confirm if it was a valid technique.

After the successful training of the models, the findings were recorded, the quality of the model predictions noted, and recommendations for future work made.

2. Literature review

In this chapter, different aspects related to AI datasets and systems were researched. ML algorithms that are commonly part of the AI systems were then researched. Finally, our main goal, the synthetic data generation, was researched with ways to do data augmentation.

2.1 Situational awareness

2.1.1 Situational awareness datasets

Existing situational awareness datasets were examined to see if they could be used for this research to help the data generation process or increase the quality of the data. It was found that although there are lots of datasets available for training AI systems, most of them are for single use-cases with the exception of the GeoLife GPS trajectory dataset [53]. Vessels trajectory predition used its own kind of a dataset with its own parameters. For vehicle trajectory analysis, another dataset had to be used and so on. One paper highlighted a dataset for detecting humans and human actions such as if they are currently handshaking, hugging, reading, etc from an aerial view [7]. Although this could be used for military purposes using drones, this was not considered helpful for this research.

2.1.2 Situational awareness systems

Paper [17] discusses a system called the Person-Action-Locator (PAL), an UAV-based situational awareness system, which is useful for search and rescue or surveillance. Its main components are:

- Deep learning that detects people and recognizes their actions. The paper also mentioned that the accuracy of the model could be improved with a Digital elevation model (DEM). DEM is basically a color map that is 3 dimensional (3D) to represent terrain, which provides greater granularity of data [23]. A possible DEM for our case is shown on Figure 1, where the blue represents height values that are lower and brown represents values of higher height. Once the 3D map exists, it is easy to convert it to 2 dimensional (2D) pictures that would be more easily processable for computers as shown in Figure 2.
- The pixel2GPS converter that was developed by the authors of the same paper [17]. This estimated Global Positioning System (GPS) positions of individuals with image

processing. GPS coordinates were taken from the Unmanned Aerial Vehicle (UAV) along with the height of the UAV, it's roll, pitch and yaw, camera angle and image 2 dimensional point to be converted. OpenCV was used additionally to do the calibration of the camera.

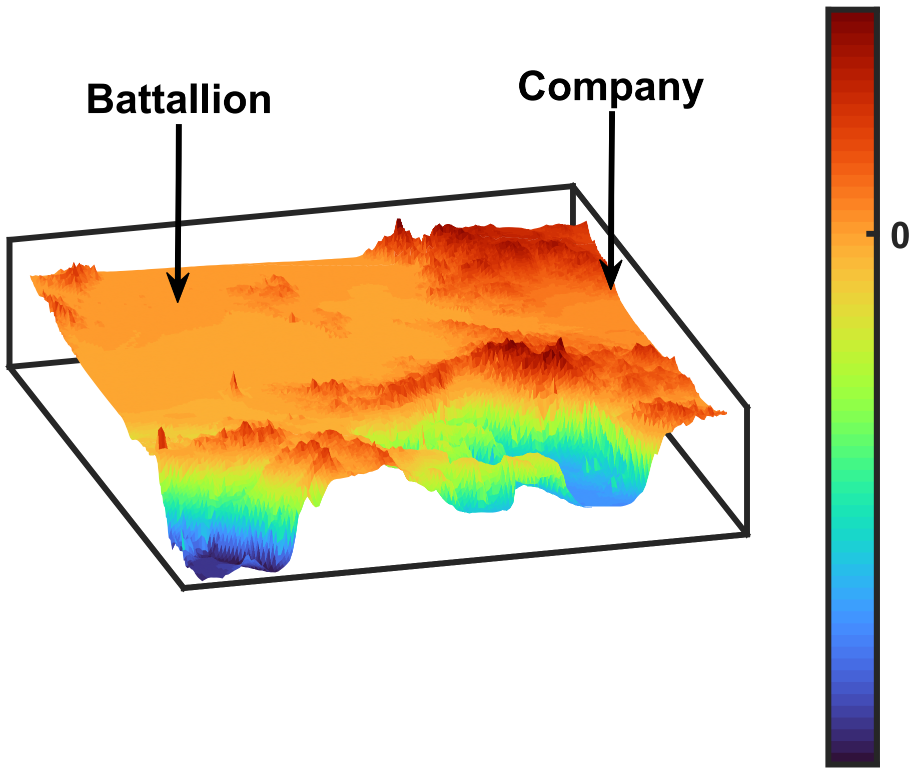

Figure 1. 3D Digital elevation model.

The same 3D dimensional elevation map can be represented in 2D as shown in Figure 2.

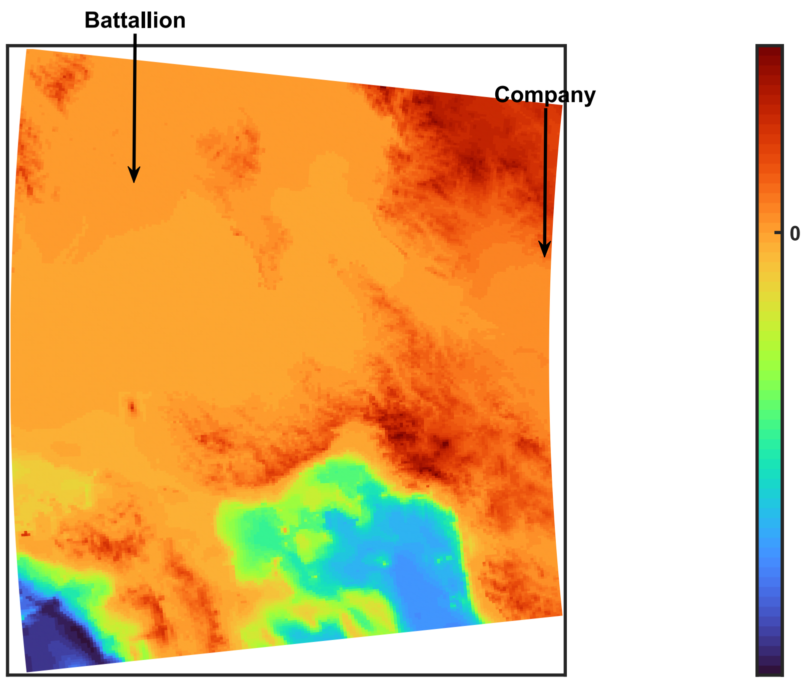

Figure 2. 2D Digital elevation model.

Another situational awareness system utilised images for road traffic analysis [54]. The authors of the paper described the usage of neural networks to make decisions by classifying the images on a low level with the use of convolutions. A simple neural network consisting of an input layer, convolution layer, ReLU (Rectified Linear Activation Unit), pooling layer and fully connected layer was used to make classification decisions. The system was trained using a Nvidia GTX 980Ti Graphical Processing Unit (GPU). It was noted, that the training process is much faster on GPUs than Central processing units (CPU), because of its parallelization abilities [2] and is thus preferred for this work.

Situational awareness has also been used in autonomous ships [44], where the authors of the paper wanted to achieve the total situational awareness of the vessel. In the paper, the predicted trajectories probabilities were calculated using an Automatic identification system (AIS) transponder. Upon further inspection, AIS transponders messages were noted to transmit lots of data including vessels position, speed and direction [25].

Furthermore, situational awareness has also been used to classify real-time social media text in the context of hurricanes. [51]. As a conclusion, the authors examined the capabilities of a CNN model in cross-event Twitter topic classification based on three geo-tagged datasets during hurricanes Sandy, Harvey and Irma. These datasets were reviewed manually and

Tweets were all manually allocated a category or a class. It was also noted by the authors of the paper that the CNN model had the capability of pre-training Twitter datasets from past events to classify an upcoming event for situational awareness. Each hurricane event contained 5000 to 15000 tweets. In the end there were 5 classifications created in the process of categorization. After showing all of the categories on a common plot, it was apparent that the Tweets rose during the time of hurricanes. They also tested the data and achieved an accuracy of 81%.

In the use case of trajectory generation, a map could be divided into a grid representation of squares which are fed into CNN as shown in [47]. It was noted that GANs are highly effective learning methods and in addition to image data, time-series data can be generated. It was noted that grid representation lost precision when the grid size was increased. An outdoor trajectory prediction dataset that could be used is GeoLife GPS trajectory dataset. This was collected by 182 users over a period of 3 years and it contained the coordinate positions and the height of users [53]. Additionally, trajectory prediction has been used previously with cars as an example and Gaussian mixture models (GMM) [50]. A coordinate system was used for the cars location and from that their trajectory could be predicted several seconds into the future. The dataset consisted of 69 real world trajectories, recorded at 3 different intersections with varying maneuvers. The parameters chosen were GPS data, yaw rate, history length of 25 meters and a prediction horizon of 2 seconds.

2.2 Machine learning

2.2.1 Machine learning algorithms

Multiple machine learning algorithms exist as highlighted in the literature review [26]. For classification some of the applications are decision trees, support vector machines (SVM) or K-Means, which is an unsupervised algorithm to cluster the data. For image processing Multilayer Perceptrons (MLPs), convolutional neural networks (CNNs), Generative Adversarial Networks (GANs) and transfer learning exist. GANs are data generation models that are able to create sets of data using a generator and a discriminator [2]. GAN-s can create realistic samples based on original data and are commonly used in image domains. An example was given that it was able to produce realistic looking bedrooms based on an image dataset of bedrooms [2]. Related to transfer learning, another paper discussed the topic of variational autoencoders (VAE) [27], where a VAE can be used for image resynthesizing. This means that a picture can be an input to the model and output will be another picture. An example was given in another paper, where an original image was transformed from a smiling face to a neutral looking face [49].

In summary, as a result of the literature review, the main techniques found that were suitable for our use case were MLPs, CNNs, GANs and transfer learning.

2.2.2 Machine learning process

To build a good deep learning model, the validation error must be constantly decreasing [39]. The general machine learning process is presented in paper [10], where the standard supervised machine learning consists of 3 steps:

1. Learning algorithm
2. A model is produced
3. Prediction is done on the model

This can further be combined with pre-processing, selection of the best trained model and evaluation. Depending on the requirements, an automatic deployment of the model may also be possible.

Explainability of the artificial intelligence (AI) comes with a trade-off. Lower accuracy often results in lower explainability or vice versa. It was noted from [4] and [19], that decision trees are on the higher end of the explainability. This is why the decision tree was chosen as the technique to represent our data, since the coordinates structure can be represented as decisions and image based. This could then be used on the final prediction image.

2.3 Synthetic data generation

In this section techniques of data generation are explored. Artificial data has been generated using GANs for machine learning tasks, although its limitation is to generate textual data [2] [37]. Some of the more commonly use cases for synthetic data generation is to avoid using the original data for privacy reasons [37]. Generation of the data may also be useful in imbalanced datasets in order to fill the gap of imbalance by generating the missing part [37]. Additionally it was noted that the testing method of choice for the generated synthetic data was a decision tree classifier.

2.3.1 SMOTE and ADASYN

There are 2 main techniques to generate data in the case of imbalanced data:

1. Synthetic Minority Oversampling Technique (SMOTE), which is the de facto standard for pre-processing of imbalanced data [38].
2. Adaptive Synthetic Sampling Approach (ADASYN), that was proposed after the success of SMOTE to learn and generate from imbalanced data [20].

These techniques seem promising, but they require the existence of data and were thus disregarded.

2.3.2 MCAR, MAR and MNAR

A review was done by the authors of the following paper [36], which reviews missing data generation approaches for uni- and multivariate data. Additionally the missing data mechanisms MCAR, MAR and MNAR are discussed. Evaluation can be conducted using root mean squared error.

Missing data mechanisms:

1. Missing completely at random (MCAR). This is a situation, where a percentage of data is missing with the probability of missing being the same for every case [11].
2. Missing at random (MAR) is a situation where data is missing only within groups defined by the observed data. MAR is also a broader class than MCAR [11].
3. Missing not at random (MNAR) is a situation, where the probability of being missing varies, meaning that over time the size of the data missing may increase [11].

The previously mentioned missing data mechanisms MCAR, MAR and MNAR are all very useful in the context of data regeneration. They could be used in future after generation of the synthetic data to correct the data in our case. However they are not researched further, because they require the initial data with missing elements.

During the course of this research it became apparent that the initial data had to be developed and one way to represent units locations is on a map of grids as paper [47] discusses. The map does not have to be divided to grids, as it is known that pictures of maps consist of pixels and each pixel can have a unit allocated to it.

2.4 Data augmentation

Data augmentation is an important topic to be researched because the data generator could be aided by increasing the dataset. This could either be in the process of creating the data, or once the generation of the data is achieved for use by the machine learning model. Data augmentation can artificially inflate the training dataset size by either warping or oversampling [39] data. Data augmentation is also useful to prevent overfitting, which means that the prediction in the end would be a perfect replicate of the original data [39]. This is something that should be avoided as the aim is for the hidden generated points to be original.

The following data augmentation techniques were investigated:

2.4.1 Data augmentation techniques

- Geometric transformations such as horizontal or vertical axis flipping. This is easiest to implement and has been proven useful on CIFAR-10 and ImageNet datasets [39].
- Color space augmentations such as isolating the color channels or increasing the brightness of the image [39].
- Rotation, where the image is rotated for number of degrees. This action is done on the whole image.
- Kernel filters, like Sharpen and blur, which move over an image with n x n size matrix. Using a blur filter will result in a blurrier image and using a sharpen filter will result in a sharper image. Sharpening and blurring are one of classical ways to apply kernel filters on images [39].
- Random erasing is used by randomly selecting a patch of an image and masking it either a black value of color or white [39].

The chosen technique in our case for the data augmentation is rotation of the image, because of its ease of implementation. Kernel filters could be useful to make the prediction more sharper. During the masking phase of the training, the random erasing of the image appears to be a good way to disregard the squares containing coordinate data either before training or in the prediction stage. Geometric transformation was avoided, because the structural representation must stay the same. Axis flipping would break the structure of units positioning.

2.5 Chosen techniques for this thesis

In conclusion, as a result of this literature review number of techniques were chosen for this. MLPs, CNNs, GANs and transfer learning were selected for their good usability with image data. Decision trees were also selected because of their readability and because of their high explainability. Finally, the techniques chosen for data augmentation were rotation of the image, kernel filters and random masking.

3. Problem statement

3.1 Research questions

This research examined the following research questions:

1. The first hypothesis was that it is possible to create a data generator which could be used in machine learning algorithms.
2. The second hypothesis was, that it is possible to classify spotted units using the machine learning algorithms once the generator is built.
3. The third hypothesis was that it is possible to hide parts of the generated data and predict the positioning of hidden units.

3.2 Research problems

To answer the research questions, the following research problems were identified:

1. What type of data augmentation should be used?
2. What's a good measurement of quality for images?
3. Which hyperparameter or hyperparameters affect the generator performance the most?

4. Background

In order to be able to make accurate predictions it is important to have as accurate initial data as possible to input into the machine learning algorithm. This thesis seeks to provide accurate data when there is no real-world data available. Within the context of this research, this is aided by a knowledge of military formations. Bigger structures contain smaller structures and those smaller structures in turn contain even smaller units. Knowing this we can construct a similar user-defined structure. The structure of the defence forces and size of the elements may differ by country, which is why the generator must take this into consideration. As data generation cannot happen without the initial data, the structure and the initial positioning of the structures must exist. From this it is possible to define elements or units in any possible way within the limitations of the user defined regions. As an example, the generated list of data could contain the following:

- Coordinate points
- Type of the element
- Type of the unit, in which the element resides
- Time
- Landscape on a given coordinate
- Direction

A typical defence forces structure from top to bottom comprises the following land forces units [6]:

- Army
- Corps
- Division
- Brigade or regiment
- Battallion
- Company
- Platoon
- Squad

Although this formation is representative, the size of the units may differ depending on the country and the nature of the operations being conducted. As the generation of the whole structure of the defence forces is a complex task, it was decided to only model the generation from Division to Company level. Although the generation of the lower levels such as individual units positioning was tested, the composition of the higher formations was considered more important to provide an overall impression of force movements. The positioning on the lower levels could also be seen as noise that fades into the background. If needed, the application could be developed further in the future to also include the positioning of the lower and higher level units but this was not included in this research.

During the course of this research it was necessary to understand NATO symbology. NATO land forces symbols have a standard format where colors and shapes of the element represent friendly, adversarial, neutral or unknown unit. For simplicity, this thesis investigated only friendly and adversarial colors.

5. Methodology

5.1 Distributions

In this section, different kinds of distributions will be presented with the reason of choice, usage and limitations.

5.1.1 Normal distribution

Normal distribution, also known as Gaussian distribution, is a distribution which is represented by a Gauss curve. This curve is specific to the normal distribution. It follows the principe that the data occuring in the center is more frequent. One of the most important properties of gaussian distribution is the 68-95-99.7 rule that shows how the data is spread out. 68% of the data falls into one standard deviation of the mean which means that if the data mean or so called center point is 1000 with a standard deviation of 500 then 68% of the data falls between the range of 500 and 1500 [28]. The remaining 32% is spread out on the left and right part of the distribution so that 16% is on the left and 16% is on the right. The 95 rule means that 95% of the data falls into 2 standard deviations of the mean. Simply said, using the same numbers as before we can find it by multiplying 2 with the value of the standard deviation and then subtracting this number from the mean to find out the left side of distribution. To find the right side of the distribution we repeat the same process but instead of subtracting we add [28]. Lastly, the 99.7 rule describes the range that data falls within the 3 standard deviations of the mean. This can be calculated similarly as the 95% range, but instead of 2 standard deviations, 3 standard deviations will be used. Although normal distribution is used in many fields, it cannot be applied to every real world problem. However, it can be applied to many and in this context it can be, because units distribution could be described with a normal distribution.

