Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Arild Sørensen Dalsgård

Segmentation of river ecology using deep learning on aerial images

Master's thesis in Computer Science
Supervisor: Odd Erik Gundersen
June 2020

NTNU
Norwegian University of Science and Technology

Arild Sørensen Dalsgård

Segmentation of river ecology using deep learning on aerial images

Master's thesis in Computer Science
Supervisor: Odd Erik Gundersen
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Kunnskap for en bedre verden

Abstract

A convolutional neural network for image segmentation of black-and-white aerial images is presented. The network can segment the images into ecological relevant segments for further analysis on river ecology from the 1940s to the 2000s. The main contributions of this thesis are 1: A convolutional neural network model for image segmentation. 2: A manually annotated dataset having 7234 512x512 pixel images covering 76 km^2 of land. The model got a mean intersection over union of 71% on the test set from the same river as it was trained on, and a mean intersection over union of 77% on a test set from a river that it did not train on.

Norwegian abstract: Et konvolusjonelt nevralt nettverk for bildesegmentering av svart-hvitt luftfoto er presentert. Nettverket kan segmentere bildene i økologiske relevante segmenter for videre analyse av elveøkologi fra 1940- til 2000-tallet. Hovedbidragene til denne oppgaven er 1: A konvolusjonelt nevralt nettverk modell for bildesegmentering. 2: Et manuelt merket datasett med 7234 512x512 piksel bilder som dekker 76 km^2 land. Modellen fikk et gjennomsnittlig snitt over union på 71% på testsettet fra samme elv som den ble trent på, og et gjennomsnittlig snitt over union på 77% på et testsett fra en elv som den ikke trente på.

Preface

This project is conducted at the Norwegian university of science and technology in Trondheim, Norway, for the Department of Computer Science and in cooperation with the Department of Civil and Environmental Engineering.

The research was conducted by me, Arild Dalsgård, with supervision from associate professor Odd Erik Gundersen. He provided guidance to the research process and the technical aspects related to machine learning. In addition to this, professor Knut Alfredsen and research fellow Jo Halvard Halleraker, who both work at the Department of Civil and Environmental Engineering, guided the project to meet the requirements in the ecological domain.

Arild Sørensen Dalsgård
Trondheim, June 8, 2020

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mapping the ecological conditions of rivers is a slow and expensive process when done in the field and it often misses the bigger picture. Automated methods such as using drones to get high resolution images requires extra resources and time to setup the drones and flight paths (Casado [2015]). Both methods lack the ability to be used on a large scale making it difficult for use in a global or even national ecological analysis. In addition, these techniques for data gathering lacks the ability to take information from past years into account to see the changes over time and the effects of human development on the ecological conditions in rivers.

The data available from before year 2000 in Norway is black-and-white aerial images (Kartverket [2019a]). These are scanned versions of analog images and they have a resolution of 0.2 meters. They cover large areas of Norway from the years 1935 to the 2000s.

With the rise of powerful new techniques for image segmentation using deep convolution networks there are new opportunities for making segmentation models that takes advantage of the context as well as individual pixel values. This allows for data with low level of information per pixel to be used.

This study presents a software system for automatically segmenting rivers from black-and-white aerial images and experiments for determining the best model.

1.1 Background and Motivation

Doing manual ecological analysis using field trips takes time and resources and covers only small sections of rivers in a working day. It also runs the risk of missing the bigger picture by only focusing on a small part of a single river. Some work has been done at making systems that automatically can segment rivers using

satellite images Murray et al. [2019], but they lack the high resolution that aerial imagery offers and they don't go as far back in time as aerial images does. Using historical black-and-white aerial images has both a high resolution and exists for the time periods that are interesting for a historical analysis. In order to do analysis on a large scale an automated system is needed.

Goals and Research Questions

The aim of my research will be to create a machine learning system that can map the ecological conditions of rivers, based on widely available aerial images that covers large geographical areas and different time periods.

The goal is defined as the overall goal that the research question **RQ1** will can help answer. Other approaches for achieving the goal could be possible, but this thesis will focus on **RQ1**.

Goal Find out if and how ecological analysis of rivers can be automated.

- **RQ1:** Can an approach using black-and-white aerial images and neural networks divide rivers into ecological segments?
  - RQ1.1 What work have been done on black and white image segmentation?
  - RQ1.2 What is the best model architecture for the segmentation?

1.2 Research Method

The focus of this thesis is to answer **RQ1**. A literature review (see Section 2.2) was conducted to set this work into context with previous works and to influence the model and data processing steps. The literature review will answer **RQ1.1**.

An automatic system to divide rivers into ecological segments were made. A image is segmented by doing a per pixel classification into one of 5 classes (see Table 1.1 for the class definitions). The class definitions were developed in cooperation with domain experts. Experiments to find the best model architecture were done and validated with a dedicated validation set. The best models according to the scores on the validation set was used to expand the dataset and train a new set of models. The best model from the new set of models according to the validation set was selected for testing on the test sets. The test was conducted to determine how and if the model could be used for ecological analysis, as well as determine its strengths and weaknesses.

Class name	Description
Water	Surface water
Gravel	Fine and coarse gravel and sand
Vegetation	Natural vegetation
Farmland	Land used for crops or livestock
Human-constructions	Buildings, roads or other man-made constructions
Unknown	All areas not marked as another class

Table 1.1: Definition of each class.

1.3 Contributions

The contributions of this thesis are a software system to answer **RQ1** and a dataset containing handmade annotations covering 76 km^2. The dataset was used for training and testing of the models. For more details on the contributions of this work see Section 6.3.

1.4 Thesis Structure

In Chapter 2 the background material for the rest of the thesis will be covered. A literature review will also be presented in Chapter 2.

The dataset collected will be presented in Chapter 3 as well as the details regarding how the annotations were made.

In Chapter 4 the system architecture will be shown. This includes the pre-processing steps and the model used, as well as validation and testing processes.

The experiments and their results will be covered in Chapter 5.

Finally the results will be discussed in Chapter 6 and some suggestions for future work will be presented.

Chapter 2

Background Theory

This chapter will cover the necessary background theory to understand the models and methods presented later in the thesis. A structured literature review process is presented, and related work is covered to put this thesis into context with other works.

2.1 Background Theory

This section will explain the background theory for this thesis. First regular feed forward neural networks will be presented and after that convolutional neural networks will be explained. Finally, a decoder encoder architecture will be presented.

2.1.1 Machine learning

Machine learning is the process of making a machine improve its performance by learning from data. For example, a machine can learn to improve the accuracy of classifying images of cats by training the machine on a large amount of cat images.

Machine learning is divided into two main subcategories, supervised and unsupervised learning. In supervised learning the machine is given an input data x to make predictions on and a ground truth y that is the correct prediction for x. The goal of the machine in this case is to learn the function $f(x) = y$ for new x and y pairs.

In unsupervised learning the machine is given an input x and has to group it with similar data. Unsupervised learning was not used in this thesis and will not be covered in more detail.

A model in context of supervised learning is a function that makes a prediction given a input. It is the function f. Many choices for models exist. A few of the most common models are Support Vector Machines (SVM), Decision Trees, Random forest and Neural Networks (NN). All of these models will adapt them self to a set of training data.

2.1.2 Neural networks

A neural network (NN) is made up by one or more layers of neurons. An input vector is given to the first layer of the neural network. This first layer is called the input layer. The elements of the input vector are denoted as x_i. Each neuron has an edge to each of the neurons in the next layer. Each of these edges has a trainable weight $w_{i,j,l}$, where i is the index for the neuron in layer $l-1$ and j is the index of the node in the l layer. l specifies the layer, starting at 0 for the input layer. The last layer is called the output layer. In Figure 2.1 an example of a neural network with one hidden layer are shown. Note that neural networks for real life problems usually are a lot larger, with more neurons in the hidden layer and multiple hidden layers.

The activation for neuron j in layer l is defined as

$$a_{j,l} = f(\sum_{i=1}^{\lambda(l-1)} (w_{i,j,l} \cdot a_{i,l-1}) + \theta_{j,l}) \tag{2.1}$$

where $\lambda(l)$ is the number of neurons in layer l, $\theta_{j,l}$ is a trainable bias term for neuron j in layer l. f is the activation function, for example ReLU. (ReLU is defined as $f(x) = max(0, x)$)

For the input layer the activation is defined as

$$a_{j,0} = x_j$$

, where x_j is the j-th element from the input vector.

Training

The training process for a NN is as follows:

1. Initialize the NN with random weights
2. Use the model to make a prediction for an input image/vector
3. Calculate the loss from the prediction using a ground truth, and adjust the weights using backpropagation

4. Repeat step 2 and 3 with new images/vectors until all images/vector are have been used

5. Repeat step 2, 3 and 4 until a minimum loss is reached

The loss function is a measure of the error of the model and the goal is to minimize the loss function. A common loss function for classification is Categorical Cross-Entropy (CCE).

$$\text{CCE} = -\sum_{i}^{C} t_i \cdot log(a_{i,L}) \tag{2.2}$$

where C is the number of classes, t_i is the ground truth for class i and $a_{i,L}$ is the softmax score from the NN for class i. The softmax activation function is defined as

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \tag{2.3}$$

for class i. Note that $f(s)_i \in [0, 1]$ and $\sum_i^C f(s)_i = 1$. The output after a softmax can therefore be thought of as a pseudo-probability of the class i being the true class ($t_i = 1$).

Backpropagation is a method to calculate how the weights of the layers before the last layer contributed to the loss and in what direction the weights should be adjusted.

To adjust the weights in a way that minimizes a loss function we can use the partial derivatives $\frac{\partial C}{\partial w}$ for each weight in the neural network. The strategy is then to adjust the weight $w_{i,j,l}$ by

$$w_{i,j,l} := w_{i,j,l} - \alpha \frac{\partial C}{\partial w_{i,j,l}}$$

, where α is the learning rate and C is the loss function.

To calculate the partial derivative $\frac{\partial L}{\partial w}$ we need a few equations from backpropagation. These equations are Equations (2.4) to (2.7) and they are taken from Nielsen [2015].

$$\delta_{j,L} = \frac{\partial C}{\partial a_{j,L}} \cdot \sigma'(z_{j,L}) \tag{2.4}$$

$$\delta_{j,l} = \sum_i w_{i,j}^{l+1} \cdot \delta_i^{l+1} \cdot \sigma'(z_j^l) \tag{2.5}$$

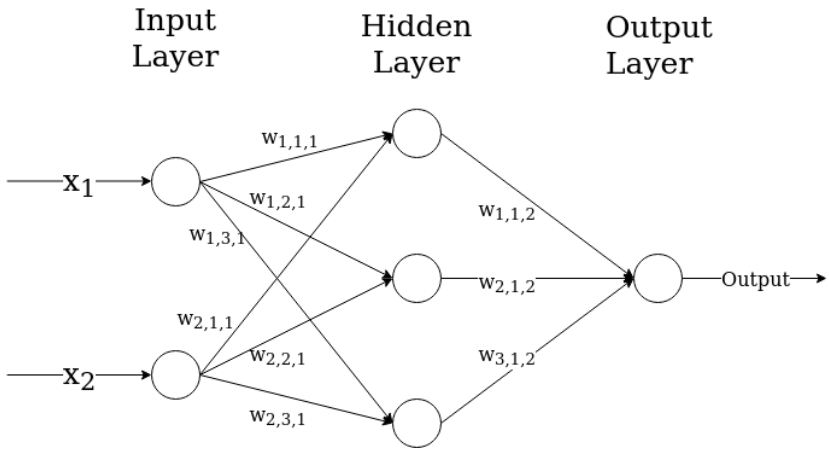


Figure 2.1: An example of a feed forward neural network with one hidden layer.

$$\frac{\partial C}{\partial \theta_{j,l}} = \delta_j^l \tag{2.6}$$

$$\frac{\partial C}{\partial w_{i,j,l}} = a_i^{l-1} \cdot \delta_{j,l} \tag{2.7}$$

where $z_{j,l}$ is the weighted input to neuron j in layer l, σ is the activation function, C is the loss function. δ_j^l is the error for neuron j in layer l.

$$z_{j,l} = \sum_{i=1}^{m} (w_{i,j,l} \cdot a_{i,l-1}) + \theta_{j,l}$$

Equation (2.4) gives us a way to compute the error for neurons in the output layer, and Equation (2.5) allows us to calculate the error for layer l if we have the errors for layer $l+1$. When both equations are combined, we can calculate the errors for every layer by starting at the output layer and working backwards. When we have all the errors we can calculate the gradient for the bias θ using Equation (2.6). Lastly the gradient can be calculated by using the activation computed in the forward pass and the error found in the backward pass in Equation (2.7). This process is called backpropagation.

2.1.3 Convolutional neural networks

Convolutional neural networks (CNN) have made huge advances and are now the most powerful techniques in computer vision. This section will cover how they

work.

