Master's Programme in Data Science

Semantic Segmentation with Neural Networks in Environment Monitoring

Johannes Elmnäinen

August 29, 2020

Supervisors:	Associate Professor Laura Ruotsalainen
	Dr. Joel Pyykkö
Examiners:	Associate Professor Laura Ruotsalainen
	Dr. Joel Pyykkö

UNIVERSITY OF HELSINKI
FACULTY OF SCIENCE
P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty Faculty of Science	Koulutusohjelma — Utbildningsprogram — Degree programme Master's Programme in Data Science

Tekijä — Författare — Author

Johannes Elmnäinen

Työn nimi — Arbetets titel — Title

Semantic Segmentation with Neural Networks in Environment Monitoring

Työn laji — Arbetets art — Level Master's thesis	Aika — Datum — Month and year August 29, 2020	Sivumäärä — Sidantal — Number of pages 81

Tiivistelmä — Referat — Abstract

The Finnish Environment Institute (SYKE) has at least two missions which require surveying large land areas: finding invasive alien species and monitoring the state of Finnish lakes. Various methods to accomplish these tasks exist, but they traditionally rely on manual labor by experts or citizen activism, and as such do not scale well. This thesis explores the usage of computer vision to dramatically improve the scaling of these tasks. Specifically, the aim is to fly a drone over selected areas and use a convolutional neural network architecture (U-net) to create segmentations of the images. The method performs well on select biomass estimation task classes due to large enough datasets and easy-to-distinguish core features of the classes. Furthermore, a qualitative study of datasets was performed, yielding an estimate for a lower bound of number of examples for an useful dataset.

ACM Computing Classification System (CCS):
CCS → Computing methodologies → Machine learning → Machine learning approaches → Neural networks

Avainsanat — Nyckelord — Keywords

lake biomass estimation, invasive alien species, neural networks, semantic segmentation, U-net

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

1. Introduction

In 2019 the Anthropocene Working Group voted to suggest a beginning of a new geological epoch to the International Commission on Stratigraphy [64]. The new epoch, called ”anthropocene”, would signify the plethora of effects human actions have on Earth. As the global consciousness on environmental issues has risen, new methodologies for combatting human effects on the environment are required. However, any method to improve the state of the environment requires good data on its current state. This thesis explores the usage of computer vision as a method for monitoring the state of environment. Two use cases are presented: the first is using computer vision to estimate biomass in lakes. The second is finding *Invasive Alien Species* (IAS) from the environment. This thesis explores the possibility of using a U-net neural network architecture [60] to produce estimates on both invasive alien species and lake biomass in an area. The network generates estimates by producing a *semantic segmentation* (ie. a pixelwise prediction task) on aerial images taken by a drone.

Alien species are flora and fauna which have travelled outside their traditional habitats [48]. While species can migrate on their own, human aid has accelerated the introduction of species to new ecological habitats since the Age of Discovery [9]. However, introduction of a new species should not always be considered unnatural in itself. Species can migrate by themselves to new habitats in response to various migratory pressures, such as destruction or overcrowding of original habitat. Furthermore, it is sometimes unclear if the species is new per se. For example, it is possible for a species to return to an area it previously migrated from. Sometimes a return of a species is even achieved through human intervention in rewilding schemes such as in the case of the European bison. The European bison survived only in zoos until rewilding attempts were made, returning it to original habitat from which it was lost [59].

Despite the arguments above, it is clear that human actions have accelerated invasive events. Furthermore, some new species have negative effects on their new ecological habitats. Such effects are disrupting the species hierarchy, nutrient cycle and plan productivity of the ecosystem they invade. Alien species which disturb their new habitat are called invasive alien species. Although the term ”invasive alien species” has faced contestation [17], the term is used by the European Union in its legislation [66].

As the focus of this thesis is in the implementation of European Union's legislation, the term ”invasive alien species” will be used in this thesis as well.

Because of their adverse effects to the environment, the European Union has regulated the prevention and management of the introduction and spread of invasive alien species [66], which Finland has implemented as part of its national legislation [55]. In the EU regulation, an invasive alien species is defined as

1. having been introduced to its new environment through human assistance,
2. have a serious negative influence on the biodiversity or ecosystem services of the environment it has been introduced to and
3. being able to reproduce.

The regulation mandates member states to mitigate the damage caused by invasive species. In Finland, tasks related to protection of the environment have been delegated to Centre for Economic Development, Transport and the Environment and Finnish municipalities [55]. The Ministry of Environment oversees and coordinates the work. In addition, the Finnish Environment Institute (SYKE) acts as a state-funded center for research and expertise within the environmental field.