Normal distribution

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2 / 2\sigma^2} \tag{5.1}$$

where μ denotes mean and σ denotes standard deviation [3].

5.1.2 Dirichlet distribution

The usage of Dirichlet distribution, also called multivariate beta distribution, is preferred for some situations. This is because in case of an attack positioning, the elements are not uniformly distributed, but instead the density of the front elements is greater than of the back elements. Additionally, the usage of Dirichlet's distribution gives the possibility to specify the variety and shape of the elements. The attack structure is sharper in the front. The shape of the Dirichlet distribution is defined by 3 alpha values α_1, α_2 and α_3 and it is typically shown on a ternary plot. Also known as a triangular plot from which in our case we can easily convert the points to the normal 2 dimensional coordinate system and rotate, resize or move the points if needed. The first attack positioning shown on Figure 3 has alpha values of 1, 3 and 1. It has a sharp shape in the front. The second Figure 4 may describe a defence positioning of the elements with the alpha values of 1, 0.01 and 1. Both of the plots were generated using a Dirichlet distribution tool in [42].

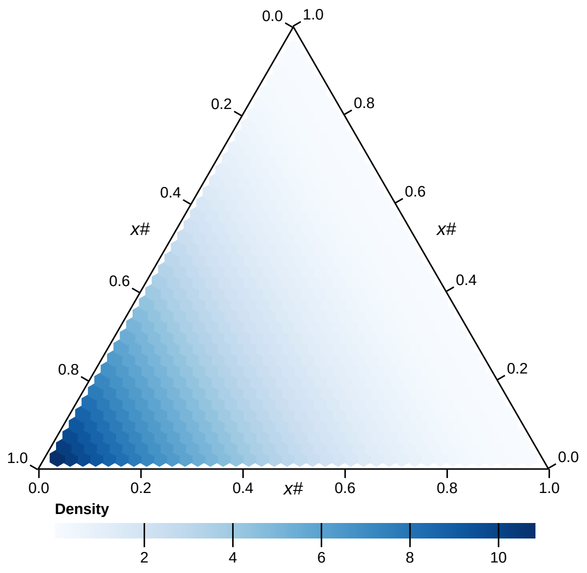

Figure 3. Attack positioning.

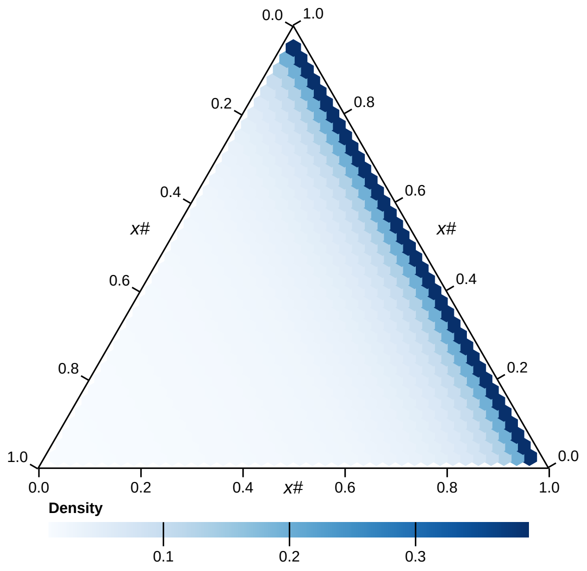

Figure 4. Defence positioning.

Probability density function of dirichlet distribution

$$f(\mathbf{x}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{K} x_i^{\alpha_i - 1} \tag{5.2}$$

A probability density fuction (PDF) is a function, that assigns a probability density to each possible value of given continuous random variables [49].

5.1.3 Copulas

Copulas are a family of uniform distributions that model correlated multivariate data [32]. Using bivariate copulas, a data of points could be generated that simulate a unit moving at an angle towards an objective [1] [16]. They could also be used to simulate a situation where 2 units collide, but this was not tested. Although a Gaussian copula played a big role in modelling financial crises, they could also be used to generate coordinates in the form of an unit attacking from a side [30]. Figure 5 shows sample of 500 generated points which are at an angle. The plot was generated using Matlab [23].

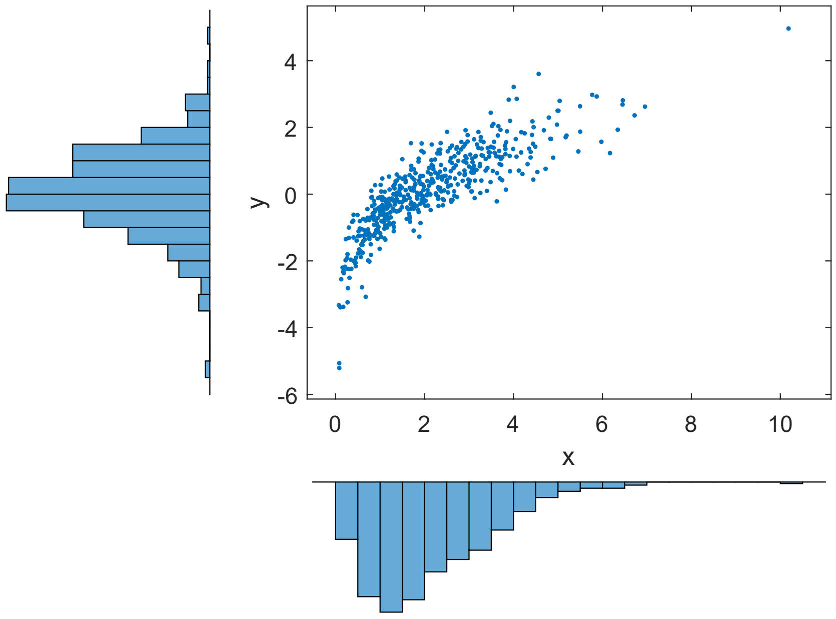

Figure 5. Points at an angle.

5.2 Used programming libraries

The solution and working application was implemented using the Python programming language and different Python libraries. The Python libraries employed and the reasons for their usage are as follows:

- OpenCV
- Scipy
- Scikit-image
- Pandas
- Plotly
- Numpy
- Keras

5.2.1 OpenCV

OpenCV library was used to perform image operations. These included image rotation, resizing and reading and writing of the images or for color conversions in the image masking stage. Additionally, it was used to reduce the number of colors on a picture in pre- and postprocessing.

5.2.2 Scipy

Scipy library was used widely to rotate points with multiple variables and to process, read and write images.

5.2.3 Scikit-learn

Scikit learn library was used to split the data into training and test datasets.

5.2.4 Pandas and Numpy

Pandas is a library mainly used in the field of data science. The main Pandas library competitor Numpy is used in science calculations and is faster to process. Pandas and Numpy libraries were used concurrently. Some actions, such as reading of the data was more easily processed on Pandas dataframes than in Numpy because of its tabular format. Numpy was used as it works with image data as the color image data is represented in 3 dimensional format consisting of 3 channels and the pixel values for each channel of image. It also works faster with science calculations.

To transform a coordinate point from one coordinate system to another using Numpy:

(x, y) = np.array([original_x_coordinate, original_y_coordinate]) / (np.array([original_-picture_width, original_picture_height]) / np.array([new_picture_width, new_picture_-height]))

This way it was possible to find out the x and y coordinates to put on a new smaller picture. Fewer pixels on a picture means less amount of processing time.

5.2.5 Plotly

This library was used to plot the generated points. It is worth mentioning that before plotting the points there was an important step to be made, which was that every generated point was scaled down to be on the right scale with the picture. This step was crucial to save the amount of computer processing time.

5.2.6 Keras

Keras is a machine learning library used to create the CNN models. Using a Keras sequential model API, the CNN models were created. Keras was also used in the cGAN model. Keras provides an easy way to save and load the model once the model has been trained.

5.3 General work

The programming language chosen for this research was Python. Python allowed for the quick compilation and running of the code. Initially an analysis was conducted to find out the structure of an army formation. Publicly available internet resources have a general structure of the army that was shown in the background chapter. We decided to use tabular data for the storage of data, more specifically Pandas dataframes. The initial formation was interpreted from the Commander's position. The Commander's location is the center point of the generation and the other units are a number of measurement units away from the Commander. Everything is therefore considered in relation to the Commander. This is the reason why the method call parameters are x and y coordinate location of the Commander. Then there are area dimension parameters defining the width and length of the generated area. First the Commander's position is added to the dataframe along with the type of the unit (Commander). After this, because everything is related to the Commander, the other generated points were chosen randomly for each generated unit, so long as they fit into the unit's region. Each unit is created separately consisting of its own limitations such as the approximate position. Most of the points are not rotated with the exception of the front line generated points, which are rotated to match the direction of the movement. This was because it resides in the front line of the whole structure and therefore is moved in front of other units. Dirichlet distributions have concentration parameters called alpha values that define how distributed the data is - the bigger the value, the more evenly spread out is the data. During the generation of the coordinate data, each category or unit is also assigned a hardcoded value such as the Commander or tank battalion. In this stage the number of rows of data for each unit is also defined - this is a mandatory step before generation of the points. After repetition of this step for each unit, the data is ready in dataframe or tabular form.

At this stage the data comprises of 3 columns called x, y and type. The x and y are float values and type is an integer value representing the type of the unit in a numeric value. In subsequent stages the number values are given a text value. An example of such generated data in current stage can be seen in Table 1.

Index	Location_x	Location_y	Type
1	9197.45752	9000.00000	0
2	9181.37090	9627.82371	0
3	9184.63288	11108.74569	1
4	9100.00000	10571.31634	1
5	9173.31216	9405.05562	0
N	...	...	0-10

Table 1. The generated data in tabular form

To convert the tabular data to an image format for the machine learning models to work, the generator is run in a loop for user specified times which is usually 1 and the received tables are concatenated if needed. The numeric unit values are then manually linked to colors. The reason to manually specify the colors in this stage is to make sure that on each generation the units are represented by the same colors. After this the images were saved in a folder with png format from where they could later be taken for further processing. Matplotlib library was used to link the units to the colors and save the images.

The colors used to represent 11 classes on an image in this thesis are the following: red, orange, blue, purple, green, violet, cyan, yellow, magenta, lime, grey.

When the generated points were shown on a plot, it was apparent that the points represented were not linearly separable, meaning they were not able to be separated by a single line on a 2 dimensional graph [2].

5.4 Pre- and postprocessing of the generated data

Pre- or postprocessing of data may be useful in the following way. If we want to attach additional data either to known locations of units or in the phase of data generation to the predicted data, we could compare the known data or generated data against a map. For example if an UAV spots 1 or multiple units at 1 or multiple locations, it is possible to use this coordinate information and compare it against 1 or multiple color maps. The concept was to have a map similar to the DEM shown before in Figure 1, but instead represented in 2D. Color maps such as a contour charts has been successfully used before to represent sea surface temperatures [3]. The conclusions may be that the unit was spotted on a mountain, nearby water or on a road. This can also be used to verify the acquired data. Automobile vehicle could be spotted on an x and y coordinate, but without the additional information given it is not possible to know it’s condition. If it’s in the water, then the automobile vehicle is most likely inoperative or the data is erroneous. Alternatively, if a boat is spotted moving on terrain and far from water then most likely false information was given. But if the boat was spotted nearby water, maybe there was misinformation and the data could still be valid. In the generation phase, if point is created on water but it's unit type, for example is a car, which is not allowed to be there, the point could be moved to another location. Alternatively, if the generated point falls onto a road, it could be a motor vehicle and this useful additional information could be attached to the point. If we know for certain that the generated point cannot be in the generated location then such areas would be excluded.

One way to do this data validation is to utilise multiple map pictures. These separate pictures can be used to verify different types of information. For example, one picture could be the terrain data showing the height from sea level. This could be used to set the height level on found point or so called Z dimension. Another stored map could be of terrain data with colors such as the areas, where one could move slower such as a swamp. If the point falls into a swamp, it is most likely slower moving than the point falling onto a road or a forest. Multiple custom maps could be made that have different color representations and the generator could be used to set each new parameter of data while comparing the position against the map. An additional possibility would be to have a map of roads and use this information to predict the possible future locations of units using the A* pathfinding algorithm. Because of A* algorithm, the graph structure can also be described as a grid and vice versa and thus is preferred to be used, because units can be placed on a grid structured map [3]. Combined with terrain data, one solution would be to set the weights on the roads with the road structure represented as a graph or grid. A similar representation has been discussed in [34], where in addition to a general approach of A* algorithm, unit safety was taken into consideration using Multi-Objective Ant Colony Optimization (MOACO). This was not investigated further.

5.4.1 Picture analysis and -processing

The chosen technique for picture processing was color quantization. First, an open source map was downloaded and an external library OpenCV functionality was used to reduce the colors in a picture to a defined value. The algorithm used in such a process is called k-means algorithm. The process itself to reduce colors is called color quantization and in simpler terms similar colors in a picture are grouped together into a common color. The parameter K defines the number of colors that the final picture will have. This OpenCV function may be used in numerous places, but in this research it was used when reducing the colors in an image that was predicted. That is because our prediction contains colors and the computer was required to understand those colors and set a class value of a specific unit.

K-means clustering algorithm

A K-means clustering algorithm uses an error function to determine when the clusters have converged. This error function is called the sum of squared errors. Clusters in this research are the groups of similar colored pixels and the convergence means that no further updates are required to update the groups of pixels. Centroids of the clusters refer to the center points of the clusters. The amount of desired K clusters means that there is also the same amount of cluster centroids.

The sum of squared errors scoring function is defined as: $SSE(C) = \sum\limits_{i=1}^{k} \sum\limits_{x_j \in C_i} ||x_j - \mu_i||^2$

The k-means algorithm works as follows [3] [40]: Given a dataset $X = x_1, x_2, ..., x_n$, and a number of desired clusters k:

- Randomly initialize k centroids $\mu_1, \mu_2, ..., \mu_k$
- Repeat until convergence:
  - Assign each data point x_j to the closest centroid:
  - $c_i = \text{argmin}\, j|x_j - \mu_j|^2$
  - Update each centroid μ_j to the mean of the data points assigned to it:
  - $\mu_j = \frac{1}{|C_j|} \sum x_i \in C_j x_j$, where $C_j = x_j : c_i = j$
- Return the final centroids $\mu_1, \mu_2, ..., \mu_k$

The concept was to represent spatial data using images to extract height data [3]. The chosen number of colors for a map reduced to was 7. This means that darker colors that were initially bright red or dark red, were dispersed to a common red. This enabled the identification of areas such as mountains. Light blue and dark blue colors faded into a common blue representing water. Another preprocessing step was to convert the asphalt road color to white, where upper and lower bound values had to be defined in order to make computer understand what roads were. Finally, the OpenCV algorithm replaced these pixel values with white. This step allowed in later phase to classify points falling on the road with ease. The preprocessed image was not shown to the user, but it was stored in the memory. This upper and lower bound can be used after prediction to convert each main color back to a class. The setting of the lower and upper bound color values was also used to remove the white background color in a predicted image.

In future, knowledge bases such as YAGO (Yet Another Great Ontology) dataset [41] could be used. Yago allows to acquire a region in the world by specifying a coordinate window. By getting this region, the height values of points on the map could be obtained. In addition, it is possible to get points of interest such as cities and the variables related.

This functionality doesn’t fall in the scope of this thesis, but it could be an area for future investigation.

6. Results

In order to validate the results of the generated data, it was necessary to mask parts of the data. One solution was to input the previously generated images described in the solution chapter, resize each image to the specified size and divide the image to an equal number of squares. Then, using a parameter to define how many numbers of squares to leave in the visible area, meaning they will not be masked, the area is masked. The unmasked squares are picked only if they contain a pixel value containing anything besides white. The remainder is masked. This process is repeated for each image and then depending on the method of validation, the pictures are merged into a single image, as the cGAN (conditional generative adversarial network) model provided in the example requires [43]. To generalize the generated data for training, pictures were rotated 90, 180 and 270 degrees before saving the merged image. This process is simpler to do than to use some kind of new network such as Deep Rotation Equivariant network (DREN). However, the authors of that network suggest that it can improve the performance of current state-of-the-art techniques [29]. Because of the universality of our solution, the generated pictures may be used for multiple networks such as CNN encoder-decoder, autoencoder and conditional GAN. They all take in an image and output an image. The only exceptions are the decision tree and the first CNN model.

6.1 Validation of the results using decision trees

The first validation was done using the decision tree to represent the generated data as a decision tree. It was chosen to understand the generated data positioning better [37]. Although the following is the output of running the decision tree on the author's generated data once, multiple of decision trees could be used as inputs to a decision forest algorithm [22]. Decision trees can be read from top to bottom and contain the rules of how the data exists. The generated decision tree may differ each time it is generated. This randomness can be turned off by setting the random state variable to a constant variable [35]. Figure 6 shows one generated decision tree with a depth of 2 meaning that lower levels are not shown.