CNNs were first presented in Fukushima [1988] but have had their breakthrough in computer vision with ImageNet (Krizhevsky et al. [2012]) outperforming traditional computer vision methods by a large margin. Deep CNNs are made by chaining multiple layers, where each layer extracts features from the image or changes the image size. The last layer is usually a softmax layer that assigns each class with an estimated probability, and the prediction is then the class with the highest estimated probability. In the following subsections three different kinds of layers are presented and explained. In Chapter 4 an architecture using the layers introduced in this section is presented.

2.1.4 Convolution Layer

A convolutional layer in a neural network have a set of filters. A filter is a 2 dimensional array with trainable weights $w_{i,j}$. A filter is applied on a input image (or feature map) by sliding the filter such that each of the pixels $p_{i,j}$ in the input image is in the center of the filter once. The filter is activated on each new location by computing this formula:

$$output(x, y) = \sum_{i=xStart}^{xStop} \sum_{j=yStart}^{yStop} p_{i,j} \cdot w_{i-xStart, j-yStart} \tag{2.8}$$

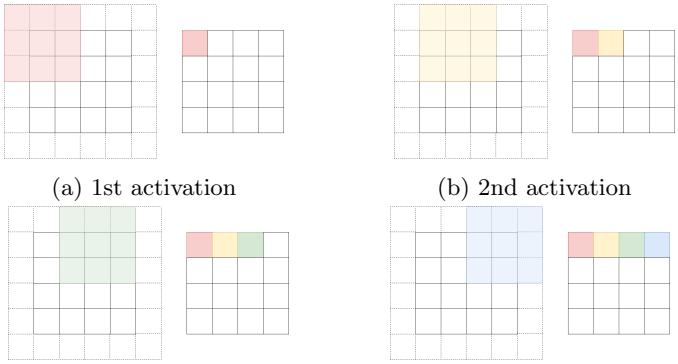
where

$$xStart = x - \lfloor fSize/2 \rfloor$$
$$xStop = x + \lfloor fSize/2 \rfloor$$
$$yStart = y - \lfloor fSize/2 \rfloor$$
$$yStop = y + \lfloor fSize/2 \rfloor$$

, $fSize$ is the size of the filter and is the dimension of the filter (often 3 or 5) and x, y is the location of the center pixel.

A new image is created with the same size as the input image but with $output(x, y)$ as the new pixel values. This new image is called a feature map. Each filter in the convolution layer creates a new feature map and each filter might have different weights producing different feature maps from the same input image. If the input to the convolutional layer have multiple channels, such as with RGB images or feature maps from previous layers, the filters of the convolutional layers will have the same number of channels. Let Z be the number of channels, then the filter activation is defined as

$$output(x, y) = \sum_{z}^{Z} \sum_{i=xStart}^{xStop} \sum_{j=yStart}^{yStop} p_{i,j,z} \cdot w_{i-xStart, j-yStart, z} \tag{2.9}$$



(a) 1st activation (b) 2nd activation

(c) 3rd activation (d) 4th activation

Figure 2.2: Example of an filter activation on the top row of a 4x4 input image with padding. The input image is shown on the left side of each subfigure and the output image is shown on the right.

where $p_{i,j,z}$ is the pixel at location i, j in layer z and $w_{i,j,z}$ is the weight at location i, j in layer z. The rest of the variables are as they were defined in Equation 2.8.

After the filters have been applied an activation function is used on the output from the filter activation. For each pixel Equation 2.10 is used.

$$activation\_output(x, y) = a(output(x, y) + b) \tag{2.10}$$

where x, y is the pixel location, $a(z)$ is the activation function, b is a trainable bias term and $output(x, y)$ is the result from the filter application defined in Equation 2.8 and 2.9.

Common activation functions are Rectified Linear Unit (ReLU) and tanh. $ReLU(x) = \max(0, x)$ Activation functions are used to add non-linearity to the network.

2.1.5 Pooling layer

A pooling layer takes an input image and returns a down sampled version of the image. There are two main types of pooling layers, max pooling and average pooling. A max pooling layer slides a n x n window over the input image by moving the window n pixels at a time. A window is activated before each movement and in the case of max pooling the maximum value in the window is returned. The return value is saved as one of the pixels of the output image. The

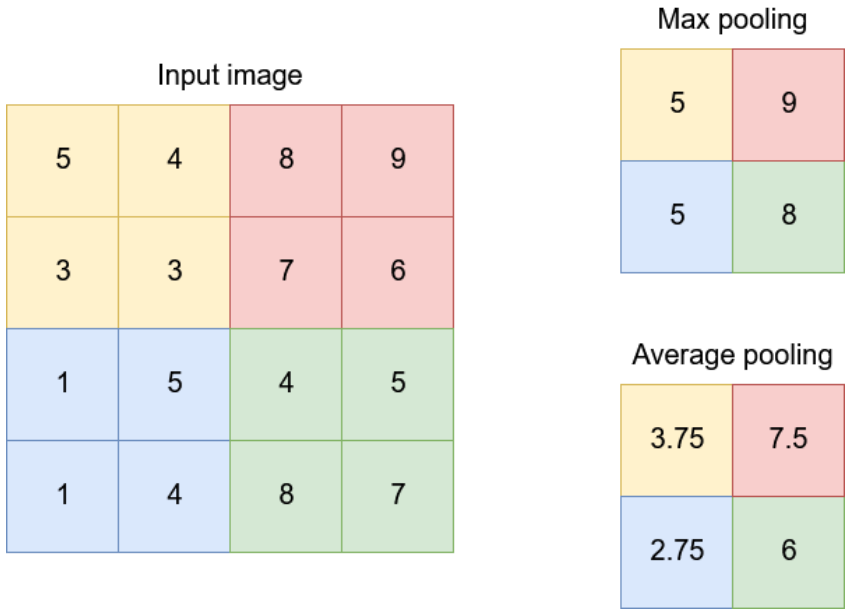


Figure 2.3: The pooling operation using a 2x2 window. The input is on the left side, the result after max pooling is in the top right and the result after average pooling is in the bottom right.

difference in average pooling is that the average value in the window is returned. See Figure 2.3 for an example of the pooling operation.

Max pooling has the advantage of preserving shapes more clearly than average pooling since the most prominent pixel in the window is passed on without loss in value. Average pooling losses the shapes by smoothing out the pixels in the window into a new value.

2.1.6 Transpose convolution

The purpose of a transpose convolution (as used in this study) is to provide a trainable upsampling method.

A transpose convolution has a set of filters just like the convolution operation, but they are used in a different way. Each pixel in the input image makes a projection using the filter (see Equation (2.11)), and that projection is added to some of the pixels of the output image.

$$Projection_{i,j} = W \cdot p_{i,j} \tag{2.11}$$

where $Projection_{i,j}$ is the projection from input pixel $p_{i,j}$ at the location (i, j). W is the filter weights in matrix form. " $\cdot$ " here is an element wise multiplication, so that each element of W is multiplied by $p_{i,j}$.

The $Projection_{i,j}$ is added to the output image by having the center of the projection added to the output pixel $o_{i,j}$, and the other values of the projection added to the corresponding adjacent pixels.

For example, given a input image with size n x n ($n > 1$) and the value 1 in the top left corner, a filter with size 3x3 with values 1. Then the $Projection_{0,0}$ would be a 3x3 matrix with 1s in each cell. When applied to the corresponding output pixel $o_{0,0}$, would be set to the value of the center of $Projection_{0,0}$, which is 1. Adjacent output pixel will also be set to new values. $o_{1,0}$ would be set to the value to the right of the center in $Projection_{0,0}$, which in this case is also 1. The same would happen for $o_{0,1}$ and $o_{1,1}$ and they would both be set to 1. For values outside the bounds of the output image like $(-1, 0)$ no values would be set and they are simply ignored.

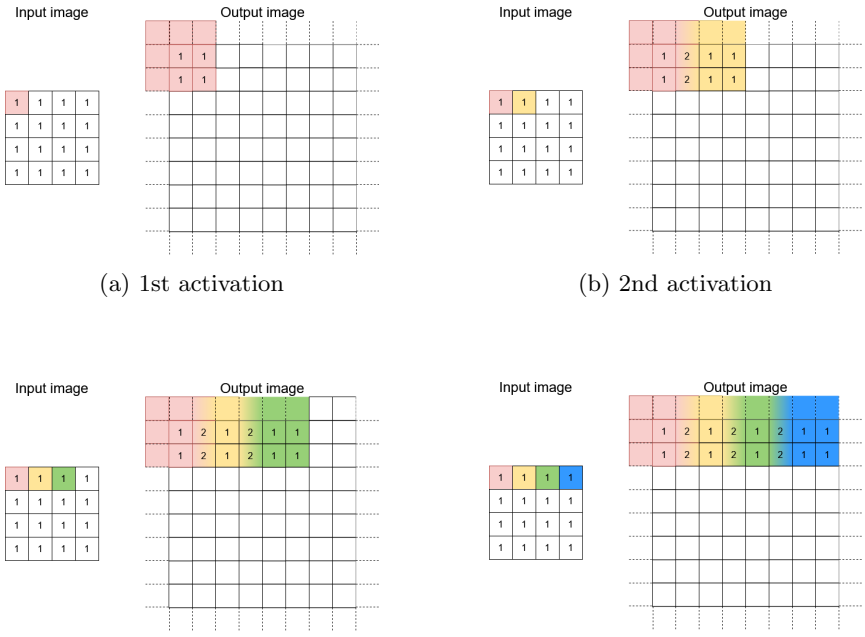
Since each projection from the input image is added to its corresponding output pixel, $o_{0,0}$ would get a contribution from $Projection_{0,1}$, $Projection_{1,0}$ and $Projection_{1,1}$ as well. These contributions are added together to give the final value for $o_{0,0}$. The same applies for the other output pixels.

Using strides of more than 1 gives a transpose convolution the upsampling effect. The filter is moved by the stride number after each activation, so that instead of $projection_{i,j}$ being applied to output pixel $o_{i,j}$ it is now applied to $o_{s \cdot i, s \cdot j}$, where s is the stride ($s \in Z^+$). This distributes the input image more, resulting in a upsampled image as the output. When using padding = "same" output pixels are added in positive x and y directions so that the output image size is $s \cdot i$ x $s \cdot j$. For example, a stride of 2 will take a input image of size n x n and output a image of size 2n x 2n. In Figure 2.4 this is shown for a 4x4 input image that gets upsampled to an 8x8 image by using a stride of 2.

2.1.7 Encoder decoder convolutional neural networks

A common way to use neural networks for image segmentation is to use an encoder decoder architecture. Image segmentation is the task of assigning a class to each of the pixels in the input image. The architecture was for example used in (Xu et al. [2019] and Wang et al. [2019]). This architecture is made up of two main parts, an encoder and a decoder.

The encoder part is a of the network extracts features and downsamples the data. Feature extraction is done by convolutional layers and the downsampling is done by pooling layers. Between the encoder and decoder there is a bottleneck



(a) 1st activation

(b) 2nd activation

(c) 3rd activation

(d) 4th activation

Figure 2.4: Example of an transpose convolution activation on the top row of a 4x4 input image. The stride is 2 and "same" padding is used. The input image is shown on the left side of each subfigure and the output image is shown on the right. The filter used is a 3x3 with each weight set to 1.

layer, this can be either a convolution layer or a feed forward fully connected layer. In the bottleneck we have a small latent representation of the input image.

The decoder part of the network takes the output from the bottleneck layer and interprets the features. The decoder also upsamples the data to the original size, and finally it produces a mapping of each pixel in the input to a class.

The U-net architecture is an encoder decoder architecture that was first published in (Ronneberger et al. [2015]) were it was used for image segmentation for the medical domain. The U-net architecture has an encoder part that is organized into blocks. The output form one block is used as the input to the next block. Each block consists of one or more layers convolutional layers followed by a pooling layer. A block can for example be two convolutional layers followed by a pooling layer. In U-net the blocks are made up of several convolution layers followed by a pooling layer. The output of the last convolution layer is stored for use in the decoder part of the network. This is called a skip connection since it allows some data to skip the rest of the encoder and be passed directly to a part of the decoder.

The decoder is also organized into blocks were each block starts with an upsampling layer (for example a transpose convolution layer with stride 2) followed by one or more convolution layers. A skip connection is used to concatenate the output of the upsampling layer with the stored value from the encoder part with the same image height and width as the output of the upsampling layer in the decoder. The output from a block is used as the input for the next block.

The advantage of using skip connections is that we avoid the loss of high-resolution information that usually happens in the encoder because of the downsampling process. This allows a high-resolution segmentation mapping to be produced by the decoder. Another advantage with using skip connections is that the path from the weights in the encoder to the output layer of the network is shorter. This shorter path allows more direct feedback form the loss calculated in the output layer when training the network, providing a more efficient training process.

2.2 Structured Literature Review Protocol

For the literature review Structured Literature Review (SLR) was used to create a reproducible literature review process as well as having a more focused and reliable way to cover related work. In addition to the studies found using structured literature review some other studies was used. They were included as they are important studies in the field of image segmentation and cited by most of the studies gathered in the structured literature review.

The SLR technique works by dividing concepts about the topic into groups. Each group was filled with keywords that represents this concept. The groups

	Concept 1	Concept 2	Concept 3
Term 1	Neural Network	Segmentation	black and white
Term 2	Artificial neural network	...	B&W
Term 3	Deep learning	...	greyscale
Term 4	...	...	grayscale

Table 2.1: A table showing the keywords associated with concept 1,2 and 3

	Concept 4	Concept 5
Term 1	river	aerial
Term 2	waterway	orthophoto
Term 3	...	orthophotograph

Table 2.2: A table showing the keywords associated with concept 4 and 5

and keywords are shown in Table 2.1 and 2.2. Multiple concepts are entered at a time into the search engine with an OR operator between keywords of the same group/concept and an AND operator between different concepts. Different combinations of concepts were searched for to get a manageable amount of results before the filtering step. The combinations are shown in table 2.3.

The search results are filtered on a set of inclusion criteria. These are shown in table 2.4. All of them must be true for the study to be selected. If a study satisfies all the inclusion criteria, then it will be selected for further reading.

The first search including all concepts on google scholar yielded 395 results. After applying the inclusion criteria to their titles (and in some cases abstracts) only one study satisfied the criteria. The study found in this search was Richard et al. [2018]. This first search was done during the fall project. Because of the low relevance of the papers found future searches were made using Scopus.

Search two was conducted using the Scopus database. The concepts were searched for in the title, abstract and keywords. The search with all concepts included yielded zero results.

Search three used concept 1, 2 and 3 to find studies regarding the technical approach to image segmentation with black-and-white images. The search was further limited to only conference papers and articles. This search yielded 140 results. 35 papers satisfied inclusion criteria 1, 2 and 3.

The papers found that passed the inclusion criteria was then ranked according to the quality criteria (see Table 2.5). Each quality criteria is worth one point and the sum of all the points for a study is used to rank it. All the studies with a score greater or equal to 7.5 were selected for further reading. This resulted in 12 studies shown in Figure 2.5.

Search number	Concepts	Search engine	Number of results
1	All	Google Scholar	395
2	All	Scopus	0
3	1, 2, 3	Scopus	258

Table 2.3: A list of the concept combinations used in the search phase of the literature review.

ID	Description
IC1	The study focuses on image segmentation
IC2	The study uses black-and-white images
IC3	The study presents a solution using neural networks

Table 2.4: The inclusion criteria for the literature review.

ID	Description
QC1	Is there a clear statement of the aim of the research?
QC2	Is the study put in context of other studies and research?
QC3	Are system or algorithmic design decisions justified?
QC4	Is the experimental procedure thoroughly explained and reproducible?
QC5	Is it clearly stated in the study which other algorithms the study's algorithm(s) have been compared with?
QC6	Are the performance metrics used in the study explained and justified?
QC7	Does the test evidence support the findings presented?
QC8	Is the data used in the study openly available?

Table 2.5: Quality criteria used in the structured literature review.

Study	QC1	QC2	QC3	QC4	QC5	QC6	QC7	QC8	Sum
DSL: Automatic liver segmentation with faster R-CNN and deeplab	1	1	1	1	1	1	1	1	8
A coarse-to-fine deep learning framework for optic disc segmentation in fundus images	1	1	1	1	1	1	1	1	8
Semantic segmentation of buildings in remote sensing images based on dense residual learning and channel adaption	1	1	1	1	1	1	1	1	8
Liver Detection Algorithm Based on an Improved Deep Network Combined with Edge Perception	1	1	1	1	1	1	1	1	8
Unsupervised pixel-wise classification for Chaetoceros image segmentation	1	1	1	1	1	1	1	1	8
The Effects of Image Pre- and Post-Processing, Wavelet Decomposition, and Local Binary Patterns on U-Nets for Skin Lesion Segmentation	1	1	1	1	1	1	1	1	8
Real-time traffic sign detection and recognition method based on simplified gabor wavelets and CNNs	1	1	1	1	1	1	1	1	8
A new method for image segmentation based on BP neural network and gravitational search algorithm enhanced by cat chaotic mapping	1	1	1	1	1	0.5	1	1	7.5
Simplified spiking neural network architecture and STDP learning algorithm applied to image classification	1	1	1	1	1	1	0.5	1	7.5
K-Means clustering and neural network for object detecting and identifying abnormality of brain tumor	1	1	1	1	0.5	1	1	1	7.5
Detector of Small Objects with Application to the License Plate Symbols	1	1	1	1	1	1	1	0.5	7.5
Automated segmentation of the corneal endothelium in a large set of 'real-world' specular microscopy images using the U-Net architecture	1	1	1	0.5	1	1	1	1	7.5

Figure 2.5: The studies selected for further reading in search 3.

2.3 Related work

In this section related work for black-and-white image segmentation will be presented. This is relevant to answering **RQ1.1**.

A lot of work has been done on black-and-white image segmentation in the medical domain (Tang et al. [2018]; Xia and Yin [2019]; Wang et al. [2019]; Ross-Howe and Tizhoosh [2018]). Image segmentation of overhead images has also been done on both modern color aerial images (Xu et al. [2019]; Iglovikov and Shvets [2018]) and multi-spectrum satellite images (Murray et al. [2019]). In my work I only found a single article that used machine learning to do image segmentation on black-and-white aerial images (Richard et al. [2018]).

In Richard et al. [2018] a system for image segmentation of black-and-white aerial images from 1955 into 14 classes were presented. The dataset they used had images covering 1200 km^2 of land. They used a dataset from 2015 that was transferred to the 1955 image style using neural network methods to make a dataset to pre-train the model on, then they fine-tuned the CNN model using hand-labeled data from 1955. They got a Mean Intersection over Union (miou) of 58.6% for their best model on the 1955 dataset and a miou of 65% on the 2015 dataset.

In Tang et al. [2018] they used the DeepLab CNN architecture (Chen et al. [2017]) to segment a black-and-white image of a liver from its surroundings. VGG16 (Simonyan and Zisserman [2014]) was used as the encoder, pretrained on the ImageNet (Krizhevsky et al. [2012]) dataset. They achieved state of the art results using this approach.

In Wang et al. [2019] blood vessels from color images were identified and then feed to a CNN U-net model (Ronneberger et al. [2015]). Fundus images where feed separately to a U-net model. This resulted in 2 segmented images that were then combined, and this combined image was feed to a U-net model to produce a final segmentation. Different versions of U-net are used for the different steps. They got an Intersection over Union (IoU) of 89% when using the 3 different U-net models compared to 87% IoU when only using a single U-net model.

In Xu et al. [2019] they used a U-net model to segment buildings from color aerial images on the Inria aerial image labeling benchmark (Maggiori et al. [2017]). They used a U-net model with the ResNeXt50 architecture as the encoder. The model got a miou of 82%.

In Xia and Yin [2019] they used Dense Net 121 (Huang et al. [2016]) to seg-

ment livers in black-and-white images.

Chapter 3

Datasets

This chapter will explain the datasets used and how they were made or collected. Section 3.1 will describe how an initial dataset was made. In Section 3.2 the method for making a new dataset using a model trained on the initial dataset will be explained. Lastly in Section 3.3 three test sets made to test model performances on different rivers or the same rivers at different times will be described.

3.1 Initial dataset

This section will describe how the initial dataset was made. First the aerial images were downloaded as described in Section 3.1.1. The images were manually annotated (see Section 3.1.2) and then divided into smaller images suitable for machine learning (see Section 3.1.3). The small images were filtered so that only the images of high quality from a machine learning perspective were kept (see Section 3.1.4). Finally, the images were split into disjoint training, test and validation datasets. Figure 3.1 gives an overview over the process of making the initial dataset.

3.1.1 Black-and-white aerial images

The image data available for Norwegian rivers before the 2000s were black-and-white analog aerial images. They were scanned from their analog format into digital images and made available at Kartverket [2019b]. Retrieval of the images were done manually by making a bounding box around the rivers that we wanted to use for the dataset. The web service norgeibilder (Kartverket [2019b]) provided images of the size 6000x8000 pixels that overlapped with the bounding box. After this step we had a dataset of raw aerial images of the Gaula and Lærdal rivers in

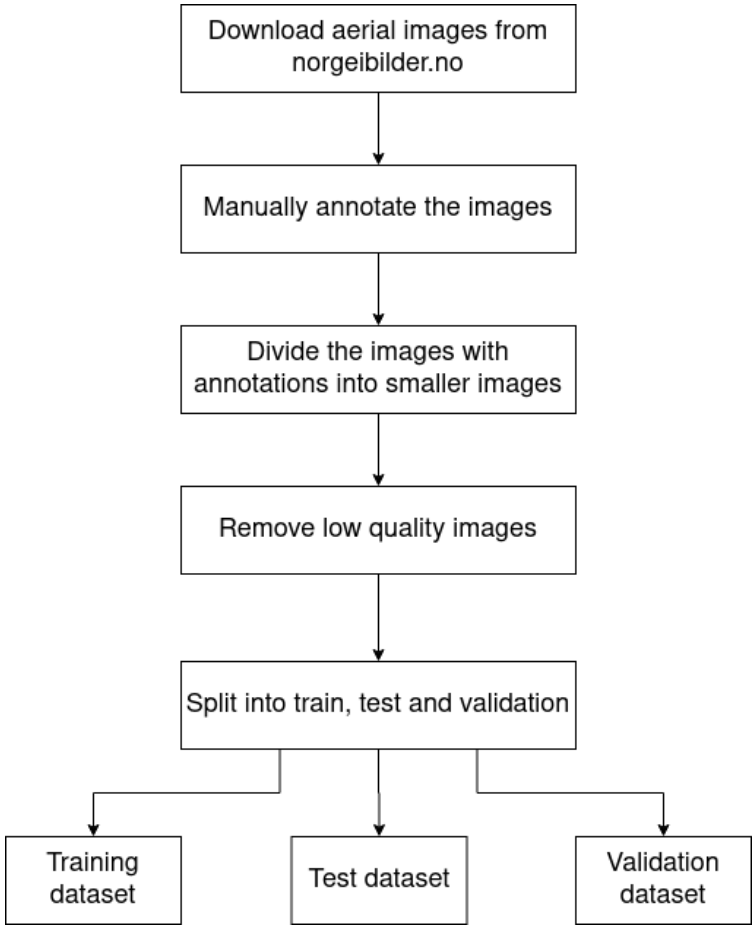


Figure 3.1: An overview over the process to make the initial dataset.

River	Year	Total number of images	Number of images with annotations
Gaula	1963	206	66
Lærdal	1976	50	38

Table 3.1: The size of the dataset for each river. (An image here means a 6000x8000 pixel image.)

Norway. See Table 3.1 for the size of the dataset for each river. The resolution of the images were 0.2 meters.

3.1.2 Annotations

A subset of the images was manually annotated by a mix of domain and non-domain experts. This was done by giving each of the experts a set of classes to identify and letting the experts segment and annotate those areas on the aerial images. The experts marked areas within 200 meters of the river. The annotations were saved as shapefiles, one file for each class for each river. The shapefiles were georeferenced and used the ETRS:25833 reference system. The Geographical Information System QGIS (QGIS Development Team [2020]) was used for making the annotations.

The classes annotated were water, gravel, vegetation, farmland and human-constructions. See Table 1.1 for the class definitions.

3.1.3 Dividing the images

For the images to be usable for machine learning they had to be divided into smaller images of size 512x512 pixels. This is done by dividing a 6000x8000 image into a grid of 512x512 images as shown in Figure 3.3.

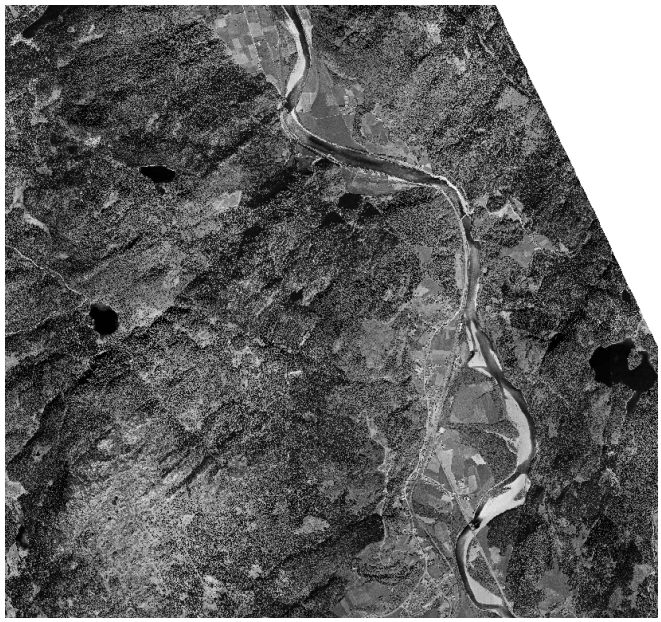
Each 6000x8000 image gives 192 512x512 images. In total this gives $192 \cdot 104 = 19968$ small images before filtering.