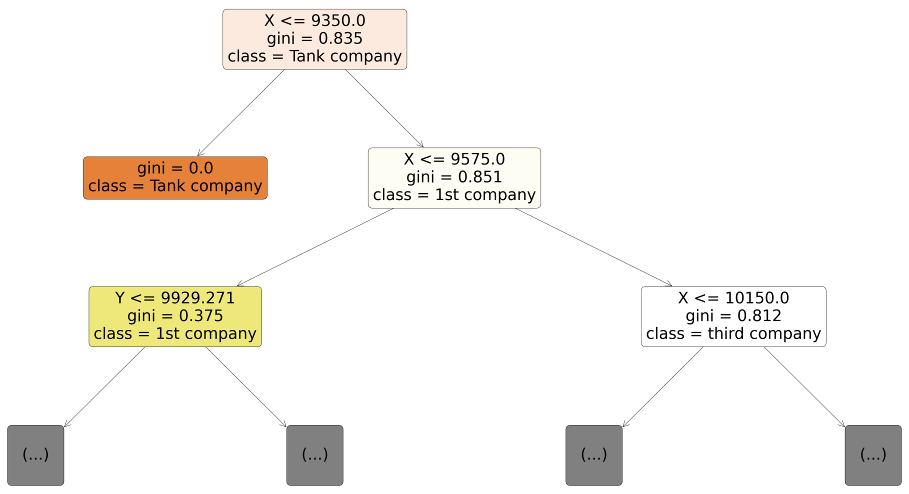

Figure 6. Part of a generated decision tree

Each node shows the Gini index value, the classified value, and the coordinate value for the required decision. The beauty of decision trees is their simplicity and the ability to see rules. Although these rules may differ on each generation, they still fit the conditions of the original data on each run [31]. The measure used by decision tree to validate the split of the branches is called Gini index and it is used as the splitting criterion [10]. The following shows the formula of Gini index.

Gini index

$$G = 1 - \sum_{i=1}^{n} p_i^2 \tag{6.1}$$

A confusion matrix of the generated decision tree can be seen in Figure 7.

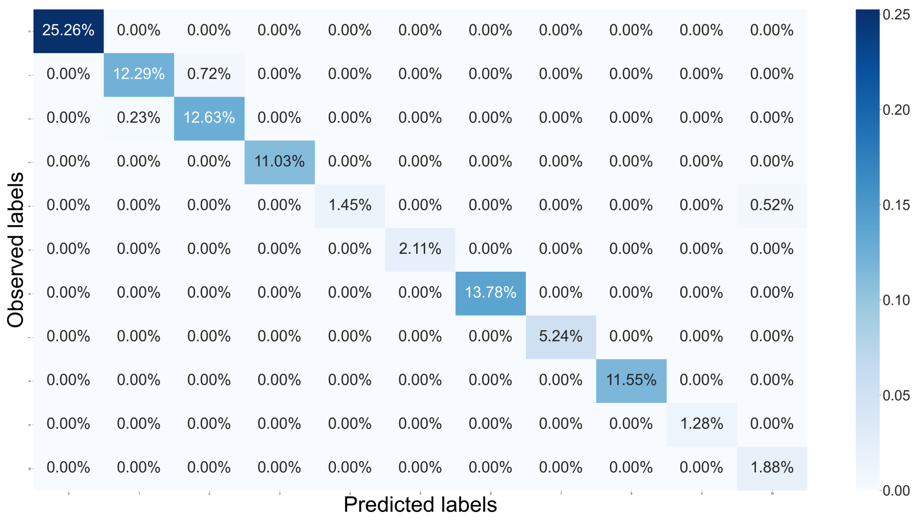

Figure 7. Confusion matrix of a decision tree.

Additionally, the precision, recall and F1-score values for each class can be seen in Table 2 and the accuracy of the model is 0.99.

Class	Precision	Recall	F1-Score
0	1.00	1.00	1.00
1	0.98	0.94	0.96
2	0.95	0.98	0.96
3	1.00	1.00	1.00
4	1.00	0.74	0.85
5	1.00	1.00	1.00
6	1.00	1.00	1.00
7	1.00	1.00	1.00
8	1.00	1.00	1.00
9	1.00	1.00	1.00
10	0.78	1.00	0.88

Table 2. Decision tree metrics.

6.2 Validation of the results using CNNs

The idea of validating the results is to give the computer data of known points and return the whole area prediction (Figure 8). The input areas should contain some data, so the network will not train with inaccuracy. The method that was followed was to mask parts of picture containing data, squares not containing data were skipped. Even if a square visible to the computer contains only 1 pixel of data that is of color besides white, the square is included.

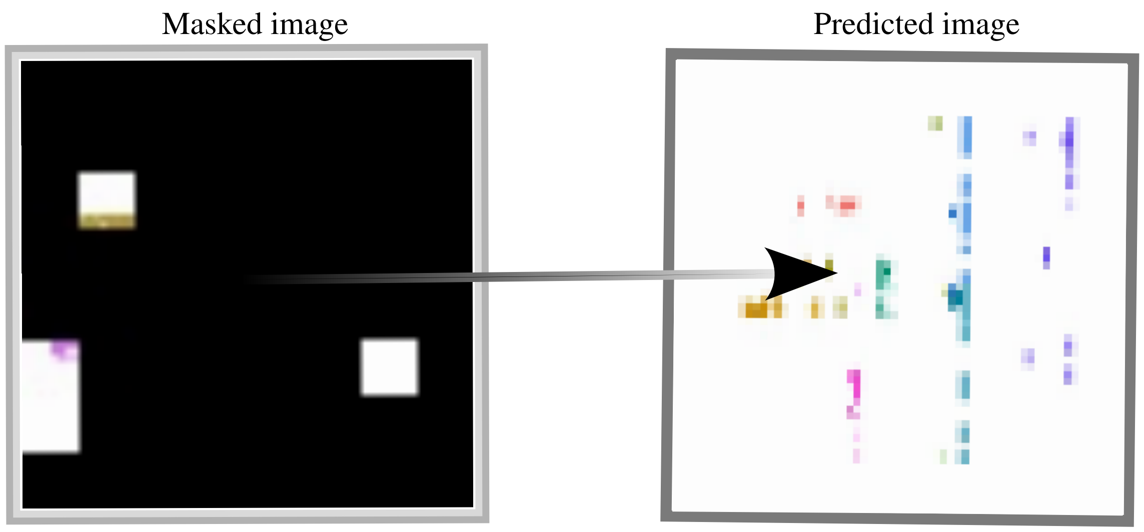

Figure 8. General idea on how to predict.

One way to solve this problem is to create pictures out of the generated points, and for each picture create a masked image as shown on figure 9.

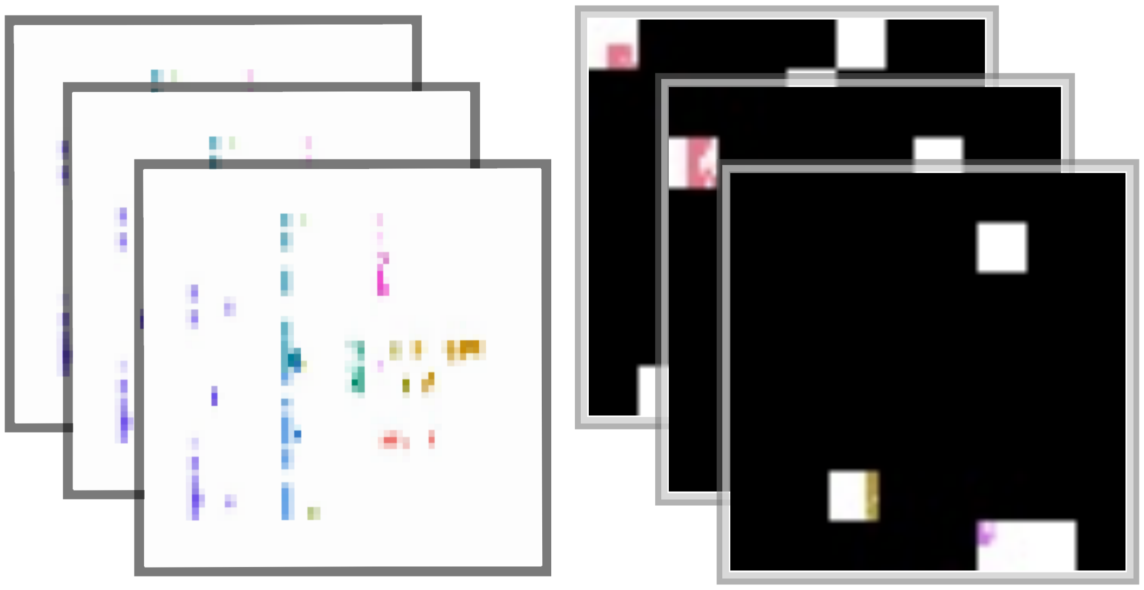

Figure 9. Original images and their masked counterparts.

The decision between (Multi layer perceptron) and a CNN (convolutional neural network) was made. CNN was chosen over MLP, because CNN understands spatial relation between pixels better than MLP. This is because CNN, as the name suggests - uses filters to detect features [52]. The kernel in our CNN model is of 3 by 3 size. Using a CNN results in higher accuracy of classification results than MLP [9] [13]. Since kernels are very useful, one type of such kernel was also used to manually emphasize features in a prediction in the context of this thesis. It is recommended to use a 2D convolutional neural network when working with image-like data [15]. 2 CNN models were created. The first classifies a coordinate tuple and the second reconstructs the image from a masked image.

The architecture of the first CNN model

The purpose of the first created CNN model was to classify points. The general steps of a classification algorithm were followed. These were the training phase and testing phase [3]. Data that goes into the model is split using train_test_split function of the Python scikit-learn package. The split ratio was chosen to be 50%. This ensured that there was no direct data contamination while training the model. Furthermore, to make sure no indirect data contamination was present, there was no normalization conducted on the whole dataset. The architecture of this classifier consisted of 4 layers. The following is the architecture of the first created and used CNN model. It was expected from this CNN model to classify an input point, which in our case was a coordinate tuple. Since it had the simplest structure, it was a good starting point experimenting with CNN-s.

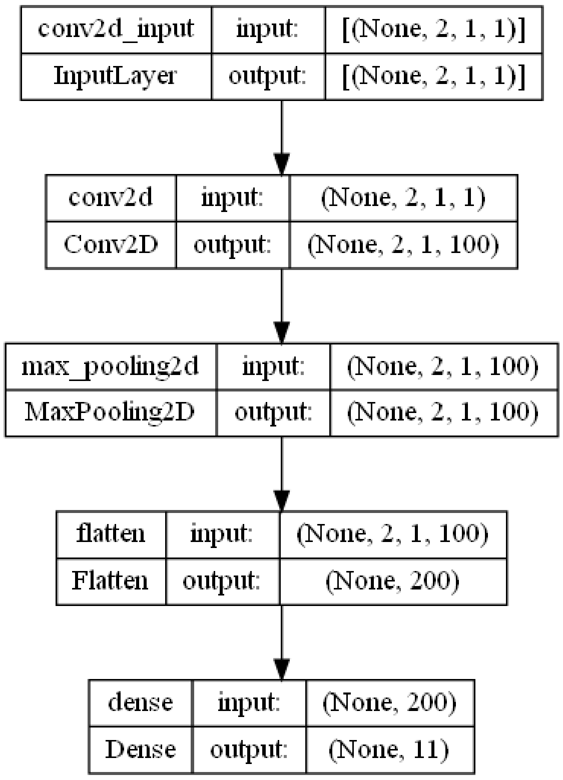

Figure 10. CNN classifier architecture.

The first layer is the convolution layer, that does the convolution operation [2]. Next is the max-pooling layer, which extracts the maximum value of the feature map. Then the output of the previous layer is flattened by making it a vector. Lastly, we classify the result using a dense layer using softmax activation function, which computes the probabilities of the results [2]. Because there were 11 possible outputs the classifier could decide between, the number of dense outputs was also a numeric value of 11. See Figure 10. Figure 11 shows the accuracy of training on said model.

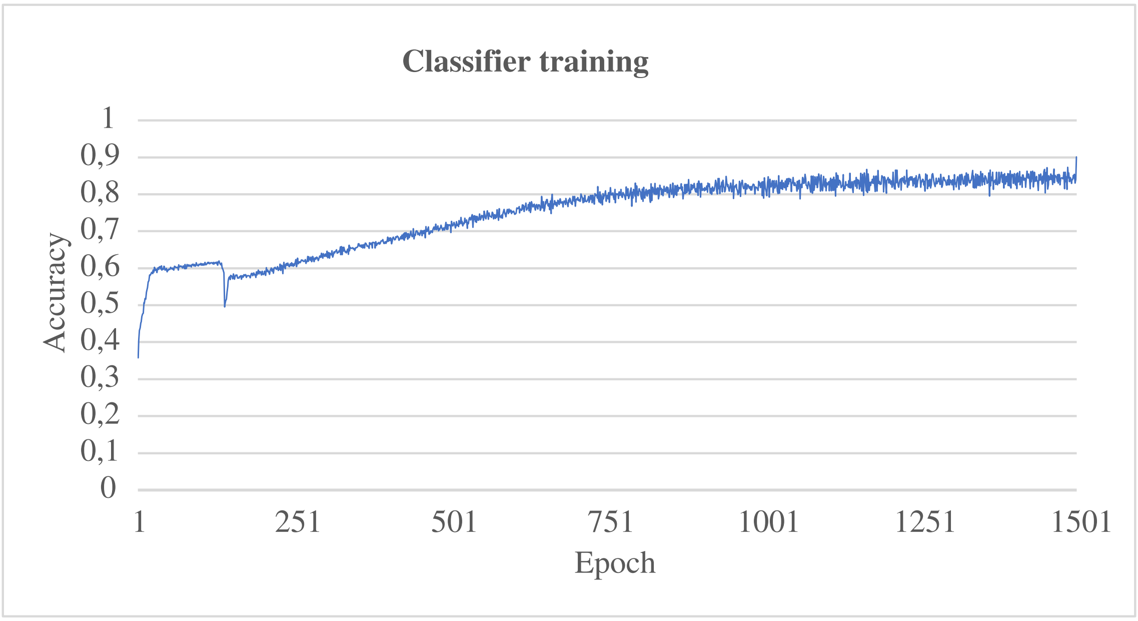

Figure 11. Prediction accuracy during classifier training

From this plot we can notice that during the first part of the training there was a steep increase and quite a high drop. However, the training rate then stabilized at around 200 epochs with the prediction accuracy increasing over time. Hyperparameter tuning would be a key here to make this training smoother. This classifier took an hour to train on an Intel I7-8850H central processing unit without graphics card. The accuracy of the model was 0.91, precision was 0.86, recall was 0.91 and F1 score was 0.88.

The architecture of the second CNN model

The purpose of the second CNN model was to predict unit locations based on some known units. This CNN model was with encoder-decoder architecture. Using the modified architecture shown in [12] with batch normalization we built a CNN encoder-decoder network, which allowed to take in N number of images with size of 64x64 along with masked images. Batch normalization was chosen, because it is able to reduce the effect of slower convergence during the training [2]. These masked images represented the locations of known units. As the trainable model was now more complex and images are used, a NVIDIA GeForce RTX 3090 GPU was used to train the model which greatly shortened the training time. The training process took 150 epochs. The images for the CNN model were created using the generator that was previously built and each generated image was masked. In the end using the Numpy python library savez function, both the masked and unmasked images were saved into a single file [14]. Before training the model using the given architecture, the dataset was loaded using numpy's load function. The normalization was conducted separately on the training part of the input images with missing parts and ground truth images that presented the full picture. Then the data was again split, using the train_test_split function of Python scikit-learn package with 50% split ratio and finally the train and test sets were normalized separately. After that the model was trained using those sets of pictures containing the training data and validated using the test data. The architecture of the built network can be seen on Figure 12. Figure 13 shows the loss decrease during the training of the network.

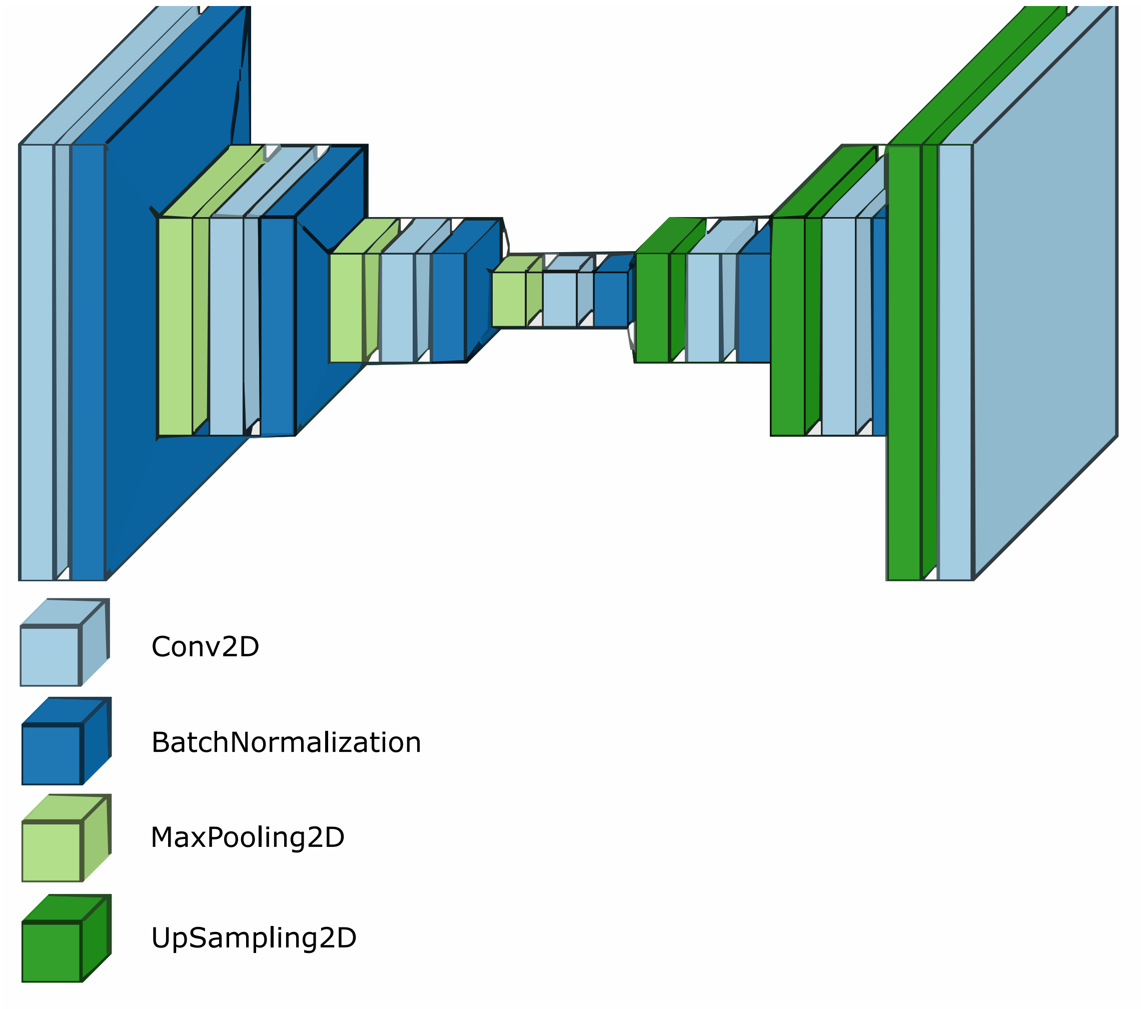

Figure 12. Architecture of chosen encoder-decoder network.

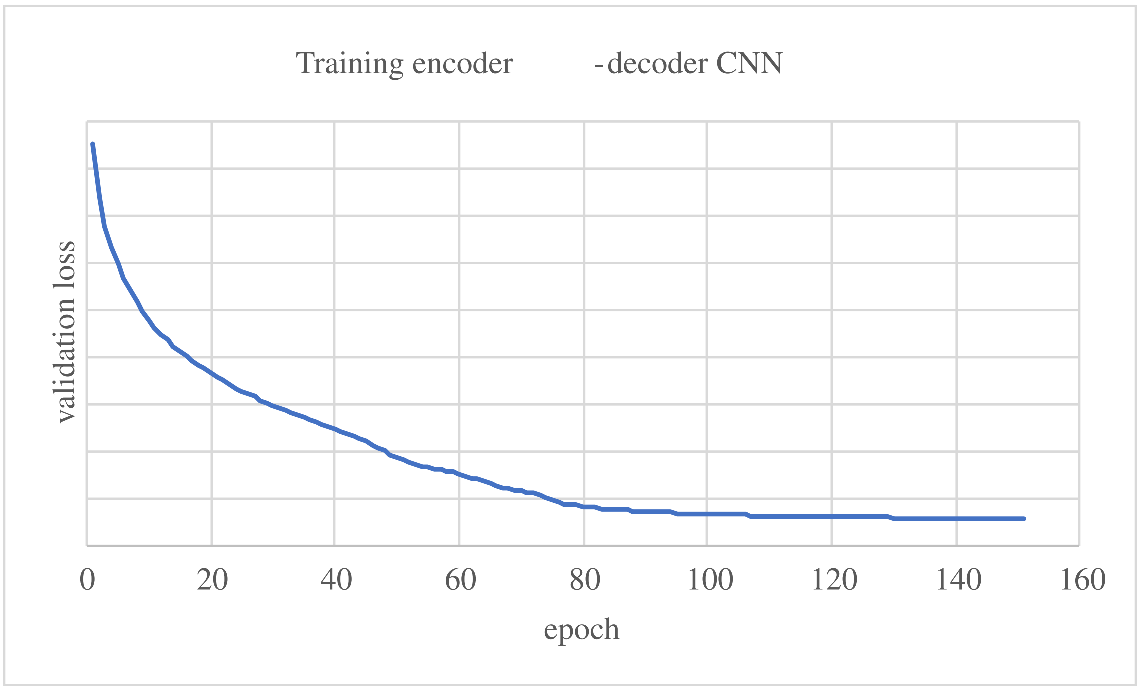

Figure 13. Training loss on encoder-decoder network.

After the training of the model was completed, it was possible to predict the locations of the units by showing the computer a part of a picture containing known units. An example of a prediction can be seen on Figure 14.

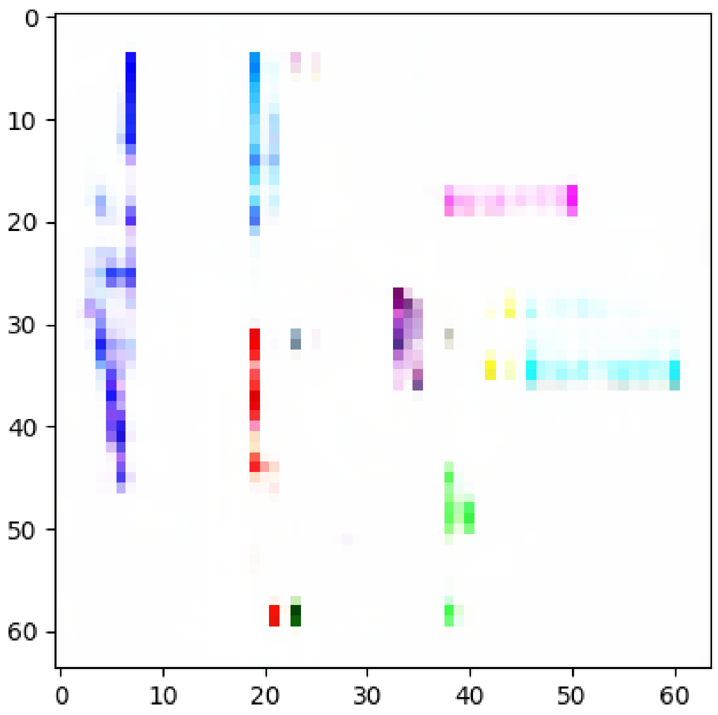

Figure 14. Result of a prediction.

The same prediction picture can be then post-processed, by optionally reducing the number of colors and comparing the terrain height value against the predicted unit's location, height values assigned to the units. Additionally, by knowing the terrain type on a given coordinate, it is possible to attach a speed variable to the unit as shown on Table 3. In the context of this thesis only height value was set based on the map color. An example of prediction placed onto a contour map can be seen on Figure 15. The map was obtained from [46].

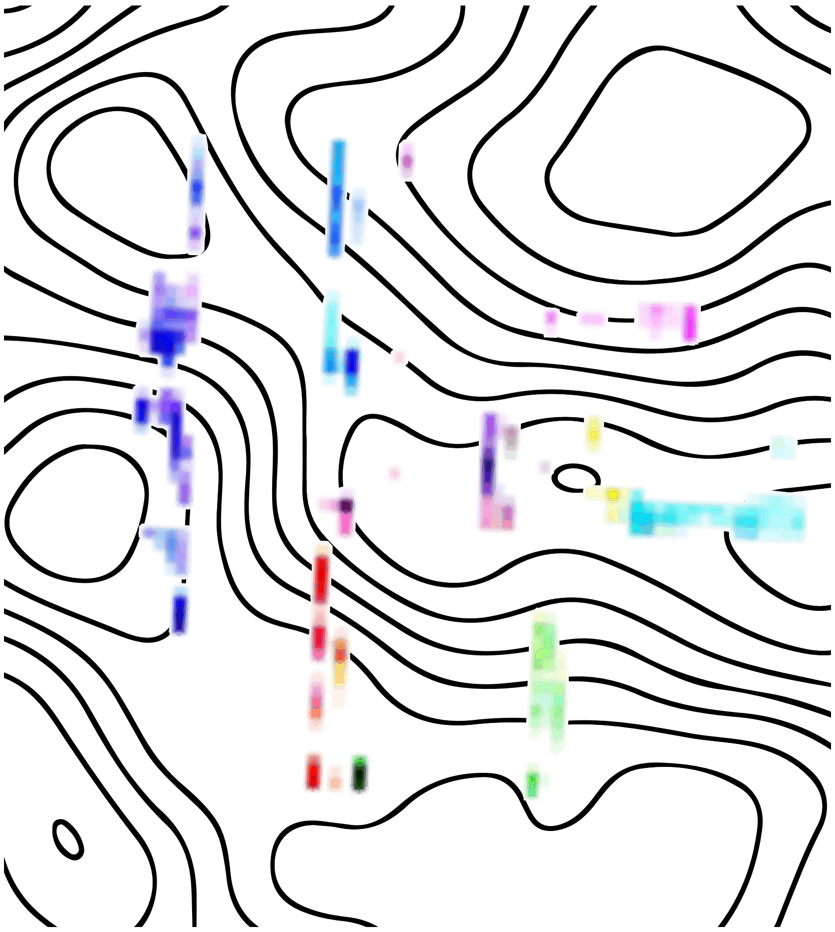
Figure 15. Prediction placed onto terrain.

Index	Location_x	Location_y	Type	Height (m)	Speed (km/h)
1	9197.45752	9000.00000	0	2500	45
2	9181.37090	9627.82371	0	2200	45
3	9184.63288	11108.74569	1	2900	90
4	9100.00000	10571.31634	1	1900	80
5	9173.31216	9405.05562	0	2000	45
N	...	...	0-10	-10000 - +10000	0-200

Table 3. Data after postprocessing.

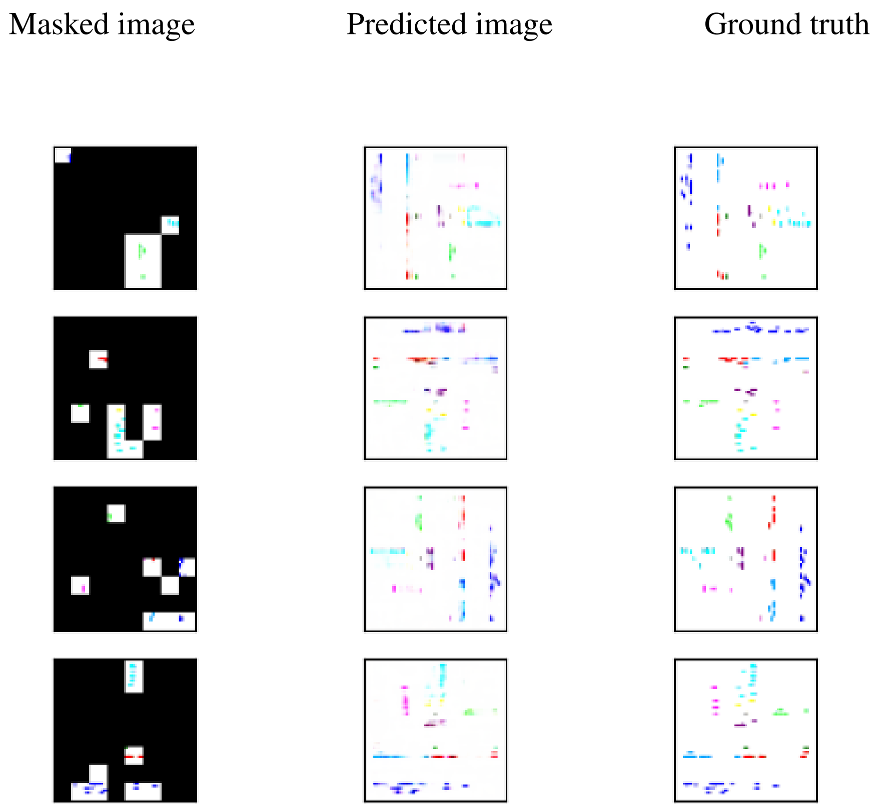

Figure 16. Results of predictions on multiple images with 8 squares visible.

The results on Figure 16 illustrate that not only did the model correctly predict the structure, but also the facing direction, noting that the training data was augmented with rotations as suggested in the literature review. It was also shown that only 8 squares of the 64 available were presented to the computer. This means that the CNN model was able to correctly predict the structure and direction with only 12.5% visible data. The time to load the trained model and predict the images varied from 3.7 seconds to 4.4 seconds. This time was measured using the Python time module. Although a human operator could understand the facing direction naturally after knowing the context of course, the computers required training. This is why it was possible to train the computer to also provide the direction as text once another model was trained. To appreciate the results that the CNN model gave, it was crucial to normalize the picture before using it as an input to the model.

To find out how well the model performed with even less available data, a series of tests were carried out showing 1 square less of available data to the computer each time.

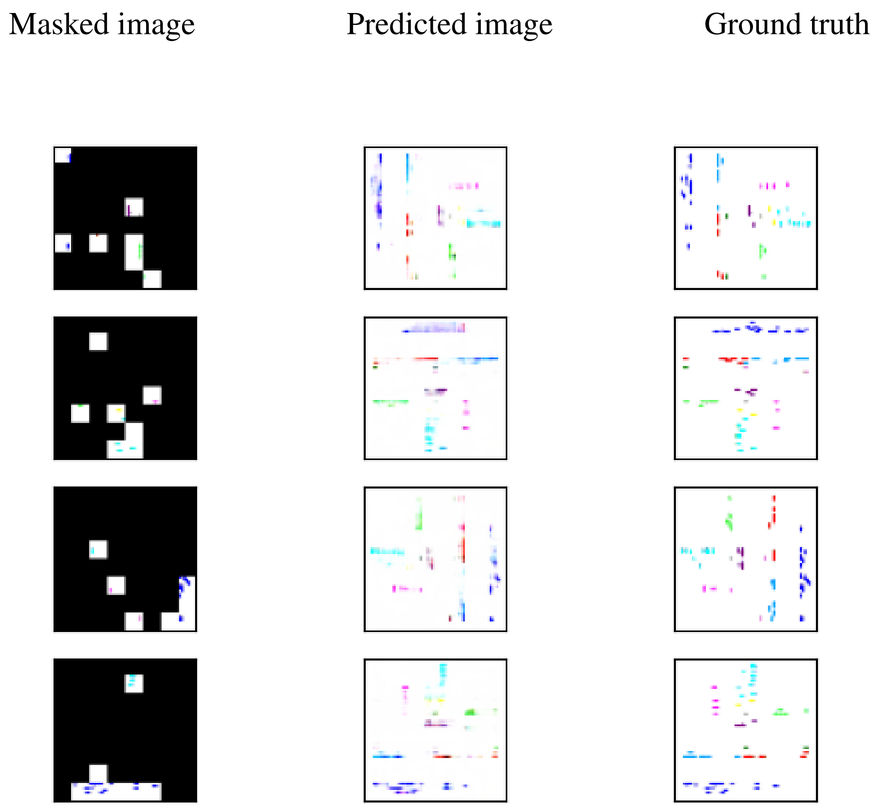

Figure 17. Results of predictions on multiple images with 7 squares visible.

In Figure 17 there didn’t seem to be any big differences compared to 8 squares shown.