3.1.4 Image filtering

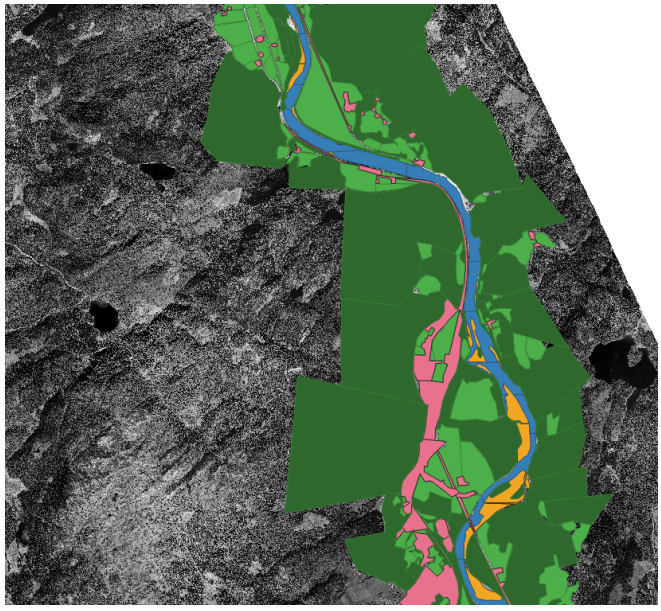
Of the 19968 images only 1694 remained after filtering. The filtering process is as follows:

1. Remove images with more than 10% of the unknown class.
2. Remove images with only a single class

The filtering step is necessary to make sure only images with a good labeling are used. This is done by the 1st filtering step. The 2nd filtering step is done



(a) Aerial image



(b) Manual annotations

Figure 3.2: A 5 km tall image of Gaula with and without manual labels. The labels for image b is blue:water, black:unknown, light green: farmland, dark green: vegetation, pink: human-constructions, brown: gravel

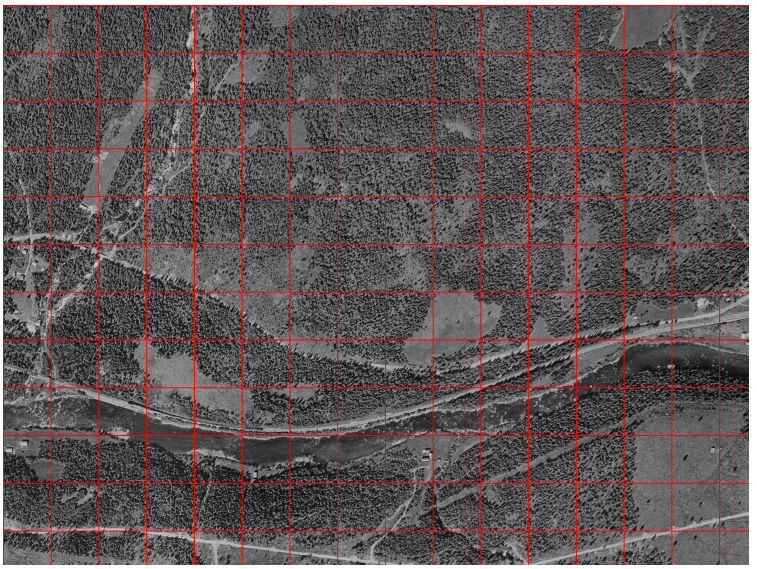


Figure 3.3: A 6000x8000 pixel image from Gaula 1963. The red squares show how the 512x512 pixel images are divided.

to reduce class imbalance and to have images that needed segmentation instead of just classification. 15109 of the images were 100% of the unknown class and they were removed. 935 of the images were of only a single class (other than the unknown class) and they were also removed. 2230 of the remaining images had more than 10% of the unknown class and were removed. This left 1694 images with good labels remaining as the dataset used for machine learning.

3.2 Using the model to extend the dataset

To increase the size of the training dataset a new dataset was made after the initial dataset. This new dataset was made by using a model trained on the initial dataset to make predictions on images from the Surna river from 1963. This river was not part of the initial dataset. The predictions were manually corrected in by a non-domain expert. This new dataset had 75 big images and when divided into 512x512 pixel images and filtered the same way as the initial dataset this dataset had 4613 512x512 images from Surna. The total size of the extended dataset was then 6307 images, from Surna 1963, Gaula 1963 and Lærdal 1976.

The new images from Surna were added to the training and validation sets and

Configuration	val miou	val acc	val loss
no image aug, freeze all	0.5672	0.8405	0.4951
no image aug, freeze first	0.5576	0.8321	0.5133
image aug, freeze all	0.6035	0.8535	0.4537
image aug, freeze first	**0.6178**	**0.8614**	**0.4332**

Table 3.2: The results of experiment 1. val miou: Validation mean intersection over union. val acc: validation accuracy. val loss: The validation loss, were the loss function was sparse categorical cross entropy.

used to train a new model. 80 % was used for training and 20 % for validation. The test set remained unchanged and only included images from Gaula and Lærdal. The images of the extended dataset covered 66 km^2.

3.2.1 Model for extending the dataset

To find the best model four different configurations of parameters was tried. Two parameters were changed, the "frozen" parameter that determined how many of the blocks in the model was untrainable. The second parameter was image augmentation. Each parameter was tried with two values, so in total there were $2^2 = 4$ configurations. For configurations with the frozen parameter set to "all", all the weights of the encoder were frozen, meaning the weights did not change during training. For configurations with the frozen parameter set to "first", the first 4 convolution blocks of the encoder were frozen, and the last convolution block was trainable. For all configurations the decoder was trainable. The model architecture is described in Section 4.2.

Early stopping was used with a patience of 10 epochs and it monitored the validation loss. Model checkpointing was used to save the best model according to validation loss during training. All results are from the model with the lowest validation loss during training of each configuration.

For configurations with data augmentations the training dataset was augmented by adding rotated, horizontal- and vertical mirrored images to the training set. This data augmentation resulted in 11 additional images for each original image in the training set. Image augmentation was never used on the validation or test set to keep them as true to the real data as possible.

The configurations and the performance of the model corresponding to those configurations are shown in Table 3.2. The best configuration according to all the measured metrics was using image augmentation and freezing only the first layers of the encoder. This is the model that was used for extending the dataset.

3.3 Test sets

To get an objective evaluation of a model, the model was tested on test sets from different rivers and at different times. Four different test set were used, and they will be described in this section.

A test set from Nea 1962 was used to evaluate the performance of the model on a river that were not included in the training or validation sets, to see how the model performs on an unseen river from the same time period.

A test set from Gaula 1998 was used to evaluate the performance of the model on a river that was included in the training data but for a different time period.

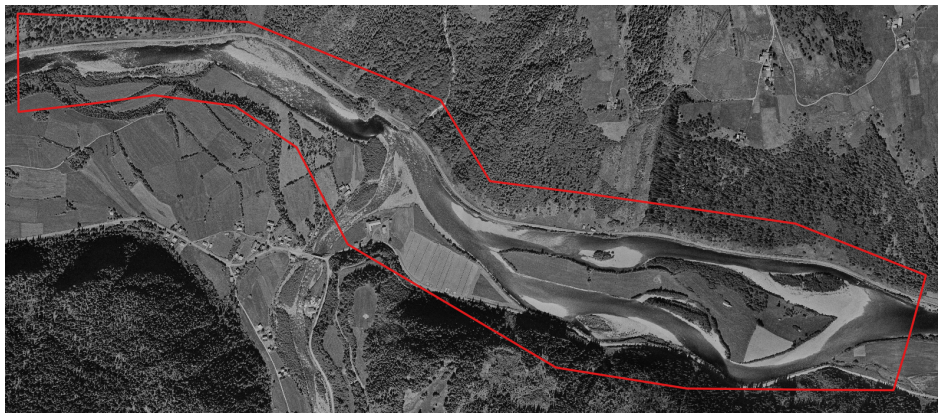
A test set from Gaula 1963 from a part of the river that was not included in the training or validation sets. This was done to evaluate the performance for the same river and the same time period as the training data. This will be especially useful when comparing the performance amongst the different test sets. The comparison with the different test sets can reveal if the model is able to generalize to different rivers and time periods.

The Nea 1962, Gaula 1963 and Gaula 1998 test sets were made by using the best model from Section 5.3 to make segmentations and manually correcting the errors in the segmentations. The use of the model was done to make the test sets faster than if the entire images had to be manually segmented. However, this can introduce a bias in the data since the edges will match with the predictions while in a manually made test set the edges would be strait lines. Also, in some cases where it was difficult for an expert to make a segmentation, the segmentation from the model might be accepted as correct while in a manually made dataset it might have been segmented as another class under uncertainty.

Lastly a test set made up of images randomly selected when making the test, validation and training sets was used. This dataset included images from Gaula 1963 and Lærdal 1976 that are right next to the images in the train and validation sets. The performance of the model on this set might be misleading because the images are similar to the training data. The images were manually labeled using the process described in Sub section 3.1.2. When referring to test sets this test set it is called the initial test set.

The number of images for each of the test sets are shown in Table 3.4 and the class distribution for the ground truth (corrected predictions) are shown in in Table 3.3. In total the test sets covered 9.7 km^2.

In Figure 3.4 the areas for the test sets are shown.



(a) The test area for Gaula 1963

(b) Test area for Gaula 1998

(c) Test area for Nea 1962

Figure 3.4: Test areas for Gaula 1998, Gaula 1963 and Nea 1962. The test area is defined by the red bounding polygon.

	W	G	V	F	H
Nea 1962	18.05%	2.18%	23.99%	50.93%	4.85%
Gaula 1998	27.89%	2.47%	22.05%	28.02%	19.57%
Gaula 1963	17.11%	8.40%	36.62%	15.25%	3.21%

Table 3.3: The class distribution for the ground truth (corrected predictions) for the test sets.

	Nea 1962	Gaula 1998	Gaula 1963	Initial test set
Images*	385	112	90	340

Table 3.4: Number of images in each test set. *The test set areas are defined using a polygon, so the number of images shown here are calculated by dividing the total number of pixels in the test area by the number of pixels in an image (512^2)

Chapter 4

Architecture and Model

This chapter will cover the system architecture for making predictions and training models (see Section 4.1). The goal of the system is to automatically segment black-and-white aerial images into one of the classes in Table 1.1. The CNN model used will be explained in Section 4.2.

4.1 System architecture

For the images to be used by the CNN model they first have to be pre-processed. This applies to both training the model and to making predictions. The pre-processing will be explained in detail in Section 4.1.1. The training process will be described in Section 4.1.2 and how the predictions were made will be described in Section 4.1.3. The overall system process flow for the system is shown in Figure 4.1. The code for the system can be found on `https://github.com/arildsd/river-segmentation`.

4.1.1 Pre-processing

The pre-processing step takes datasets of black-and-white images, where each image is 512x512 pixels (102.4m x 102.4m) as input. The process of making the input datasets are covered in Section 3.1. The datasets are made up of three disjoint datasets, a training dataset, a test dataset and a validation dataset. An overview of the pre-processing steps are shown in Figure 4.2.

Each pixel value in the images were an integer between 0 and 255. Images were normalized by dividing each pixel by 255, so that the new values were between 0 and 1.

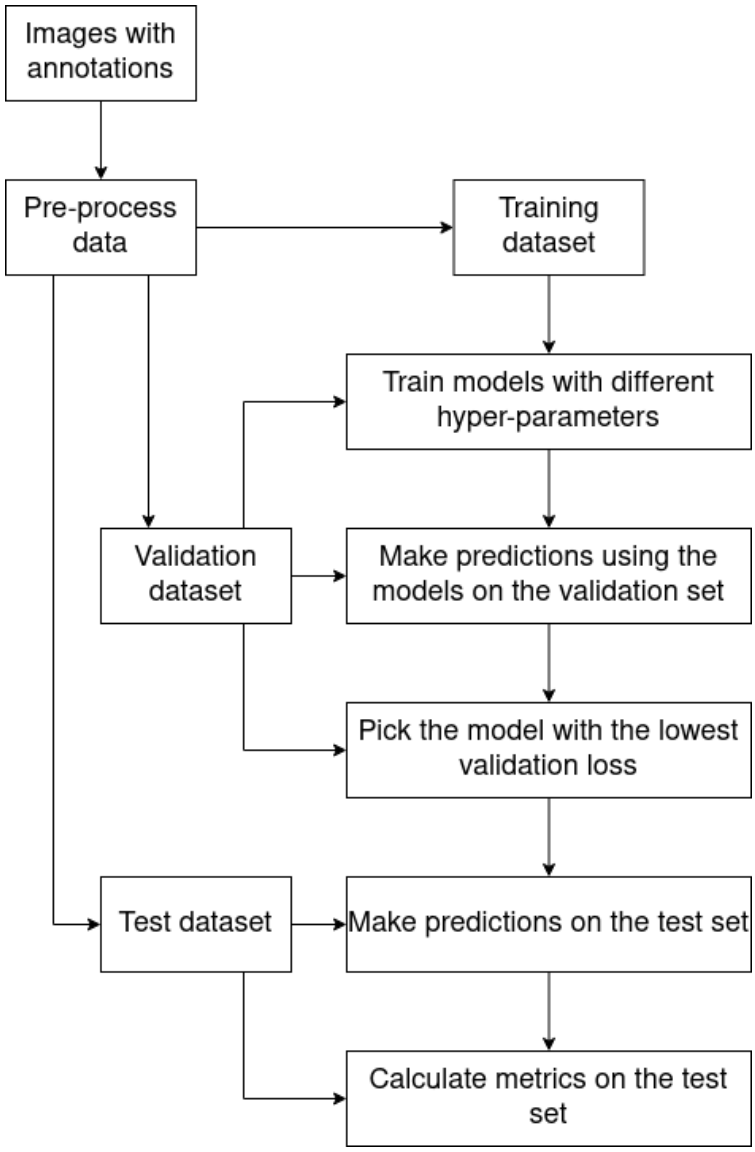


Figure 4.1: The overall process flow for the system.

Then the pixels with the unknown class is replaced with the class of the pixel that is closest that has a non-unknown class.

The next pre-processing step was to duplicate the black-and-white channel to get images with 3 channels. Each channel was a copy of the original black-and-white channel and this step was done because the pretrained part of the model was trained on a RGB dataset and expects an input with 3 channels. This allows us to take advantage of a model trained on over a million images (see Section 4.2 for more detail on the pretrained model). A drawback of this approach is that the space used to store the dataset in memory is 3 times the original size.

After the duplication step the validation and test sets was ready to be predicted on by the model. The training dataset was augmented by adding rotated and flipped versions of the original images to the dataset. Rotation with 90°, 180°and 270°was applied. Horizontal and vertical flipping was used, and rotations were applied to the flipped images as well. This resulted in 11 additional images for each original image in the training dataset.

Image augmentation was used to help avoid the model memorizing the specific details in the training data instead of learning how to segment rivers in general. For example if the training data had a river going from the top (north) to the bottom (south) with vegetation on the left (west) side of the river, the model might learn that all rivers must be to the right of vegetation. With rotated and flipped images the training data would also contain images were the vegetation were on the right, on the top and at the bottom of the river. This would remove the problem of river always being on the right of vegetation in the dataset.

After the training data was augmented all the data was ready for use by the model.

4.1.2 Training the model

The model is trained by letting it make predictions, calculate the loss using the ground truth and then correcting the weights for the model (see Section 2.1.2 for more detail). An epoch is when all the training images has been used for making predictions. After each epoch the model made predictions on the validation set and calculated the validation loss.

Early stopping was used to stop the training process if the validation loss did not decrease in 10 epochs. This was done to prevent overfitting, that is to prevent the model to only learn the specifics of the training dataset and not the general properties of rivers images. Generally, when overfitting starts to happen the training loss will continue to decrease, but the validation loss will start to increase.

Model checkpointing was used to save the best model according to the validation loss.

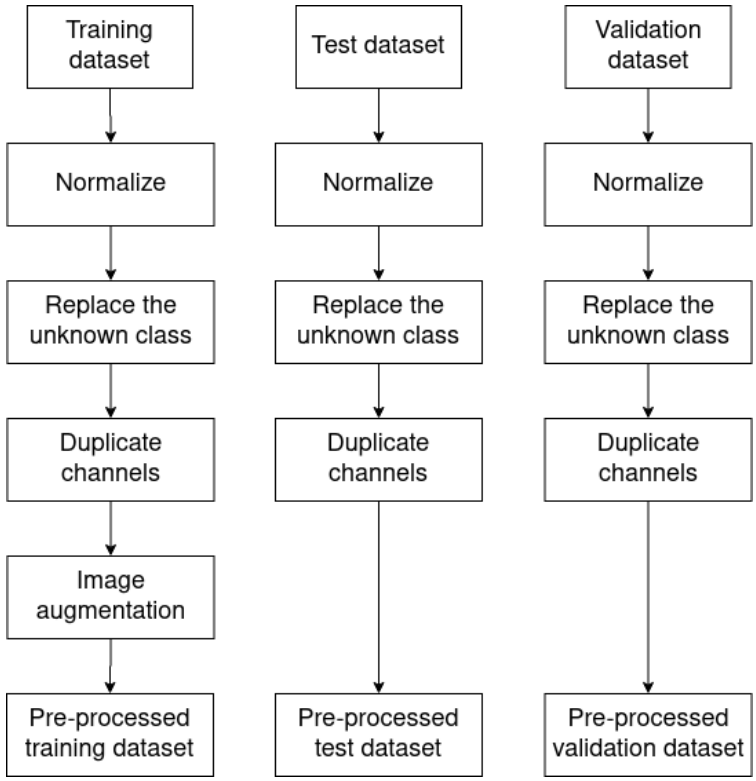


Figure 4.2: The pre-processing steps.

4.1.3 Making predictions

To make a prediction using a trained model requires the input data to be pre-processed the same way as when pre-processing the training data. After the input data is pre-processed the model can make predictions on it.

4.2 Model

The model used for making predictions was a deep CNN U-net model. It had an encoder part that extracts features from the input image and downsamples the images. It also had a decoder part decodes the downsampled feature maps from the encoder part. The decoder part also upsamples the feature maps to create an output segmentation mask with the same size as the input image. For more details on this type of model architecture see Section 2.1.7.

When an image is given to the model the image is first processed at the first convolution layer in the first block in the encoder part. The results from this layer is passed on to the 2nd layer and more features are extracted. This process of extracting features and passing them on to the next layer continues for all the layers in the CNN. The output of the last convolution layer in each block in the encoder is saved for use in the corresponding block in the decoder layer. The decoder layers interpret the feature maps from the previous layer. The architecture of the model is shown in Figure 4.3.

Figure 4.3: The model architecture. The number under each box gives the number of channels for the output feature map. The tilted number at the end of each block is the image size for that block. Light brown boxes are convolution layers, red boxes are max pooling layers. Light blue boxes are transpose convolution layers and dark blue boxes are concatenation between the skip connection from the encoder and the previous transpose convolution. The last layer is a softmax layer.

4.3 Encoder

The VGG16 model (Simonyan and Zisserman [2014]) is used as a pretrained encoder for this model. VGG16 had 13 trainable convolution layers as well as 5 pooling layers. The layers are arranged as 5 convolution blocks, where each block has 2-3 convolution layers followed by a pooling layer. VGG16 was used on the ImageNet(Krizhevsky et al. [2012]) dataset for image classification. When used on the ImageNet dataset it had a fully connected classification part after the encoding part. This part classified the input image as one of the 1000 classes in the ImageNet dataset. For this model the classification part of VGG16 was removed and only the convolution/pooling part was used.

The advantage of using a pretrained encoder like VGG16 is that many of the basic features can be extracted using the first blocks of the pretrained network. These basic features can for example be edges and simple shapes. These features will be useful for almost all image classification/segmentation problems and will therefore be learned for most datasets. Since the encoder already know how to extract the basic features the training process can focus on learning the specific features needed for the segmenting river ecology.

To determine how many of the blocks of the that encoder should be trained a few different models were trained with a different number of blocks marked as untrainable for each model. The best model according to the validation set was chosen. This ensures that only the blocks that extract useful features for the river ecology domain are used without further training.

4.4 Decoder

The decoder part of the network had 5 upsample blocks and 1 convolution block directly after the encoder (see Figure 4.3). The upsample blocks had a transpose convolution layer with a stride set to 2, this layer would upsample the image/feature map. Following the transpose convolution layer there were two convolution layers. The convolution layers extracted features from the previous layer without changing the shape of the input image/feature map.

After each transpose convolution layer, the feature maps from the corresponding downsample block are concatenated to the channels of the output from transpose convolution. This is called a skip connection since the feature map that is concatenated skips part of the downsample-upsample process.

The last layer in the decoder was a softmax layer that assigns a pseudo probability for each class for each pixel. The class with the highest value of the pseudo probability for the pixel is the classification for that pixel.

Chapter 5

Experiments and Results

This chapter will cover the experiments that were done and the results from those experiments. The goal of the experiments is to determine the best model for image segmentation of rivers using black-and-white aerial images.

5.1 Experimental Plan

First a model was trained on the extended dataset (see Section 5.3) and tested on a some test sets. The model architecture could then be evaluated in relation to **RQ1.2** "What is the best model architecture for the segmentation?" in Section 6.2.

In Section 5.4 a pre-processing step that adjusts for light intensity differences was tried, and this experiment is also relevant to **RQ1.2**.

5.2 Experimental Setup

The experiments were performed on the NTNU IDUN computing cluster (Själander et al. [2019]). The cluster had more than 70 nodes and 90 GPGPUs. Each node contained two Intel Xeon cores, at least 128 GB of main memory, and was connected to an Infiniband network. Half of the nodes were equipped with two or more Nvidia Tesla P100 or V100 GPGPUs. All the models were implemented in Tensorflow 2.0 (Abadi et al. [2015]) for python 3.7 (Van Rossum and Drake [2009]). Numpy 1.8 (Oliphant [2006]) and Gdal 3.0 (GDAL/OGR contributors [2020]) was used for data pre-processing.

The models is described in Chapter 4 and the dataset is described in Chapter 3.

The models are measured on a set of metrics. These are Mean Intersection over Union (miou), Accuracy, Precision, Recall and loss. The metrics are defined in Equations (5.1) to (5.6). miou is used to measure how well the segments from the model overlaps with the ground truth. Accuracy is included as it is an easily interpretable metric, but since the dataset is unbalanced it can be misleading so it should be combined with other metrics. Recall for a class c is how many of the pixels for c in the ground truth did the model actually segment as c. Precision for a class c is how many pixels the model segmented as c was actually c in the ground truth.

Notation: C is the set of classes, $pred_c$ is the predicted true pixels for class c, $true_c$ is the true pixels for the class c and $\overline{true_c}$ is the pixels that is not class c.

$$\text{miou} = \frac{1}{|C|} \sum_{c \in C} \frac{|pred_c \cap true_c|}{|pred_c \cup true_c|} \tag{5.1}$$

$$\text{Accuracy} = \frac{\text{correctly segmented pixels}}{\text{total number of pixels}} \tag{5.2}$$

$$\text{Precision}_c = \frac{|pred_c \cap true_c|}{|pred_c \cap true_c| + |pred_c \cap \overline{true_c}|} \tag{5.3}$$

$$\text{Average Precision} = \frac{1}{|C|} \sum_{c \in C} \text{Precision}_c \tag{5.4}$$

$$\text{Recall}_c = \frac{|pred_c \cap true_c|}{|true_c|} \tag{5.5}$$

$$\text{Average Recall} = \frac{1}{|C|} \sum_{c \in C} \text{Recall}_c \tag{5.6}$$

Tables and figures in this chapter uses the abbreviations and color coding defined in Table 5.1.

Class name	Abbreviation	Color
Water	W	Blue
Gravel	G	Brown
Vegetation	V	Dark green
Farmland	F	Light green/ yellow
Human-constructions	H	Red
Unknown	U	Purple

Table 5.1: Abbreviations and colors for the different classes.

5.3 Experiment 1: Training models with an extended dataset

In this experiment six different configurations for the model was tried. The dataset consisted of images from Gaula 1963, Surna 1963 and Lærdal 1976. The models trained in this experiment were trained on the extended dataset. See Section 3.2 for the details on how the extended dataset was made.

The same model architecture as when making the extended dataset was used, and the same pre-processing and image augmentation methods were used. As images augmentation yielded the best results in when extending the dataset this was used for all the configurations in this experiment. See Chapter 4 for the system and model architecture.

Early stopping was used with a patience of 10 epochs and monitoring the validation loss. Model checkpointing was used to save the model with the lowest validation loss. The model with the highest mean intersection over union on the validation set was chosen for further testing.

In Table 5.2 the validation scores for the different configurations are shown. The "Freeze 2" configuration was the best one according to both validation miou and accuracy, and the "Freeze 1" configuration was best according to the validation loss. In the freeze 2 configuration the first two convolution blocks in the encoder part of the model are untrainable. In the freeze 1 configuration the first convolution block of the encoder was untrainable. Since the freeze 2 configuration had more of the weights frozen overfitting to the validation set was less of a problem since the learnable weights were fewer. This combined with the best scores on two of the metrics made this configuration the best one and the one that was used on the test sets.