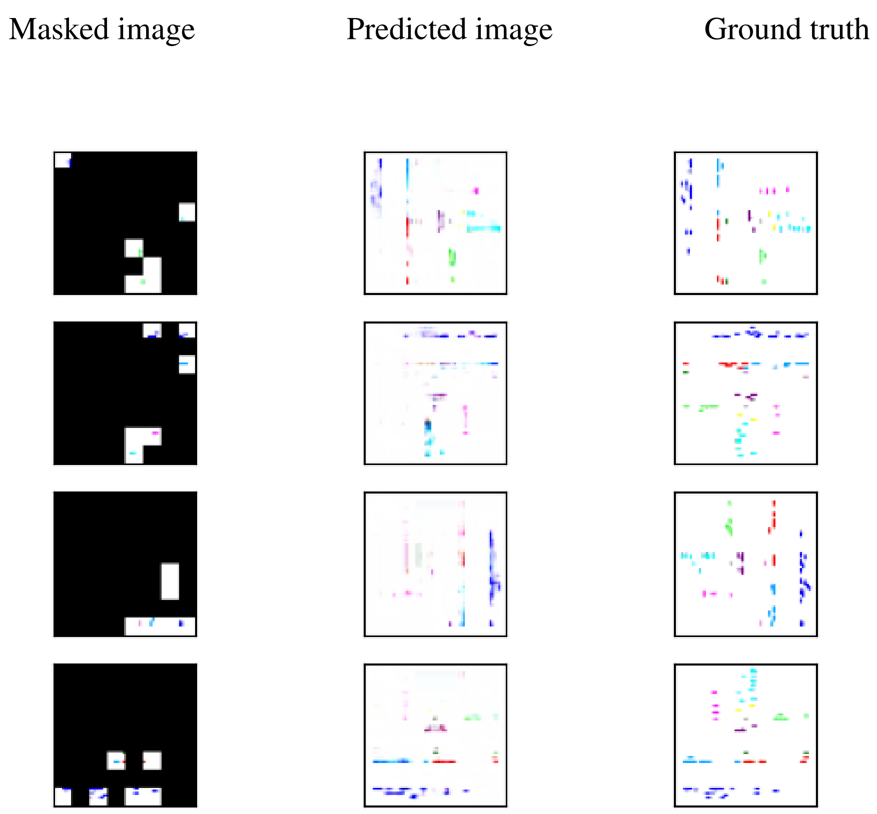

Figure 18. Results of predictions on multiple images with 6 squares visible.

In Figure 18, only 6 squares were shown to the computer with the resulted prediction having some of the space in the opposite side of the shown squares being blurred out.

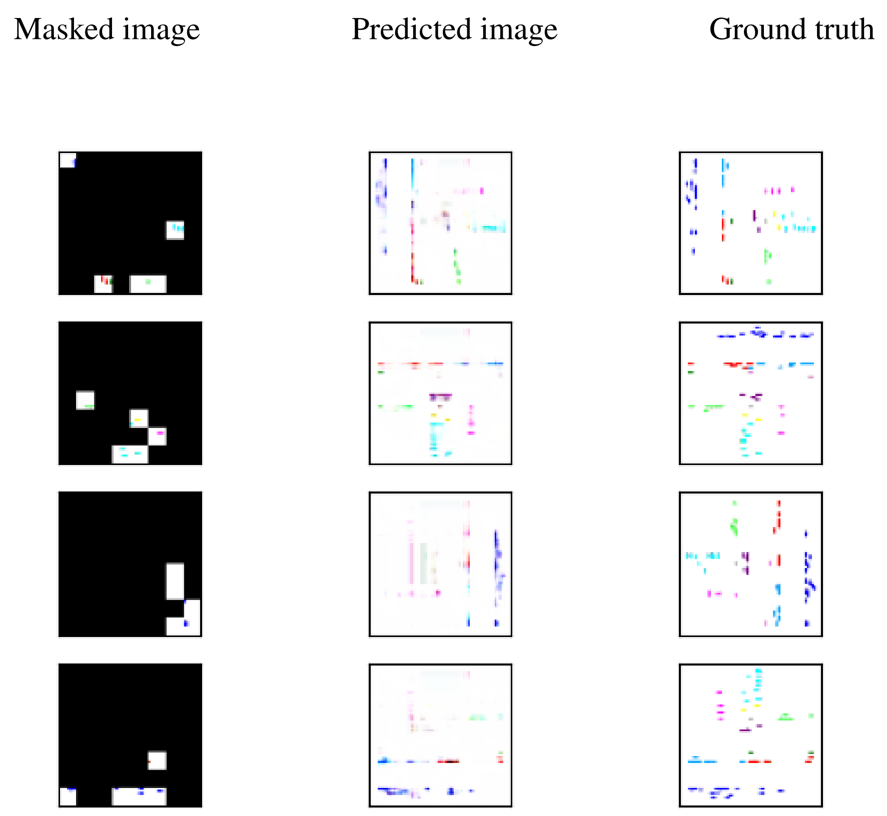

Figure 19. Results of predictions on multiple images with 5 squares visible.

Similar to the previous prediction, showing 5 squares to the computer results in areas opposite to the shown areas being blurred as shown in Figure 19.

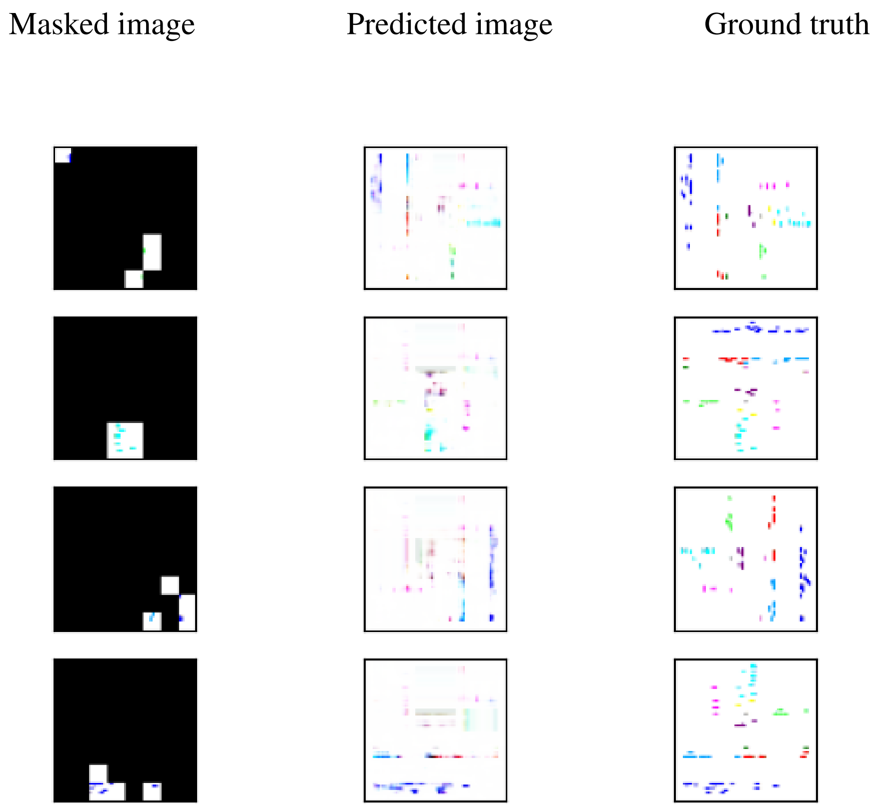

Figure 20. Results of predictions on multiple images with 4 squares visible.

In Figure 20, with 4 squares shown to the computer, the blurred areas were more common.

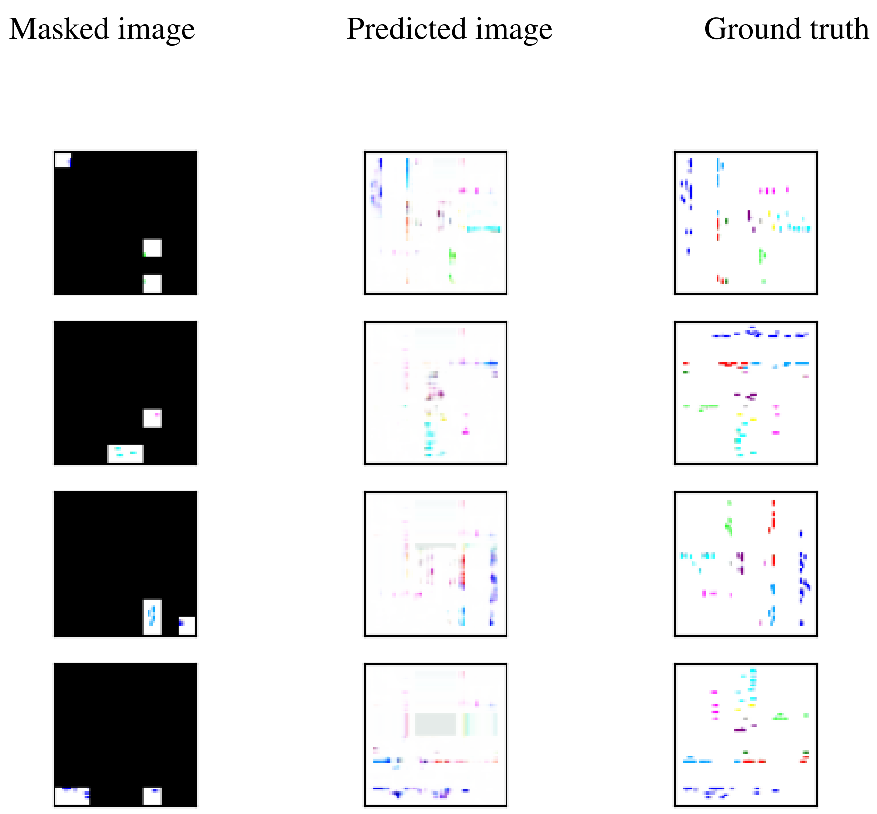

Figure 21. Results of predictions on multiple images with 3 squares visible.

In Figure 21, the trend continued. With 3 squares shown to the computer, a large area was blurred out in the middle of one test case. Additionally, it was noted that the algorithm opens up an area near the available area.

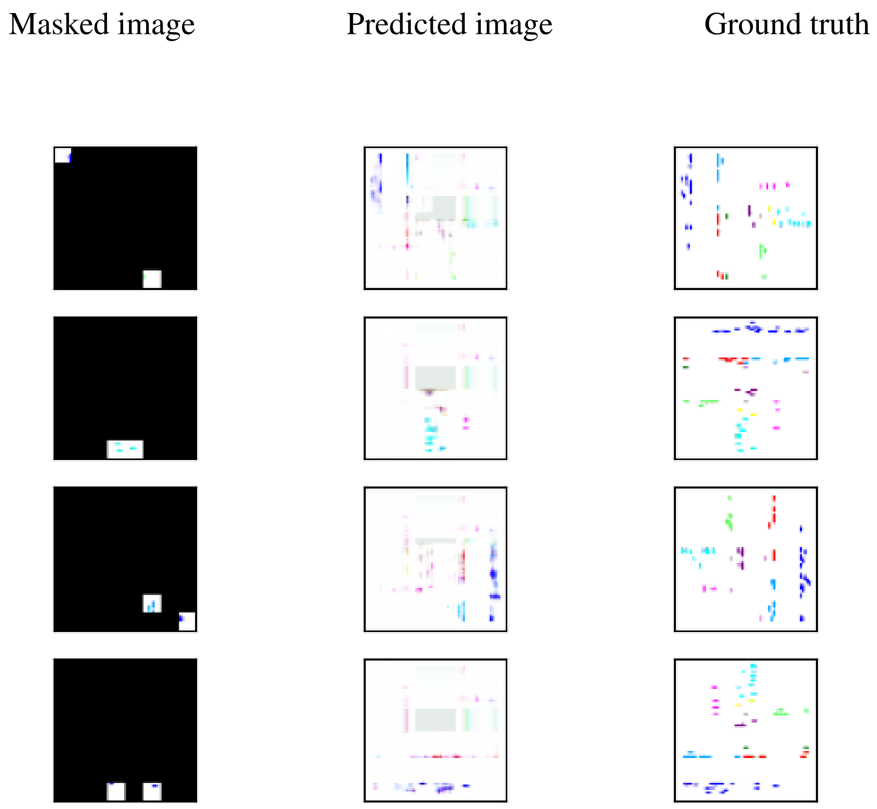

Figure 22. Results of predictions on multiple images with 2 squares visible.

In Figure 22, showing only 2 squares to the computer, the large blurred areas became more common. If we interpret the masked pictures, the results were minimal as only a few points were shown to the computer. Even a human operator would have difficulty predicting the rest of the picture.

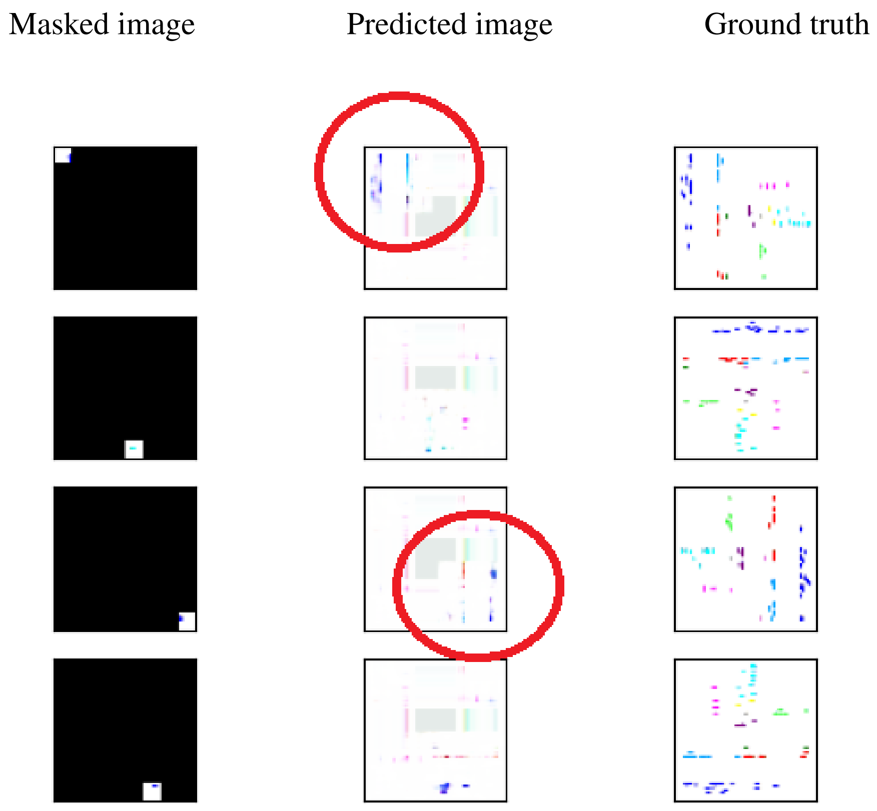

Figure 23. Results of predictions on multiple images with 1 square visible.

In Figure 23, one interesting observation was made when showing only 1 square to the computer. Although all of the predicted images were now without structure, the information around the shown square was able to be interpreted. The distance that was able to be interpreted correctly was 2 squares in every direction, including the diagonal direction. See Figure 23, where the visible area shown in red circles was mostly correct even though the values were minimal, the computer was less certain. An investigation was conducted using a new image that resulted from a prediction using 1 square to see if a better result could be achieved. To test this idea, the predicted image using 1 open square was used and the area nearby was colored black - meaning that 25% of the area was now visible to the computer. The only visible area this time was the most clear area from the previous prediction. Initially, doing another prediction with this new input image resulted in a similar result, which could be because the classes were already thin in the previous prediction. However, when applying a 3*3 sharpen kernel to the 1 square prediction result and using that instead, better results were achieved and the greyed out area was no longer seen in the middle. The blur kernel was also tested, but the result was worse than with initial image as shown in Figure 24.

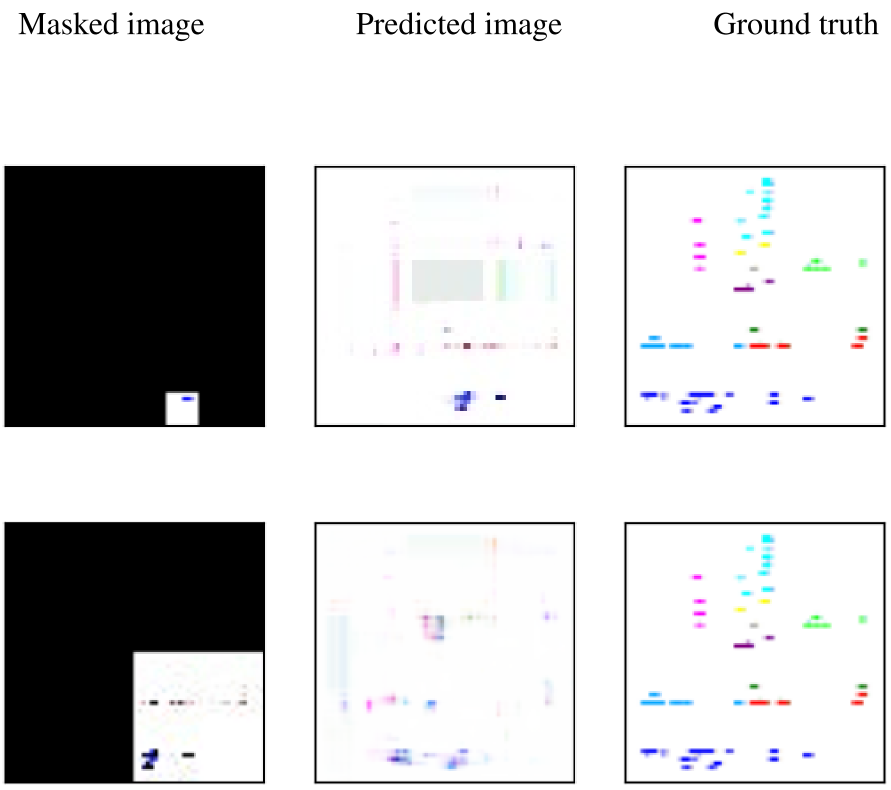

Figure 24. Using one prediction result as an input to the next prediction

The conclusion from those tests indicated that it is necessary to show the computer at least 7 squares of original data, which corresponds to 10.9%. Anything lower than that and there will either be large, blurred areas or missing data. Showing only 1 square, which corresponds to 1.5% data, opened up 20% of the whole structure near the shown square. Using one prediction as an input to the next, resulted in information not to be trusted and a sharpen kernel was useful to boost the quality of output.