The model trained with the freeze 2 configuration was used to predict on the test sets. The results for each test set are shown in Table 5.3. The confusion matrices are shown in Figures 5.1 and 5.2. The confusion matrices show the

Configuration	val miou	val acc	val loss
freeze all	0.7006	0.8919	0.3102
freeze first	0.7414	0.9064	0.2765
freeze 3	0.7463	0.9078	0.2726
freeze 2	**0.7627**	**0.9107**	0.2653
freeze 1	0.751	0.9076	**0.2625**
freeze none	0.7513	0.9073	0.2661

Table 5.2: The validation scores from experiment 1.

relationship between the predicted class and the true class for each class. A perfect model would give a confusion matrix with 100% along the diagonal and 0% for every other cell in the matrix. The unnormalized confusion matrices are included in Section 6.5 in the appendix.

The images from Gaula 1998 were brighter than the images in the training data. This caused some issues for the model as gravel is over predicted as shown in Figure 5.3, where some farmland is wrongly segmented as gravel.

The model correctly segmented the water as water with almost no errors (see Figure 5.4). This fits with the high recall for water, 89.8%, in the test set. (Shown in Figure 5.1c).

For Gaula 1998 human constructions were mostly correctly segmented with a recall of 85.7% and a precision of 89.22%. This can also be seen in Figure 5.5.

The model predicted 17.88% of the vegetation as water. This is an error unique to the Gaula 1998 images, and a possible explanation for this is the higher light level in the Gaula 1998 images. The high light level makes the usually dark vegetation look more like the slightly lighter water texture. An example of vegetation predicted as water is shown in Figure 5.6.

In Figure 5.7 a representative example of the predictions on Gaula 1963 are shown. Some sections of the water was wrongly predicted as farmland as shown in Figure 5.8.

The images from Gaula 1947 was the oldest images used in this project, and the results on these images will be useful in evaluating how good the model is at old images. The images were darker than the images in the training data, and this probably resulted in a worse performance.

The model was able to correctly segment vegetation, as shown in Figure 5.9, but struggled with predicting gravel correctly. Gravel was often predicted as water (see Figure 5.10) Farmland was in many areas not recognized correctly and it was wrongly segmented as water and gravel as seen in Figure 5.11.

Test set	miou	Accuracy	Average Precision	Average Recall
Nea 1962	77.49%	93.40%	91.61%	83.76%
Gaula 1998	53.81%	73.17%	69.77%	72.07%
Gaula 1963	71.30%	87.60%	81.24%	84.72%

Table 5.3: Test scores for each test set using the best model from experiment 1. miou: Mean intersection over union.

In Figure 5.13 from Nea 1962 we can see that the model has segmented some vegetation as water. Here the vegetation was difficult to see because of shadows and the model did not have any information about the surrounding area, only the part inside the red rectangle.

Nea 1962		Predicted class				
		W	G	V	F	H
	W	95.36%	0.14%	1.83%	2.39%	0.28%
	G	22.68%	53.15%	8.04%	10.07%	6.05%
	V	3.14%	0.11%	90.51%	4.59%	1.64%
	F	1.78%	0.03%	1.12%	96.79%	0.27%
True class	H	0.09%	0.00%	2.80%	14.15%	82.96%

(a) Nea 1962

Gaula 1963		Predicted class				
		W	G	V	F	H
	W	91.31%	0.38%	1.38%	6.93%	0.00%
	G	7.84%	76.73%	6.72%	6.10%	2.60%
	V	2.10%	1.75%	88.96%	2.30%	4.90%
	F	0.60%	2.49%	8.37%	88.12%	0.42%
True class	H	2.85%	2.19%	7.34%	9.15%	78.47%

(b) Gaula 1963

Gaula 1998		Predicted class				
		W	G	V	F	H
	W	89.20%	3.90%	2.12%	4.72%	0.06%
	G	6.99%	67.18%	2.30%	16.68%	6.86%
	V	17.88%	9.17%	54.71%	14.64%	3.60%
	F	0.05%	31.58%	1.10%	63.53%	3.74%
True class	H	0.01%	3.03%	2.26%	9.00%	85.70%

(c) Gaula 1998

Figure 5.1: Row normalized confusion matrix for the model predictions on the test sets. The values along the diagonal are the recall for that class. See Table 5.1 for the link between classes and abbreviations.

Nea 1962		Predicted class				
		W	G	V	F	H
	W	88.84%	1.99%	1.44%	0.83%	1.08%
	G	2.56%	94.34%	0.77%	0.43%	2.79%
	V	3.89%	2.24%	94.71%	2.13%	8.32%
	F	4.69%	1.43%	2.49%	95.29%	2.92%
True class	H	0.02%	0.00%	0.59%	1.33%	84.88%

(a) Nea 1962

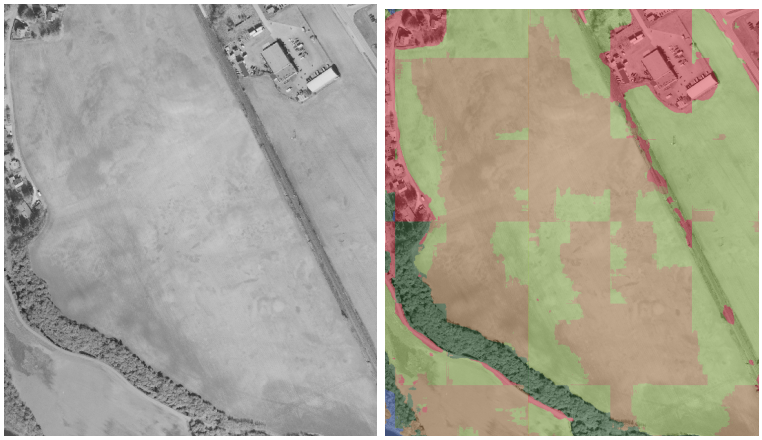
Gaula 1963		Predicted class				
		W	G	V	F	H
	W	90.65%	0.84%	0.68%	7.29%	0.00%
	G	3.83%	84.81%	1.62%	3.15%	4.76%
	V	4.46%	8.42%	93.37%	5.17%	39.04%
	F	0.53%	5.00%	3.66%	82.58%	1.40%
True class	H	0.53%	0.92%	0.68%	1.80%	54.80%

(b) Gaula 1963

Gaula 1998		Predicted class				
		W	G	V	F	H
	W	85.76%	7.66%	4.39%	5.37%	0.09%
	G	0.59%	11.67%	0.42%	1.68%	0.90%
	V	13.59%	14.23%	89.61%	13.17%	4.22%
	F	0.04%	62.27%	2.29%	72.60%	5.57%
True class	H	0.01%	4.18%	3.28%	7.18%	89.22%

(c) Gaula 1998

Figure 5.2: Column normalized confusion matrix for the model predictions on the test sets. The values along the diagonal are the precision for that class. See Table 5.1 for the link between classes and abbreviations.



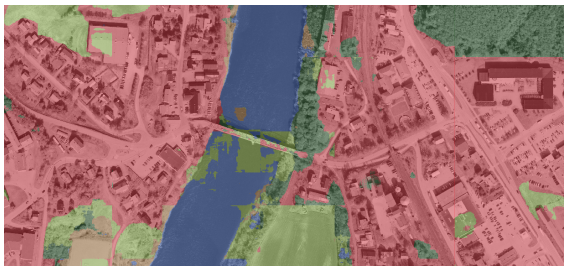
(a) Image (b) Image with predictions

Figure 5.3: Example from Gaula 1998. Farmland predicted as gravel.

(a) Image (b) Image with predictions

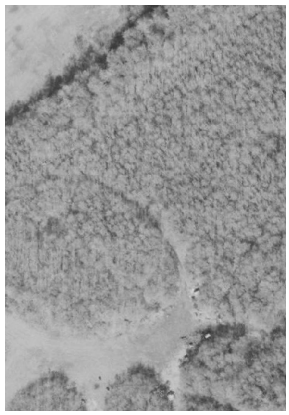
Figure 5.4: Example from Gaula 1998. Water predictions.

(a) Image

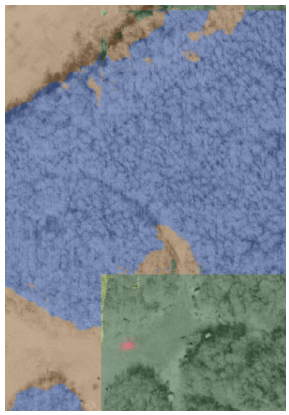


(b) Image with predictions

Figure 5.5: Example from Gaula 1998. Constructions with bridge.

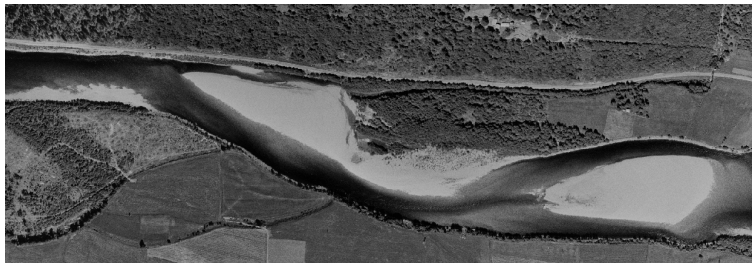


(a) Image

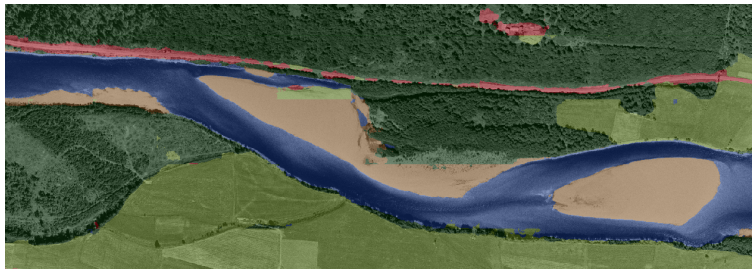


(b) Image with predictions

Figure 5.6: Example from Gaula 1998. Vegetation segmented as water.

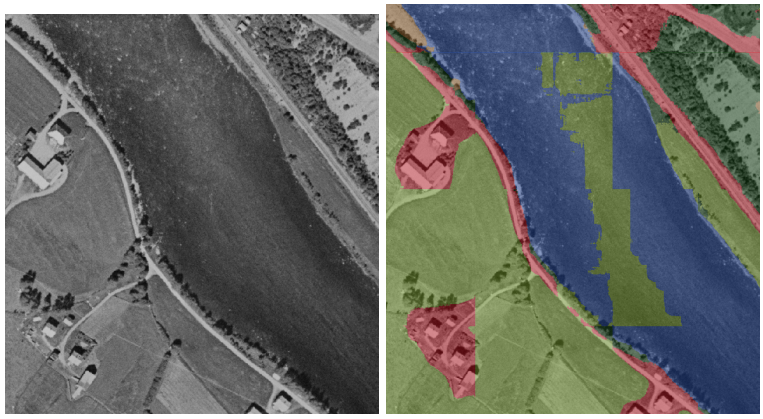


(a) Image



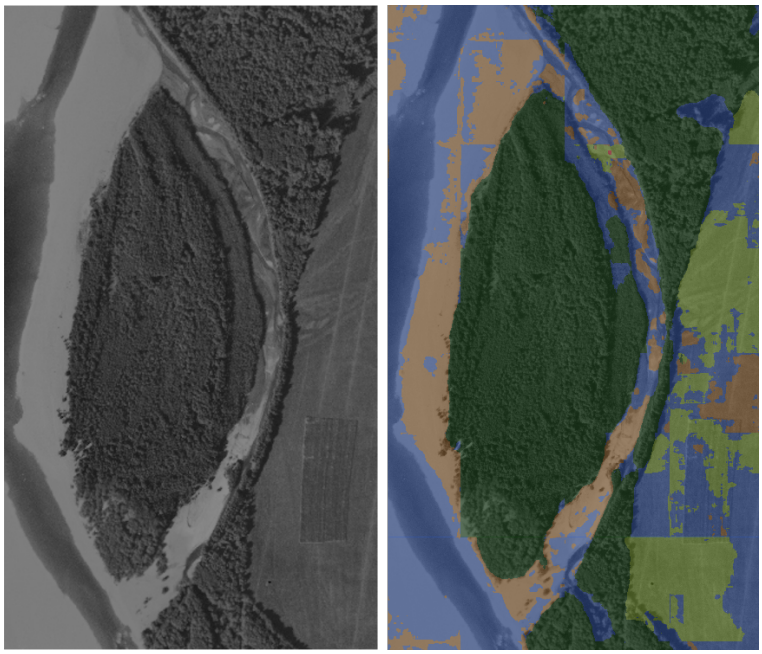
(b) Image with predictions

Figure 5.7: Generic example from Gaula 1963.