To measure results of the trained model quality, the first assessment method was root mean square error (RMSE) conducted for each square, as the literature review suggested. The results can be seen in Figure 25. The results show, that showing each square less of data results in higher root mean square error.

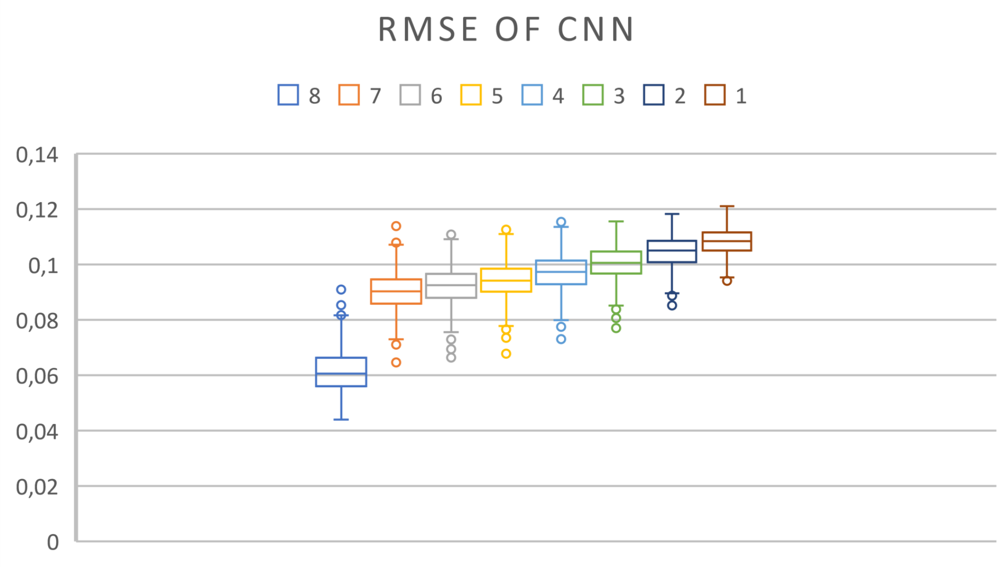

Figure 25. Root mean square error score on CNN for each shown square.

Then, another measurement was conducted using Structural Similarity Index (SSIM). The predictions were measured when showing squares from 1 to 8. The SSIM is a score that measures the similarity between an original and processed signal, based on a structural information [8] [48]. The results can be seen on Figure 26.

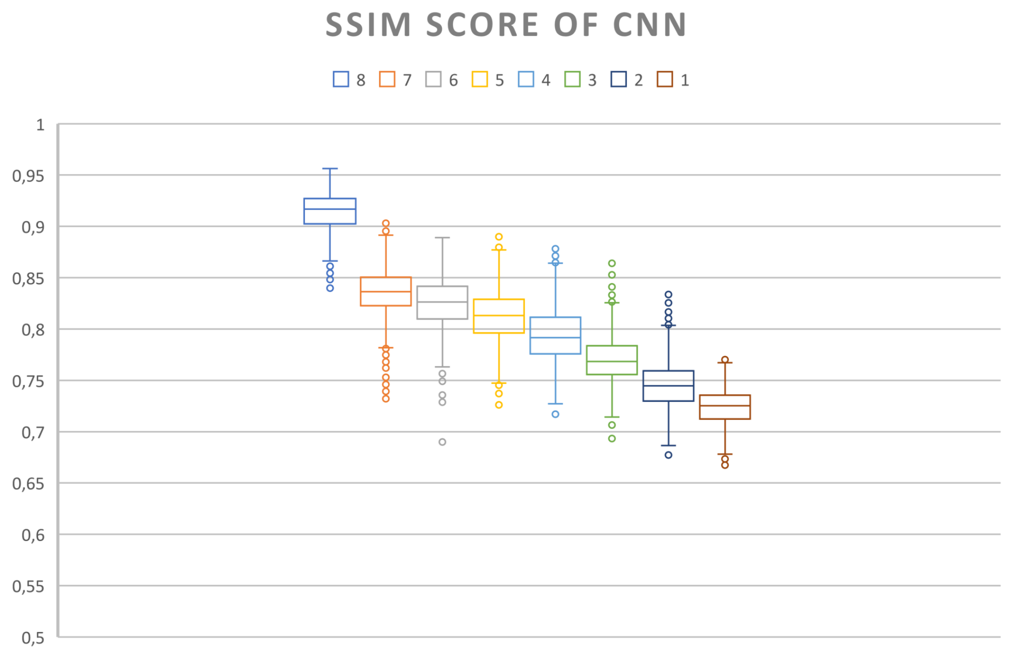

Figure 26. Structural similarity index measure score on CNN for each shown square.

6.3 Validation of the results using vision transformers

While previously an encoder-decoder network was used, training the autoencoders was next conducted. An autoencoder is also an encoder-decoder type neural network which consists of a bottleneck called a latent space. The key difference between a normal encoder-decoder network and an auto-encoder is that auto-encoders are a special case of encoder-decoder models, in which the input and output are the same [33]. In auto-encoder the information is encoded to the latent space, also known as a code and then decoded back to the original structure [45] [49] [2] This bottleneck is the smallest piece of the network [2]. Following the example in [5] and the paper [21], that states it is possible to reconstruct the data with up to 75 percent masking ratio, good results were obtained. This means that image can be input to the provided Vision Transformer (ViT) and a reconstructed image can be returned that is - similar to previously created CNN model prediction. Compared to a convolutional neural network which operates with convolutions on a regular grid, usually a cartesian grid where the objects are even squares, it is easier to integrate mask tokens or positional embeddings into Vision Transformers (ViT) [21]. The ViT-s are successors to convolutional neural networks and should thus be preferred [21]. So, following the example in [5], the generated data was saved into an uncompressed numpy array format. This had dimensions of 64 times 64 whereas the original example deals with pictures of 48x48. The model was tested and Figure 27 shows 3 pictures, the first of which is the original picture fed into the network. The second is the masked image and the third is the reconstructed or so called predicted image. As seen previously with the CNN, Figure 28 shows similar result but with smaller available masks shown to the computer.

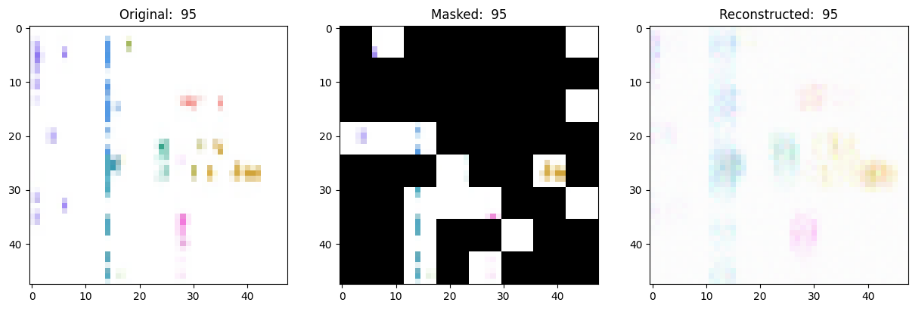

Figure 27. Masked image modeling autoencoder pretraining

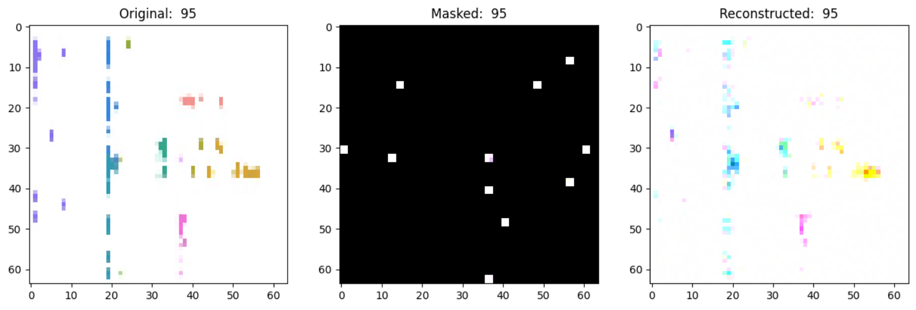

Figure 28. Masked image modeling autoencoder pretraining with smaller patch sizes

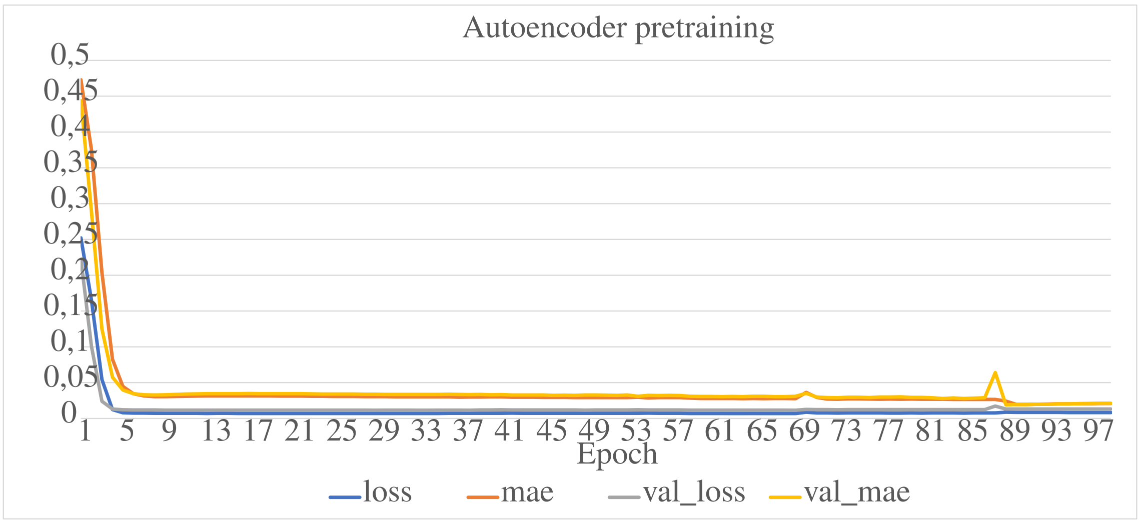

Figure 29. Training loss on autoencoder during pretraining.

The autoencoder results are similar to the CNN results. It is to be noted from this test that smaller available masks assist the computer to predict the output image with a higher resolution. As it is desirable to have a little variety in the generated data, as opposed to the replicated data, no further testing with autoencoders was done. The conditional generative adversarial networks was tested next,

6.4 Validation of the results using GANs

Image-to-image task such as the one currently tested, fits conditional generative adversarial networks (cGANs), a subtype of GANs perfectly. This is, because its intention is to turn one image (masked image) into another (unmasked image) [18]. Using the pix2pix framework [24] and the tensorFlow core tutorial [43], it was possible to reproduce the results and train the network on our own dataset which resulted in a masked image being unmasked. The network was trained on 40 epochs where every 5 epochs a model was saved. In total 8 models were saved. As this time the training process was not aided with a GPU, each epoch took between 219 and 230 seconds as shown in Figure 30, where the average was 221.87 seconds. This means that the most accurate model was trained in less than 2.5 hours on a AMD Threadripper 3960X 24-Core/48-thread CPU (Central Processing Unit). The fluctuating time can easily be interpreted as the extra time required to save the model after each 5 epochs.

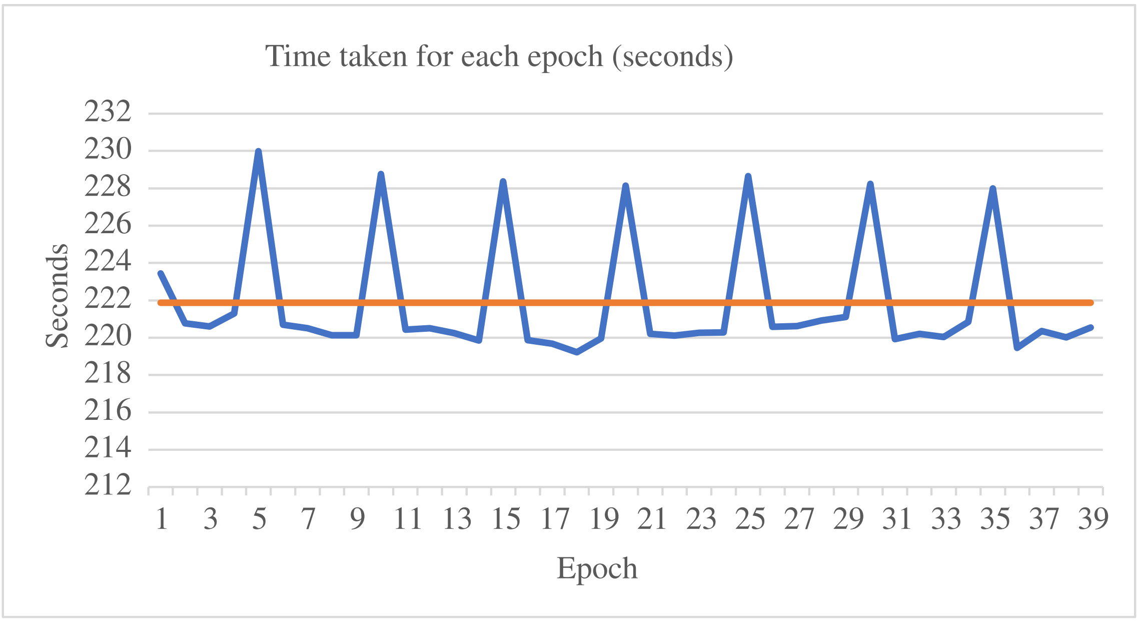

Figure 30. cGAN training process

The results of the cGAN prediction are shown on Figure 31, where similarly to the CNN example, it was observed that 4 predictions with 8 squares of visible data were seen on the trained model.

For comparison with the architecture of the CNN, the cGAN architecture consists of a generator (Figure 37) and a discriminator (Figure 38).

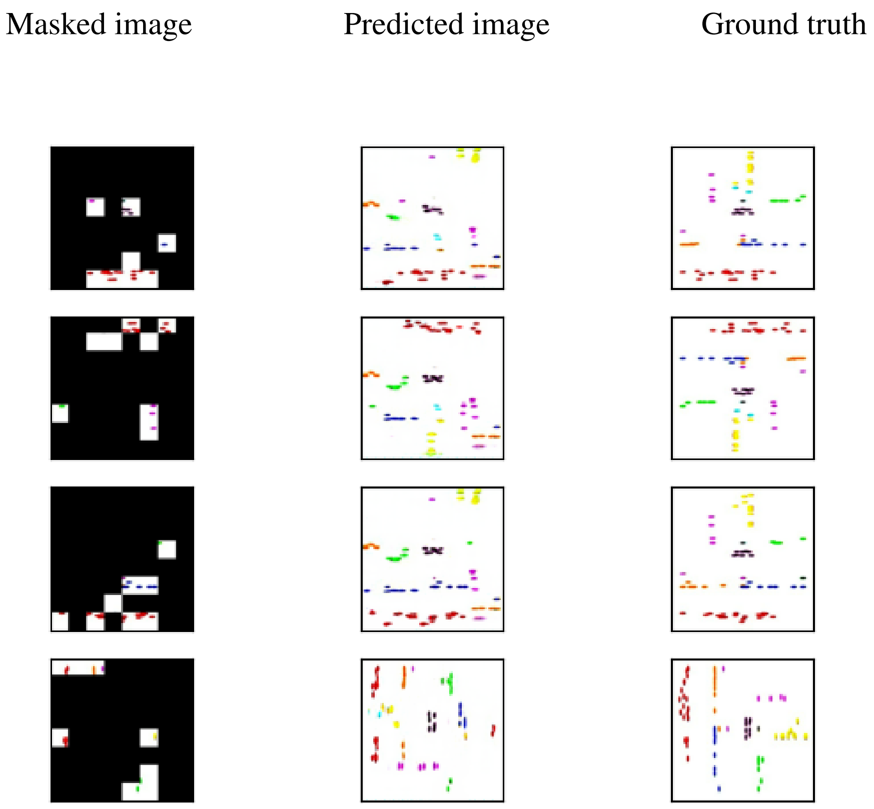

Figure 31. cGAN model prediction results after last epoch and 8 squares visible

It was observed, that the model worked because the facing directions were correct, but some of the predicted locations were in other locations compared to the original picture. This trend continues until the computer is shown only 1 square as seen on Figure 32.