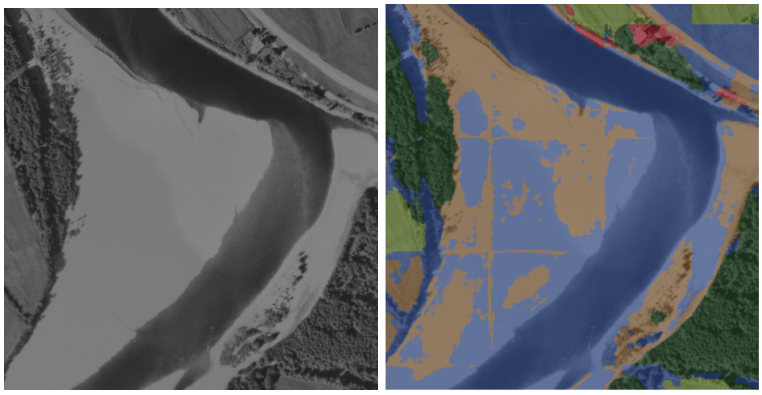
(a) Image (b) Image with predictions

Figure 5.8: Example from Gaula 1963. Water segmented as farmland.



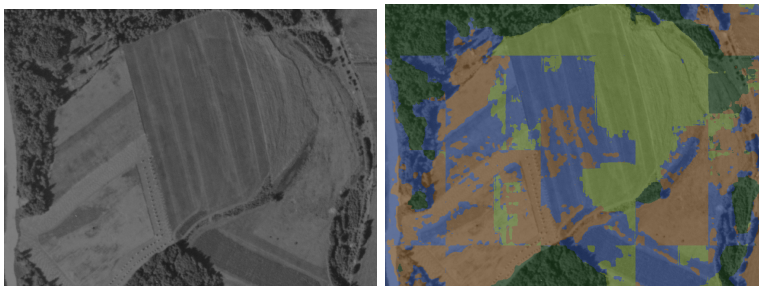
(a) Image (b) Image with predictions

Figure 5.9: Example from Gaula 1947. Correctly segmented vegetation.



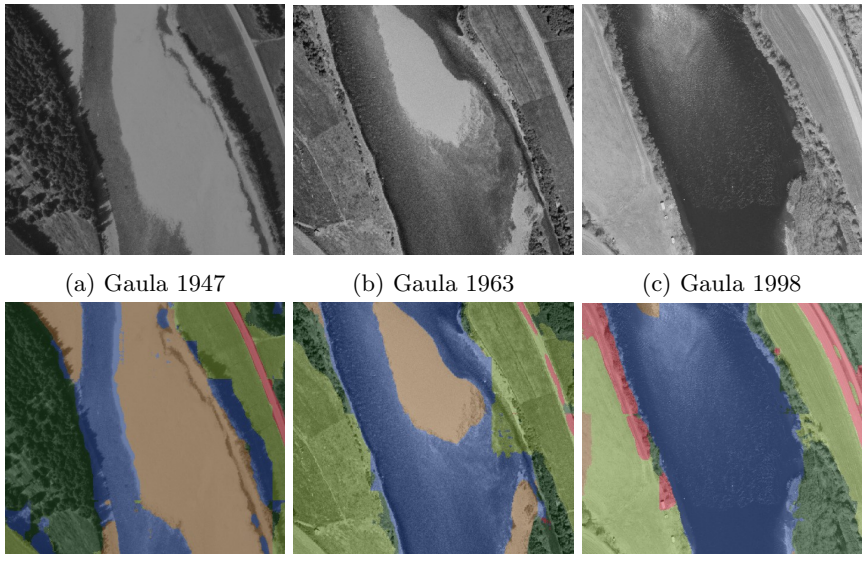
(a) Image (b) Image with predictions

Figure 5.10: Example from Gaula 1947. Gravel predicted as water.



(a) Image (b) Image with predictions

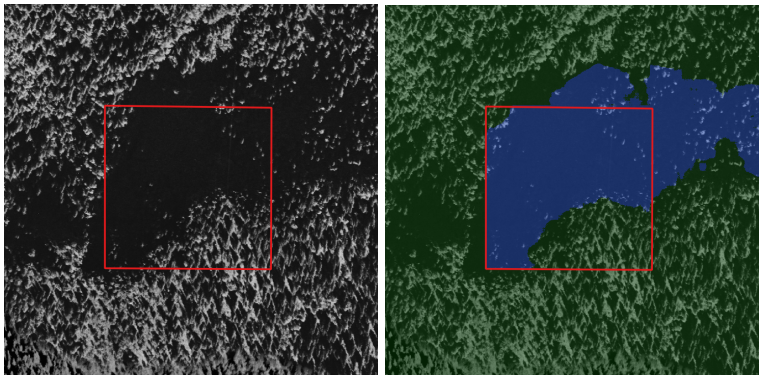
Figure 5.11: Example from Gaula 1947. Wrongly segmented farmland.



(a) Gaula 1947 (b) Gaula 1963 (c) Gaula 1998

(d) Gaula 1947 predictions (e) Gaula 1963 predictions (f) Gaula 1998 predictions

Figure 5.12: The same geographical area of the Gaula river with predictions for 1947, 1963 and 1998. The 1947 and 1998 predictions are intensity adjusted.



(a) Image (b) Image with predictions

Figure 5.13: Example from Nea 1962. Vegetation segmented as water. The red rectangle shows the limits of the image that the model had as input.

5.4 Experiment 2: Light intensity adjustment

A problem encountered when predicting on new datasets was that the light intensity was different than the light intensity in the training data. This was especially problematic for the Gaula 1998 test set, as it had a mean light intensity of 133 and the training data had a mean light intensity of 77.

In this experiment the light intensity was adjusted for to make the mean light intensity of the datasets equal to the mean light intensity of the training set. This was done as a pre-processing step when making predictions by adding the difference in light intensity means ($\Delta\mu$) to each pixel in the dataset (see Equation (5.7)). The best model from experiment 1 was then used to make predictions on the dataset.

$$\Delta\mu = \mu_{train} - \mu_{dataset} \tag{5.7}$$

where μ_{train} is the mean light intensity of the training data and $\mu_{dataset}$ is the mean light intensity of the dataset that is predicted on.

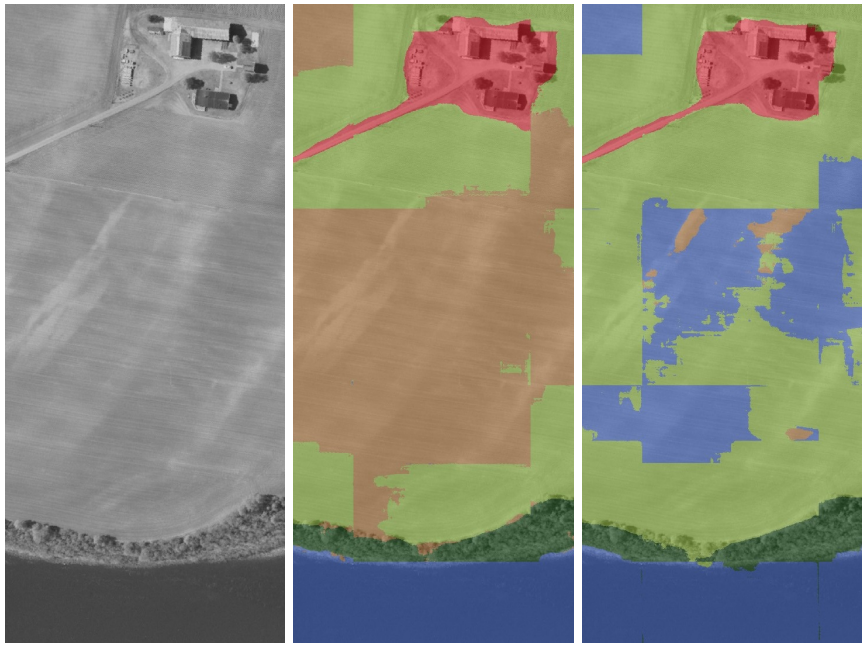
The metrics for the predictions for this experiment is shown in Table 5.4. For Gaula 1998 the miou changed from 53.81% without intensity adjustments to 65.18% with intensity adjustments, a quite significant amount. Nea 1962 had a mean light intensity of 90, and the intensity adjusted predictions were slightly worse, 74.95% miou compared to 77.49% miou for the predictions made without intensity adjustment.

	miou	accuracy	precision	recall
Nea 1962	77.49%	93.40%	91.61%	95.36%
Nea 1962 intensity adjusted	74.95%	92.39%	90.75%	81.71%
Gaula 1998	53.81%	73.17%	69.77%	89.20%
Gaula 1998 intensity adjusted	65.18%	85.24%	75.85%	76.73%

Table 5.4: Test scores for the Nea 1962 test set and the Gaula 1998 test set, with and without light intensity adjustments. Precision is average precision and recall is average recall.

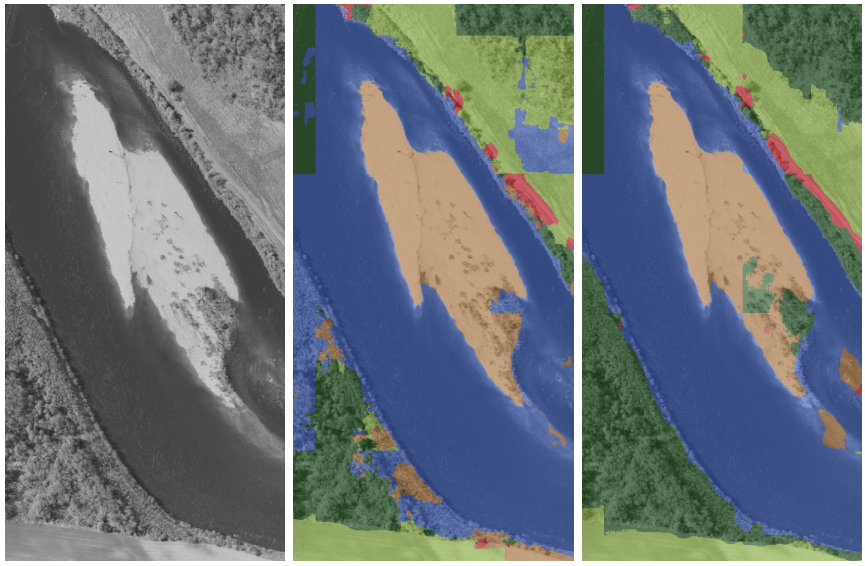
In Figure 5.14 we can see that the model segments farmland as gravel with the normal predictions while it segments it as partially water in the case of the intensity adjusted predictions. The gravel texture is brighter, so it is more likely to be predicted in the bright non adjusted image, while the water texture is darker and is more likely to be predicted in the adjusted darker image. Another observation from Figure 5.14 is that the wrong predictions is in generally the same area of the images, indicating high uncertainty from the model in that area regardless of the brightness.

In Figure 5.15 the model correctly found the gravel island and the surrounding water for both normal predictions and the intensity adjusted predictions. Vegetation is wrongly segmented as water (bottom right corner) and as farmland (top left corner) in the prediction without intensity adjustments. Vegetation has a dark texture in the training data and when the bright images were not adjusted it became too bright to be segmented properly. In the intensity adjusted predictions both areas were correctly segmented as vegetation.



(a) Image (b) Predictions (c) Intensity adjusted

Figure 5.14: Gaula 1998. Comparison of predictions with and without intensity adjustments. Images of wrongly segmented farmland.



(a) Image (b) Predictions (c) Intensity adjusted

Figure 5.15: Gaula 1998. Comparison of predictions with and without intensity adjustments. Images of a gravel island.

Chapter 6

Evaluation and Conclusion

In this chapter the work in this thesis will be discussed, the contributions will be presented and some avenues for future work will be shown.

6.1 Evaluation

Four different test set was used to evaluate the performance of the model on different rivers and at different time periods. This made it possible to compare determine if the model generalized to other rivers than the training rivers. An issue with the light intensity was discovered due to this testing on Gaula 1998 and a solution was proposed and tested in Section 5.4. If this work had only used a randomly sampled test set from the original dataset then the comparison to different rivers would have been a lot weaker.

This thesis did not compare the performance of the CNN models to a simple baseline algorithm. This makes it difficult to determine if machine learning was needed or if the same results could have been archived by a simple baseline. The reason neural networks were chosen as the model was that rivers can have a huge variety in shapes and textures, and in order to capture the features for rivers in general a model that could adapt itself to fit many different datasets was needed. This is also the reason for training the model on images from more than one river.

6.2 Discussion

This section will discuss the results presented in Chapter 5. The predictions were made on images of the Gaula river from 1947, 1963, 1998 and from Nea 1962.

The aim of this discussion is to find the strengths and weaknesses of the model and determine how the model can be used for automatic ecological analysis.