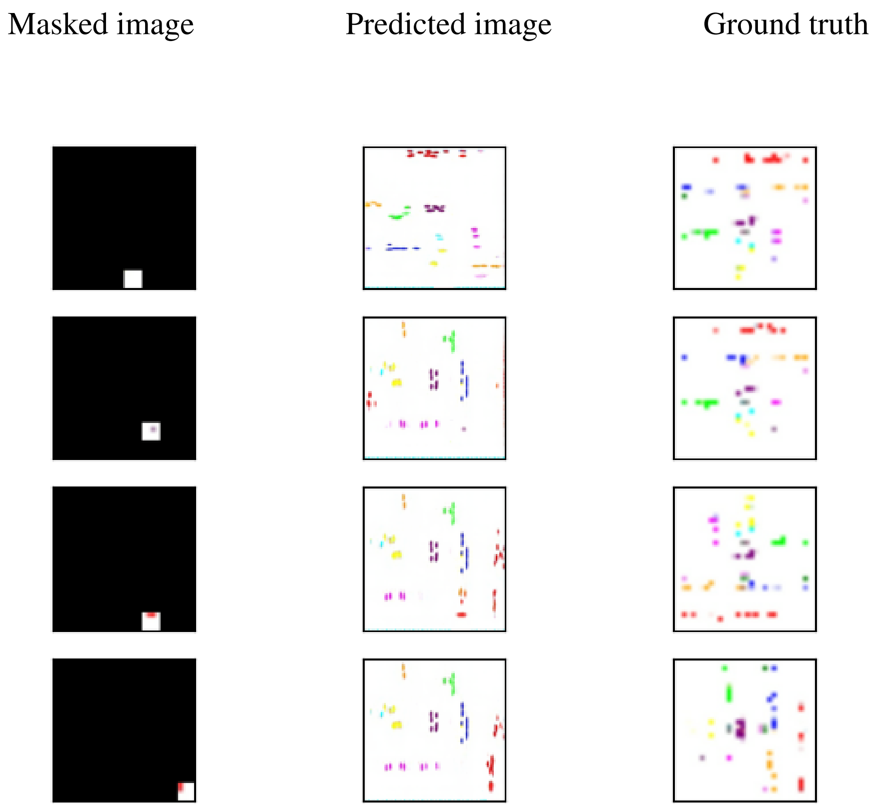

Figure 32. cGAN model prediction results after last epoch and 1 square visible

Unlike the CNN model that predicted what it knew with high certainty, the conditional GAN seemed to learn the basic structure and vary the elements around the whole picture. It knew the direction, because this was always correct with 8 squares shown. For comparison on Figure 33 we can see the results of predictions on the first saved cGAN model that got the directions wrong. It was also observed that there was an unexpected occurrence where the model tried to place one class in 2 directions. However, during the last epoch the model overcame this.

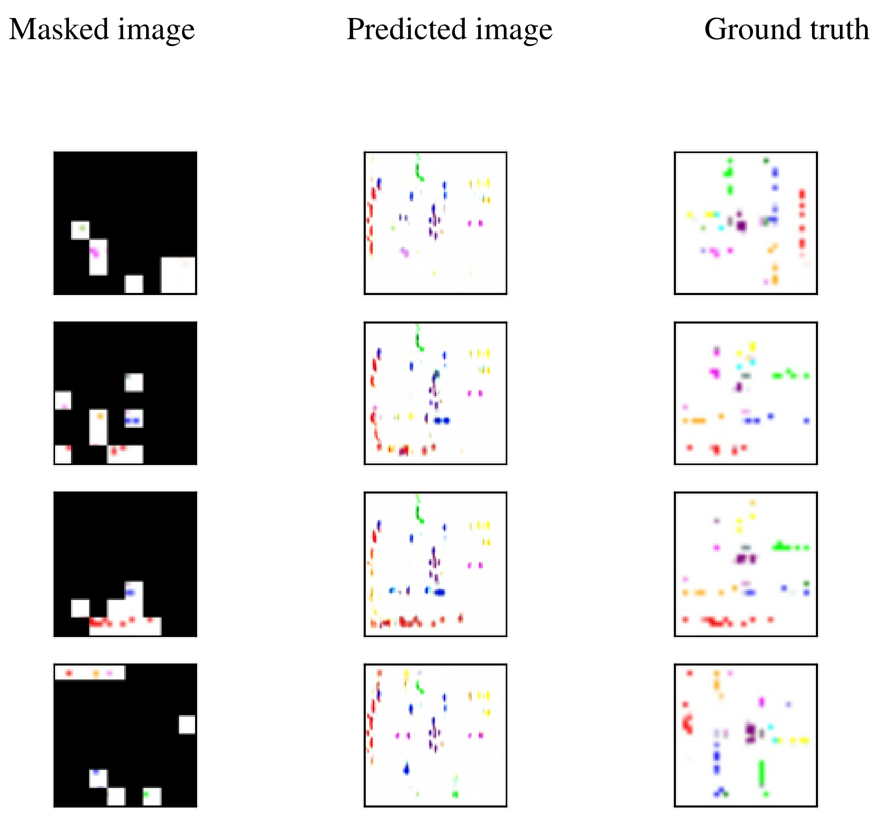

Figure 33. cGAN model prediction results after 5 epochs and 8 squares visible

To measure the quality of the CGAN model the same RMSE score was measured, but it appeared to not be a good measurement, because of the unique positioning of placement of the classes as seen in Figure 34.

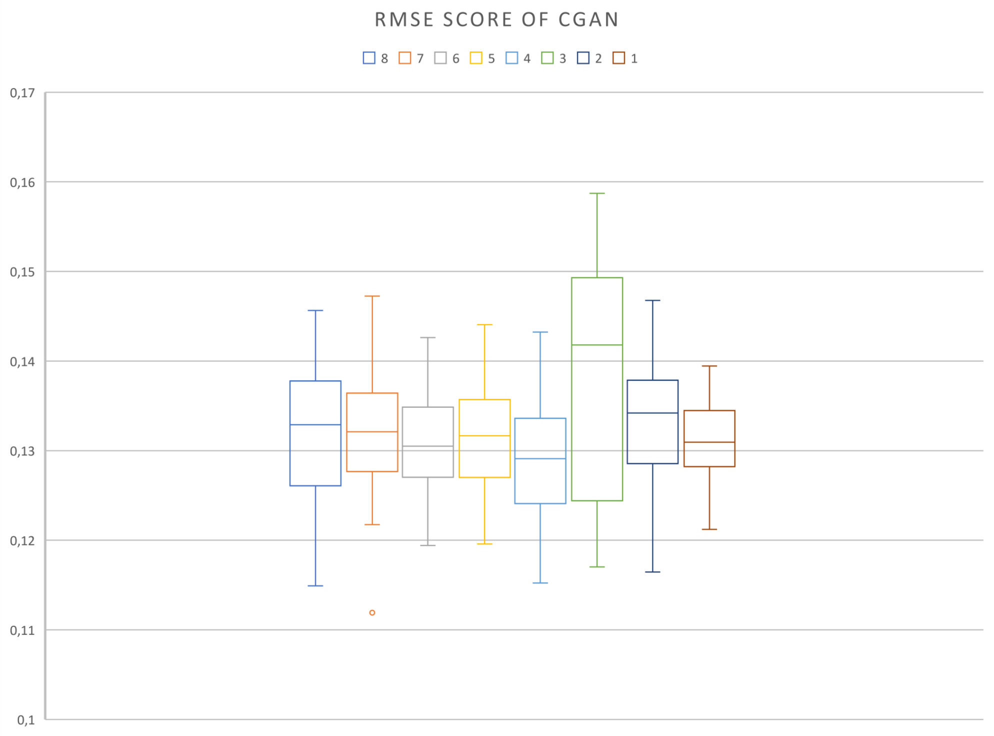

Figure 34. Root mean square error score on cGAN for each shown square.

To measure the quality of the results compared to the CNN more accurately the same SSIM score was chosen to be calculated for each prediction as with CNN. The average of each category from 1 square shown to the 8 was taken and the results are shown on Figure 35.

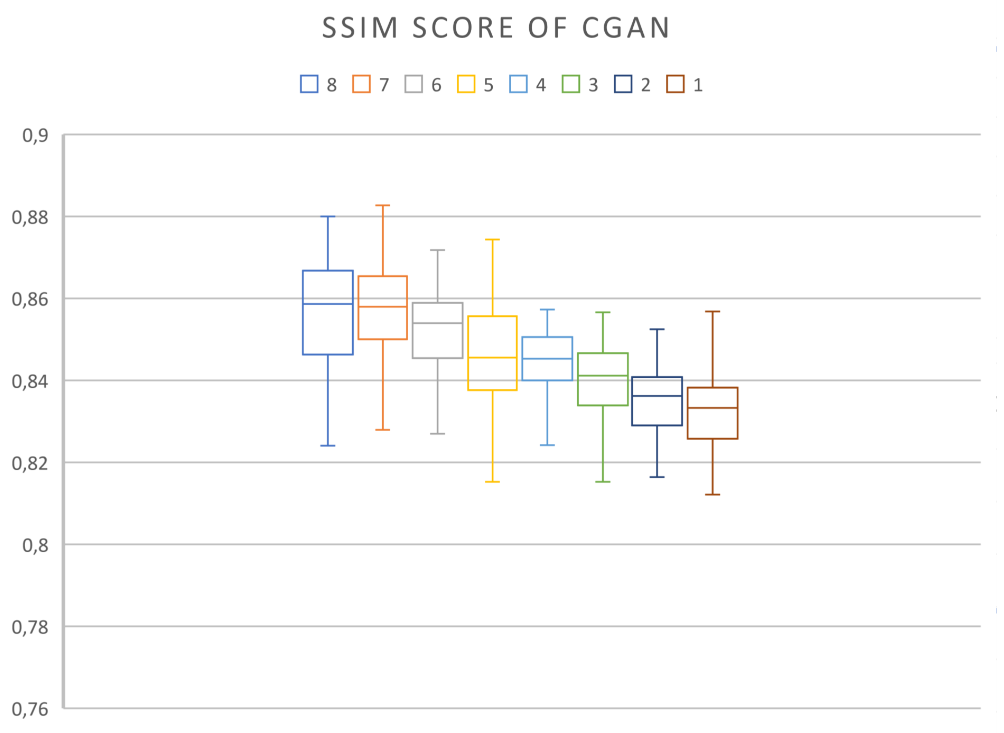

Figure 35. Structural similarity index measure score on CGAN for each shown square.

Because the gap between minimum and maximum values was quite low, and because the measure scores of 7 and 8 squares and 4 and 5 were very close to each other, as seen on Figure 35, another score was decided. This was called the VIF (visual information fidelity) score as shown in Figure 36. Both the visual information fidelity and structural similarity index measure are image quality assessment (IQA) scores.

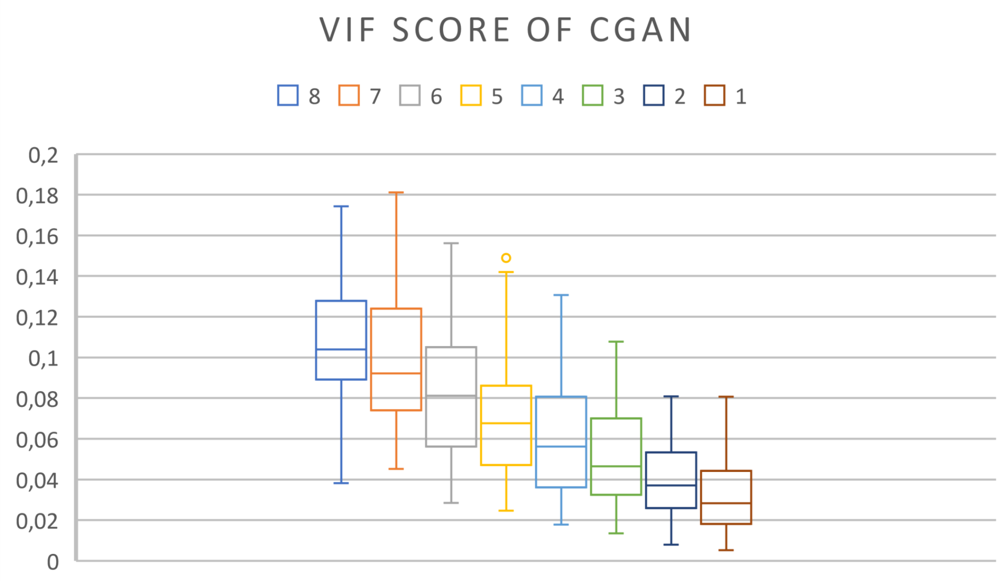

Figure 36. Visual Information Fidelity score on CGAN for each shown square.

This time the varying area is larger which makes less room for error. With an SSIM score, the difference between the minimum and maximum value was 0.07055, whilst with VIF, the difference between the minimum and maximum value was 0,17585. Additionally, it is possible to distinguish the different predictions by their score value more easily. SSIM indicates that there is a steeper drop in prediction quality when showing the computer less than 7 squares of data rather than between 8 and 7 squares of shown data. This was also the reason why not every prediction series with images was shown. For this reason, it is recommended to show the computer 7 or more squares of data. The best quality prediction model was the last trained model this time.

7. Discussion

One of the goals of this thesis was to build a generator of adversary forces positioning in case of missing data. After reviewing the literature, it was apparent that the generator had to be built from first principles. Having considered how to distribute the positions of the units based on their type and taking into consideration the structure of the army, the generator was built. A number of prediction methods were then tested using convolutional neural networks, generative adversarial networks and transfer learning.

It was discovered that convolutional neural networks work quite well in predicting the unit's positions and direction. It achieved available data positions of only 10.9%, which translated to 7 visible squares out of 64. It was also discovered that it is possible to get better results by reusing the prediction result.

Subsequently, to explore other networks transfer learning was tested, which resulted in similar positioning results as the convolutional networks did. After discovering this, the generative adversarial networks were tested next because of their ability to interpret image data. This was a good decision as the generative adversarial networks resulted in the whole area prediction. This was achieved even when only minimal positioning data was shown to the computer, whilst CNN-s only predicted the area around the known square. Additionally, compared to the CNN-s and transfer learning, the generative adversarial network had little variety in the results. However, similarly to the CNN, the sweet spot was 10.9%, or 7 squares of visible data, which achieved good results.

Even though the augmentation of additional data was proven, additional consideration of the area could be added in future to the units. This would include parameters such as pathfinding, speed, or weather conditions of the locations, some of which were discussed earlier. This would improve the situational awareness of the generator.

8. Conclusion

Within this thesis, a synthetic data generator was built with the goal of predicting the positioning of adversary units. To discover possible solutions, a literature review was conducted in order to use the correct techniques. Having discovered that the generator had to be built from first principles, the coordinate positions of the units were generated using Normal and Dirichlet distributions.

Additionally, in order to explore ways of adding further data to the generated units, a post processing step was done that compared units against a topographical map.

By randomly hiding parts of generated data, a number of machine learning techniques were used for predicting the hidden units. The predictions were conducted using conditional neural networks (CNNs) and generative adversarial networks (GANs), of which the CNNs were built from scratch. It was discovered that CNN-s and GAN-s both require the same amount of visible positioning data in order to return a good prediction. This was found to be 10.9%. The difference of the two being that CNN-s, having minimal amount of data return with high certainty the area around of known points and with less certainty the area further away. However, GAN-s return the whole area prediction with a variety, even if the direction was not correct.

In conclusion it can be determined that the generated data had been successfully used to predict the location of military units. This demonstrated the achievement of the aim of this thesis and its potential for further development.

References

[1] whuber (https://stats.stackexchange.com/users/919/whuber). *How to construct a multivariate Beta distribution?* Cross Validated. URL: https://stats.stackexchange.com/q/549262.

[2] C.C. Aggarwal. *Neural Networks and Deep Learning: A Textbook*. Springer International Publishing, 2018. ISBN: 9783319944630. URL: https://books.google.ee/books?id=achqDwAAQBAJ.

[3] Charu C. Aggarwal. *Data Mining*. Springer International Publishing, 2015. DOI: 10.1007/978-3-319-14142-8. URL: https://doi.org/10.1007%2F978-3-319-14142-8.

[4] Plamen Angelov and Eduardo Soares. “Towards explainable deep neural networks (xDNN)”. In: *Neural Networks* 130 (2020), pp. 185–194. ISSN: 0893-6080. DOI: https://doi.org/10.1016/j.neunet.2020.07.010. URL: https://www.sciencedirect.com/science/article/pii/S0893608020302513.

[5] Sayak Paul Aritra Roy Gosthipaty. *Masked image modeling with Autoencoders*. 221. URL: https://keras.io/examples/vision/masked_image_modeling/.

[6] *ARMOR- AND MECHANIZED-BASED OPPOSING FORCE ORGANIZATION GUIDE*. 1997. URL: https://man.fas.org/dod-101/sys/land/row/100-60.pdf.

[7] Mohammadamin Barekatain et al. “Okutama-Action: An Aerial View Video Dataset for Concurrent Human Action Detection”. In: *CoRR* abs/1706.03038 (2017). arXiv: 1706.03038. URL: http://arxiv.org/abs/1706.03038.

[8] Tianliang Bian. “An ensemble image quality assessment algorithm based on deep feature clustering”. In: *Signal Processing: Image Communication* 81 (2020), p. 115703. ISSN: 0923-5965. DOI: https://doi.org/10.1016/j.image.2019.115703. URL: https://www.sciencedirect.com/science/article/pii/S0923596519306435.

[9] Abdelaziz Botalb et al. “Contrasting Convolutional Neural Network (CNN) with Multi-Layer Perceptron (MLP) for Big Data Analysis”. In: *2018 International Conference on Intelligent and Advanced System (ICIAS)*. 2018, pp. 1–5. DOI: 10.1109/ICIAS.2018.8540626.

[10] Nadia Burkart and Marco F. Huber. “A Survey on the Explainability of Supervised Machine Learning”. In: *CoRR* abs/2011.07876 (2020). arXiv: 2011.07876. URL: https://arxiv.org/abs/2011.07876.

[11] S. van Buuren. *Flexible Imputation of Missing Data*. A Chapman & Hall book. CRC Press, 2018. ISBN: 9781138588318. URL: https://books.google.ee/books?id=bLmItgEACAAJ.

[12] Francois Chollet. *Building Autoencoders in Keras*. 2016. URL: https://blog.keras.io/building-autoencoders-in-keras.html.

[13] Meha Desai and Manan Shah. “An anatomization on breast cancer detection and diagnosis employing multi-layer perceptron neural network (MLP) and Convolutional neural network (CNN)”. In: *Clinical eHealth* 4 (2021), pp. 1–11. ISSN: 2588-9141. DOI: https://doi.org/10.1016/j.ceh.2020.11.002. URL: https://www.sciencedirect.com/science/article/pii/S2588914120300125.

[14] NumPy Developers. *NumPy Reference*. 2008-2022. URL: https://numpy.org/doc/stable/reference/index.html.

[15] O. Duerr, B. Sick, and E. Murina. *Probabilistic Deep Learning: With Python, Keras and TensorFlow Probability*. Manning Publications, 2020. ISBN: 9781617296079. URL: https://books.google.ee/books?id=-bYCEAAAQBAJ.

[16] Arturo Erdely and Manuel Rubio-Sanchez. *Copula-based statistical dependence visualizations*. [Online; accessed 08-April-2023]. 2022. URL: https://arxiv.org/pdf/2204.00265.pdf.

[17] Rúben Geraldes et al. “UAV-Based Situational Awareness System Using Deep Learning”. In: *IEEE Access* 7 (2019), pp. 122583–122594. DOI: 10.1109/ACCESS.2019.2938249.

[18] Ian Goodfellow et al. “Generative Adversarial Networks”. In: *Commun. ACM* 63.11 (Oct. 2020), pp. 139–144. ISSN: 0001-0782. DOI: 10.1145/3422622. URL: https://doi.org/10.1145/3422622.

[19] David Gunning and David Aha. “DARPA’s Explainable Artificial Intelligence (XAI) Program”. In: *AI Magazine* 40.2 (June 2019), pp. 44–58. DOI: 10.1609/aimag.v40i2.2850. URL: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/2850.

[20] Haibo He et al. “ADASYN: Adaptive synthetic sampling approach for imbalanced learning”. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. 2008, pp. 1322–1328. DOI: 10.1109/IJCNN.2008.4633969.

[21] Kaiming He et al. *Masked Autoencoders Are Scalable Vision Learners*. 2021. arXiv: 2111.06377 [cs.CV].

[22] Tin Kam Ho. "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Vol. 1. 1995, 278–282 vol.1. DOI: 10.1109/ICDAR.1995.598994.

[23] The MathWorks Inc. *MATLAB version: 9.13.0 (R2022b)*. Natick, Massachusetts, United States, 2022. URL: https://www.mathworks.com.

[24] Phillip Isola et al. *Image-to-Image Translation with Conditional Adversarial Networks*. 2018. arXiv: 1611.07004 [cs.CV].

[25] Dennis Jankowski, Arne Lamm, and Axel Hahn. "Determination of AIS Position Accuracy and Evaluation of Reconstruction Methods for Maritime Observation Data". In: *IFAC-PapersOnLine* 54.16 (2021). 13th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2021, pp. 97–104. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2021.10.079. URL: https://www.sciencedirect.com/science/article/pii/S2405896321014816.

[26] Krishna Kalyanathaya and Krishna Prasad Karani. "A Literature Review and Research Agenda on Explainable Artificial Intelligence (XAI)". In: *International Journal of Applied Engineering and Management Letters* (Feb. 2022), pp. 43–59. DOI: 10.47992/IJAEML.2581.7000.0119.

[27] Diederik P. Kingma and Max Welling. "An Introduction to Variational Autoencoders". In: *CoRR* abs/1906.02691 (2019). arXiv: 1906.02691. URL: http://arxiv.org/abs/1906.02691.

[28] Donna Kirk. *8.6 the normal distribution - contemporary mathematics*. Mar. 2023. URL: https://openstax.org/books/contemporary-mathematics/pages/8-6-the-normal-distribution.

[29] Junying Li et al. *Deep Rotation Equivariant Network*. 2018. arXiv: 1705.08623 [cs.CV].

[30] Donald MacKenzie and Taylor Spears. "'The formula that killed Wall Street': The Gaussian copula and modelling practices in investment banking". In: *Social Studies of Science* 44.3 (2014). PMID: 25051588, pp. 393–417. DOI: 10.1177/0306312713517157. URL: https://doi.org/10.1177/0306312713517157.

[31] Basim Mahbooba et al. "Explainable Artificial Intelligence (XAI) to Enhance Trust Management in Intrusion Detection Systems Using Decision Tree Model". In: *Complex.* 2021 (Jan. 2021). ISSN: 1076-2787. DOI: 10.1155/2021/6634811. URL: https://doi.org/10.1155/2021/6634811.

[32] Mathworks. *Copulas: Generate Correlated Samples*. [Online; accessed 08-April-2023]. 2023. URL: https://www.mathworks.com/help/stats/copulas-generate-correlated-samples.html.

[33] Shervin Minaee et al. “Image Segmentation Using Deep Learning: A Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.7 (2022), pp. 3523–3542. DOI: 10.1109/TPAMI.2021.3059968.

[34] Antonio Mora, Pedro Castillo, and Juan Merelo Guervós. “Resolution of the bi-criteria military unit pathfinding problem applying MOACO algorithms”. In: *Military Operations, Health and Technology* (Nov. 2012), pp. 55–88.

[35] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[36] Miriam Seoane Santos et al. “Generating Synthetic Missing Data: A Review by Missing Mechanism”. In: *IEEE Access* 7 (2019), pp. 11651–11667. DOI: 10.1109/ACCESS.2019.2891360.

[37] Fabio Henrique Kiyoiti dos Santos Tanaka and Claus Aranha. “Data Augmentation Using GANs”. In: *CoRR* abs/1904.09135 (2019). arXiv: 1904.09135. URL: http://arxiv.org/abs/1904.09135.

[38] Anuraganand Sharma, Prabhat Kumar Singh, and Rohitash Chandra. “SMOTified-GAN for class imbalanced pattern classification problems”. In: *CoRR* abs/2108.03235 (2021). arXiv: 2108.03235. URL: https://arxiv.org/abs/2108.03235.

[39] Connor Shorten and Taghi Khoshgoftaar. “A survey on Image Data Augmentation for Deep Learning”. In: *Journal of Big Data* 6 (July 2019). DOI: 10.1186/s40537-019-0197-0.

[40] Michael Steinbach, George Karypis, and Vipin Kumar. “A Comparison of Document Clustering Techniques”. In: *Proceedings of the International KDD Workshop on Text Mining* (June 2000).

[41] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. “Yago: A Core of Semantic Knowledge”. In: *16th International Conference on the World Wide Web*. 2007, pp. 697–706.

[42] Herb Susmann. *Dirichlet Distribution*. May 2021. URL: https://observablehq.com/@herbps10/dirichlet-distribution.

[43] Tensorflow. *pix2pix: Image-to-image translation with a conditional GAN*. 2022. URL: https://www.tensorflow.org/tutorials/generative/pix2pix.

[44] Sarang Thombre et al. “Sensors and AI Techniques for Situational Awareness in Autonomous Ships: A Review”. In: *IEEE Transactions on Intelligent Transportation Systems* 23.1 (2022), pp. 64–83. DOI: 10.1109/TITS.2020.3023957.

[45] Elena Trunz et al. “Efficient structuring of the latent space for controllable data reconstruction and compression”. In: *Graphics and Visual Computing* 7 (2022), p. 200059. ISSN: 2666-6294. DOI: https://doi.org/10.1016/j.gvc.2022.200059. URL: https://www.sciencedirect.com/science/article/pii/S2666629422000122.

[46] Vedran. *OpenClipart*. URL: https://freesvg.org/contour-map.

[47] Xingrui Wang et al. “Large Scale GPS Trajectory Generation Using Map Based on Two Stage GAN”. In: *Journal of Data Science* 19.1 (2021), pp. 126–141. ISSN: 1680-743X. DOI: 10.6339/21-JDS1004.

[48] Zhou Wang et al. “Image quality assessment: from error visibility to structural similarity”. In: *IEEE Transactions on Image Processing* 13.4 (2004), pp. 600–612. DOI: 10.1109/TIP.2003.819861.

[49] Tom White. “Sampling Generative Networks: Notes on a Few Effective Techniques”. In: *CoRR* abs/1609.04468 (2016). arXiv: 1609.04468. URL: http://arxiv.org/abs/1609.04468.

[50] Jürgen Wiest et al. “Probabilistic trajectory prediction with Gaussian mixture models”. In: *2012 IEEE Intelligent Vehicles Symposium*. 2012, pp. 141–146. DOI: 10.1109/IVS.2012.6232277.

[51] Manzhu Yu et al. “Deep learning for real-time social media text classification for situation awareness – using Hurricanes Sandy, Harvey, and Irma as case studies”. In: *International Journal of Digital Earth* 12.11 (2019), pp. 1230–1247. DOI: 10.1080/17538947.2019.1574316. eprint: https://doi.org/10.1080/17538947.2019.1574316. URL: https://doi.org/10.1080/17538947.2019.1574316.

[52] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. “Interpretable Convolutional Neural Networks”. In: *CoRR* abs/1710.00935 (2017). arXiv: 1710.00935. URL: http://arxiv.org/abs/1710.00935.

[53] Yu Zheng et al. *Geolife GPS trajectory dataset - User Guide*. Geolife GPS trajectories 1.1. July 2011. URL: https://www.microsoft.com/en-us/research/publication/geolife-gps-trajectory-dataset-user-guide/.

[54] Qing Zhu. “Research on Road Traffic Situation Awareness System Based on Image Big Data”. In: *IEEE Intelligent Systems* 35.1 (2020), pp. 18–26. DOI: 10.1109/MIS.2019.2942836.

Appendix 1 – Non-Exclusive License for Reproduction and Publication of a Graduation Thesis[1]

[1]The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be valid for the period.

Appendix 2 – cGAN generator

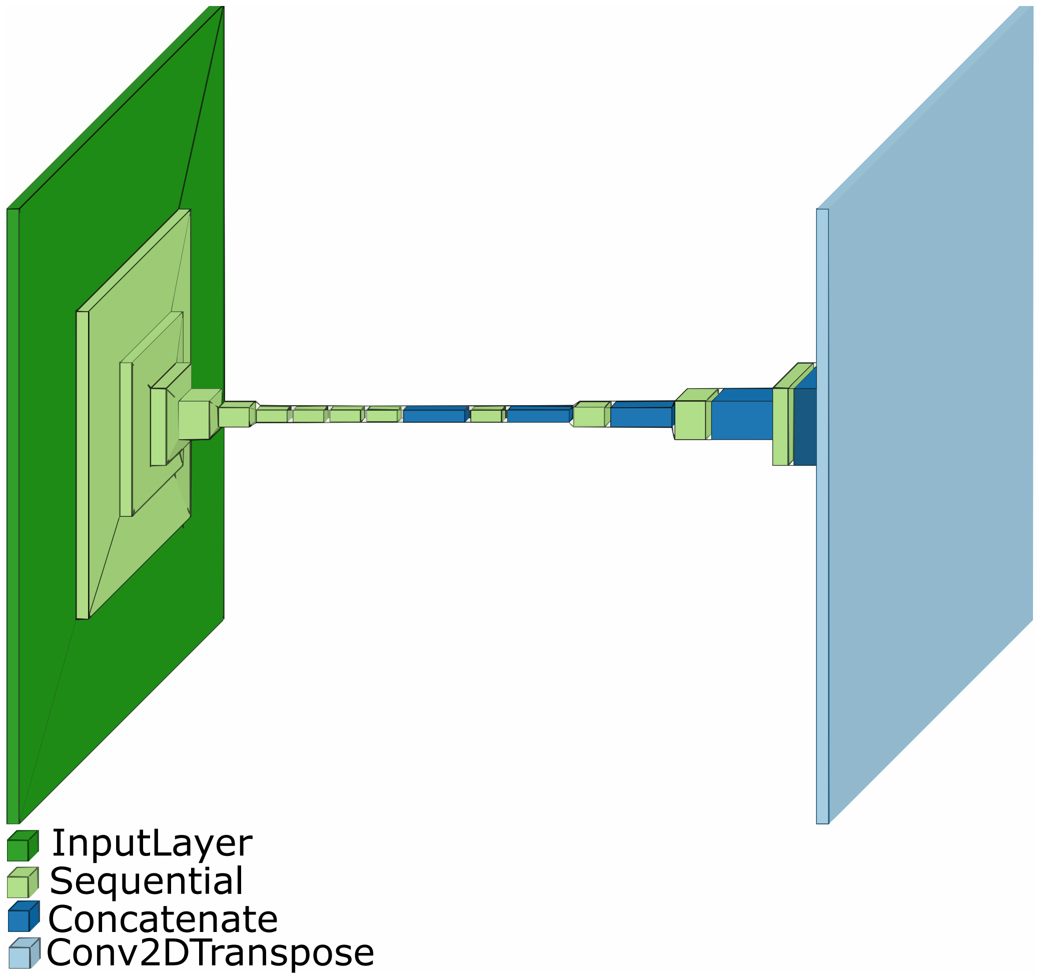

Figure 37. cGAN model generator architecture.

Appendix 3 – cGAN discriminator

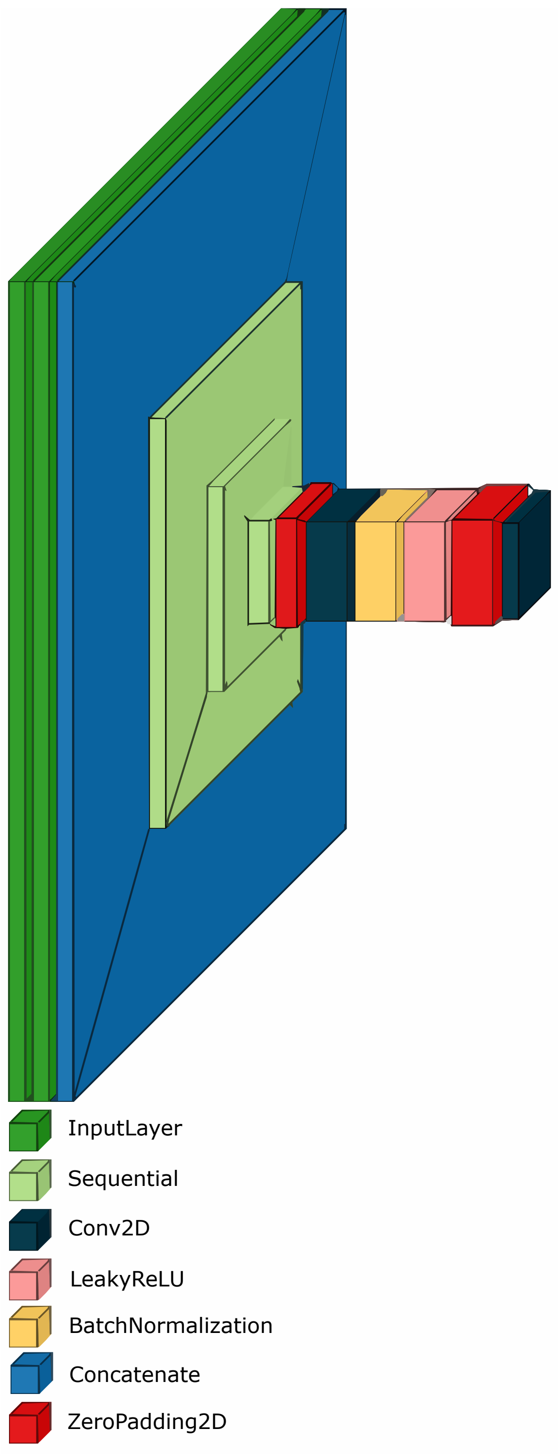

Figure 38. cGAN model discriminator architecture.

Appendix 4 – The formulas

Normal distribution

$$P(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2 / 2\sigma^2} \tag{1}$$

Probability density function of dirichlet distribution

$$f(\mathbf{x}|\boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{i=1}^{K} x_i^{\alpha_i - 1} \tag{2}$$

Gini index impurity

$$G = 1 - \sum_{i=1}^{n} p_i^2 \tag{3}$$