An issue encountered when applying the model to the Gaula 1998 test set was that the light intensity was different from the light intensity in the training data. In Section 5.4 predictions were made after first adjusting the light intensity of the images. For the Gaula 1998 test set the mIoU increased from 53.81% without intensity adjustment to 65.18% with intensity adjustments. This is a major improvement, but the results are however still below that of the results on the other test sets (see Table 5.3). This suggests that only adjusting the light intensity mean is not enough to compensate for the differences among the datasets.

The model has a high performance (71.3% mIoU) on the test set from Gaula 1963 for all the different classes except human-constructions. On water, vegetation and farmland both recall and precision are above 80% for the Gaula 1963 test set (see Table 5.3 and Figures 5.1b and 5.2b). For this data the model could be used to look large scale changes in land if gravel and human-constructions are manually checked and if needed corrected.

Surprisingly the model performed better on the test set for Nea 1962 than for the Gaula 1963 test set with 77.49% mIoU for Nea 1962 compared to 71.3% for Gaula 1963. In the Nea 1962 test set the recall for the gravel class was only 53.15% while its precision was 94.34%. This means that almost half of the gravel in the images was not segmented as gravel by the model, but for those areas that was segmented as gravel 94% was actually gravel. For all other classes both recall and precision was above 80% (see Table 5.3 and Figures 5.1a and 5.2a). For this dataset manually segmenting the gravel areas that the model missed would allow for large scale analysis of the river.

For some parts of the datasets it was difficult to make a segmentation based on the local features in the image. An example of this is shown in Figure 5.13. In this example some vegetation is covered by a shadow so it is almost impossible to make a correct segmentation based only on the local information. With the current model this area was wrongly segmented as waters. This is caused by a limitation in the model. The limitation is that the model does not use any of the context around the input image, it only uses the 512x512 input image that it is given to make the output segmentation. If the model was able to combine the local features with the context around the local features, it could have been possible to correctly segment the shadow area as vegetation because all the surrounding area was correctly segmented as vegetation.

To answer **RQ1** "Can an approach using black-and-white aerial images and neural networks divide rivers into ecological segments?", a neural network model can divide a river into ecological segments. The model can be used to make predictions for new rivers from the same time periods as the training data as

well as for different time periods. When using the model on images that are either much brighter or much darker than the training images light intensity adjustments should be made. Before the predictions from the model are used for analysis a manually qualitative assessment should be made, to detect issues with certain classes. The tests done in this thesis indicate that for some datasets gravel can be wrongly segmented.

6.3 Contributions

The first contribution is a training/validation dataset containing 6307 images covering a 66 km^2 area from Gaula 1963, Surna 1963 and Lærdal 1976. As well as a test set of in total 927 images covering a 9.7 km^2 area. This dataset will be published with a later publication. 24.5% of the dataset was made during fall project. The rest of the dataset was made during the work for the master thesis.

Another contribution is the CNN model that can be used to segment black-and-white aerial images of rivers into ecological segments that can be used for large scale ecological analysis. The model and code for using the model can be found on `https://github.com/arildsd/river-segmentation`.

A short communication for the River Research and Applications journal is being made in parallel with this thesis by Knut Alfredsen. The short communication uses the work done in this thesis to demonstrate how neural networks can be used for remote sensing applications on rivers.

6.4 Future Work

Try with a lower resolution of for example 0.5 meters instead of 0.2 meters. This would allow for more aggressive data augmentation methods to be used without using a huge amount of memory. The loss in detail for images would be small, since the main limiting factor is the quality of the physical images that were scanned.

To overcome the light intensity issue presented in Section 5.4 a more elegant solution would be to use standardization as pre-processing step when training the model as well as when making predictions using the model.

Different models like Dense Net 121 (Huang et al. [2016]) could be a better candidate for transfer learning because it has fewer trainable weights, but still has a good performance on the image net dataset. It was not done for this thesis because Dense Net is conceptually more complex and more difficult to implement than VGG16.

Using a model that can use information about the surrounding areas of an image would allow high quality segmentation to be made even when the local

features are hard to use.

Using post-processing to remove some errors like farmland predicted in the middle of the river that is completely surrounded by water.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Casado, M.R.; Gonzalez, R. K. T. V. A. (2015). Automated identification of river hydromorphological features using uav high resolution aerial imagery. *Sensors*.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848.

Fukushima, K. (1988). Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural networks*, 1(2):119–130.

GDAL/OGR contributors (2020). *GDAL/OGR Geospatial Data Abstraction software Library*. Open Source Geospatial Foundation.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. (2016). Densely connected convolutional networks.

Iglovikov, V. and Shvets, A. (2018). Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation. *arXiv preprint arXiv:1801.05746*.

Kartverket (2019a). Kartverket. `https://www.kartverket.no/en`. Accessed: 2019-11-23.

Kartverket (2019b). Norge i bilder. `https://www.norgeibilder.no`. Accesed: 2019-11-23.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc.

Maggiori, E., Tarabalka, Y., Charpiat, G., and Alliez, P. (2017). Can semantic labeling methods generalize to any city? the inria aerial image labeling benchmark. In *2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, pages 3226–3229.

Murray, N. J., Phinn, S. R., DeWitt, M., Ferrari, R., Johnston, R., Lyons, M. B., Clinton, N., Thau, D., and Fuller, R. A. (2019). The global distribution and trajectory of tidal flats. *Nature*, 565(7738):222–225.

Nielsen, M. A. (2015). *Neural Networks and Deep Learning*. Determination Press.

Oliphant, T. E. (2006). *A guide to NumPy*, volume 1. Trelgol Publishing USA.

QGIS Development Team (2020). *QGIS Geographic Information System*. Open Source Geospatial Foundation.

Richard, A., Benbihi, A., Pradalier, C., Perez, V., Durand, P., and Van Couwenberghe, R. (2018). Automated segmentation and classification of land use from overhead imagery.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In Navab, N., Hornegger, J., Wells, W. M., and Frangi, A. F., editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham. Springer International Publishing.

Ross-Howe, S. and Tizhoosh, H. R. (2018). The effects of image pre-and post-processing, wavelet decomposition, and local binary patterns on u-nets for skin lesion segmentation. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition.

Själander, M., Jahre, M., Tufte, G., and Reissmann, N. (2019). EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure.

Tang, W., Zou, D., Yang, S., and Shi, J. (2018). Dsl: Automatic liver segmentation with faster r-cnn and deeplab. In *International Conference on Artificial Neural Networks*, pages 137–147. Springer.

Van Rossum, G. and Drake, F. L. (2009). *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

Wang, L., Liu, H., Lu, Y., Chen, H., Zhang, J., and Pu, J. (2019). A coarse-to-fine deep learning framework for optic disc segmentation in fundus images. *Biomedical Signal Processing and Control*, 51:82–89.

Xia, K. and Yin, H. (2019). Liver detection algorithm based on an improved deep network combined with edge perception. *IEEE Access*, 7:175135–175142.

Xu, Z., Wu, C., Zheng, E., Liu, Y., Guan, S., and Ma, Y. (2019). Semantic segmentation of buildings in remote sensing images based on dense residual learning and channel adaption. In *2019 4th International Conference on Electromechanical Control Technology and Transportation (ICECTT)*, pages 117–123.

Appendices

6.5 Confusion matrices

This section contains unnormalized confusion matrices. The values in the matrices are pixels.

Gaula 1963		Predicted class				
		W	G	V	F	H
	W	4616865	18974	69677	350626	21
	G	194870	1906062	166941	151528	64632
	V	227115	189143	9628260	248917	530167
	F	27026	112405	377188	3972452	18994
True class	H	27036	20778	69610	86733	744163

Figure 6.1: Gaula 1963 raw data confusion matrix.

Gaula 1998		Predicted class				
		W	G	V	F	H
	W	7353219	321547	174580	389028	5218
	G	50959	489946	16740	121641	50009
	V	1165300	597656	3565740	954221	234341
	F	3767	2615122	91289	5260939	309589
True class	H	545	175361	130638	520619	4958515

Figure 6.2: Gaula 1998 raw data confusion matrix.

Gaula 1998 intensity adjusted		Predicted class				
		W	G	V	F	H
	W	7708403	22373	459844	49264	3708
	G	72177	288235	169978	118354	80551
	V	254497	12107	5658350	326239	266065
	F	183542	528207	328503	6848724	391730
True class	H	37530	20418	640737	395003	4691990

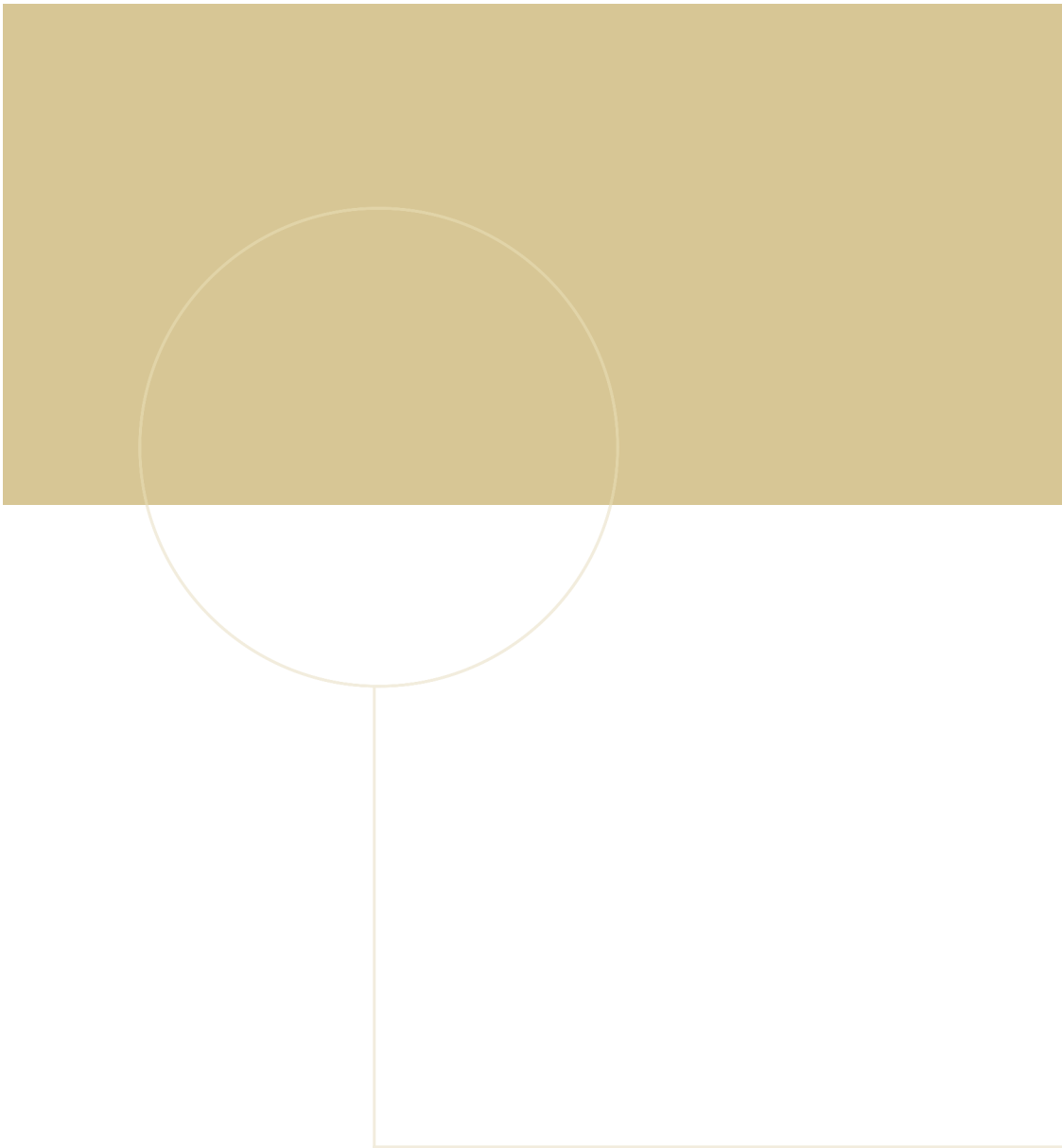
Figure 6.3: Gaula 1998 raw data confusion matrix. Intensity adjusted predictions.

Nea 1962		Predicted class				
		W	G	V	F	H
	W	17401611	24818	334146	435232	51839
	G	500887	1173842	177621	222346	133663
	V	762247	27885	21953073	1113104	398508
	F	918807	17745	576601	49842203	139870
True class	H	4576	0	137073	693214	4064195

Figure 6.4: Nea 1962 raw data confusion matrix.

Nea 1962 intensity adjusted		Predicted class				
		W	G	V	F	H
	W	17532825	18837	380304	268354	47326
	G	537988	1072987	219348	256653	121383
	V	792477	28268	21822224	1234176	377672
	F	1044748	9000	1108364	49141264	191850
True class	H	10051	1126	218151	824388	3845342

Figure 6.5: Nea 1962 raw data confusion matrix. Intensity adjusted predictions.



NTNU

Norwegian University of
Science and Technology