However, locating invasive alien species is a difficult problem. The naive solution would be to have experts exploring the land, spotting IAS as they go. This solution does not scale well vis-à-vis personnel costs, especially in sparse countires like Finland. Another way to solve this problem is to educate civilians to spot invasive species and require them to act against them. Indeed, Finland has a law on managing the risks posed by invasive species [55]. The law requires owners of property to eradicate or contain invasive species to a reasonable degree. However, this does not mandate action on property owned by the government. Thus, the same law which requires civilians to eradicate IAS mandates SYKE to remove IAS where encountered. As such, SYKE has a need for an efficient way of monitoring large patches of land with as low costs as possible. This has been taken into account in the Finnish Government's Alien species strategy, which instructs SYKE to create methods for IAS surveillance in Finland [67].

The other application for computer vision in this thesis is measuring lake biomass. Biomass estimation can be used as a proxy variable for eutrophication [12]. Eutrophication can cause the body of water to suffer from cyanobacteria, make the water undrinkable, reduce its recreational value and ultimately cause hypoxia, which leads to death of lake life through suffocation. Eutrophication can have more subtle effects as well. For example, increased algal blooms in the water can reduce visibility, which hinders predatory fish's ability to pursue their prey. To protect the lakes, SYKE has an interest in keeping track of lake biomass in Finland.

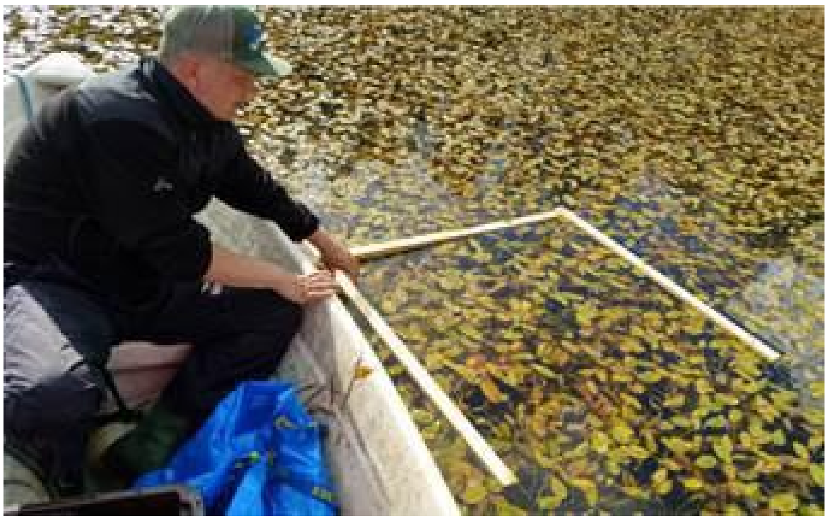


Figure 1.1: The traditional method for lake biomass estimation at SYKE.

In this work, biomass estimation is implicit, as opposed to explicit measurement [63]. Specifically, water biomass is estimated through features visible on surface level. The explicit alternative would be to dig up the biomass and weigh it, but this method has direct consequences to the body of water and makes longitudal studies difficult.

Traditionally, SYKE has estimated lake biomass with a human resource-intensive solution. This method involves a researcher on a rowboat and a square net to estimate the amount of biomass in a lake, as depicted in Figure 1.1. A large enough area of the lake is sampled, after which a reliable estimation can be achieved. The drawback of the traditional method is that the process consumes large amounts of a professional's time.

This thesis presents *semantic segmentation* as part of a method for large-scale detection of IAS in the wild as well as biomass estimation. Semantic segmentation is a process where an image is divided into its constituent parts, each of which should have a meaningful interpretation [71, 25]. These segmentations allow computer systems to interpret visual scenes and detect objects in them. Such systems can operate in various domains, such as self-driving vehicles [72], medical imaging [60] or terrain mapping [40].

As an example of input and output for the segmentation task at hand, Figure 1.2 has two images: in the left image there is a orthogonal picture of a winter dock in Kaisaniemi, Helsinki, during winter. The image on the right is a hand-crafted segmentation of the picture on the left side. The different segments have the following semantic meanings: sea(blue), walkways(ochre), docks(purple), cars(dark green), boats(light green), driveways(beige), trash bins(grey) and parks(yellow). It should be noted that the segmentation given in Figure 1.2 is but one of many possible interpretations. Other valid segmentations could be land-sea, ice-water, people-background, etc.

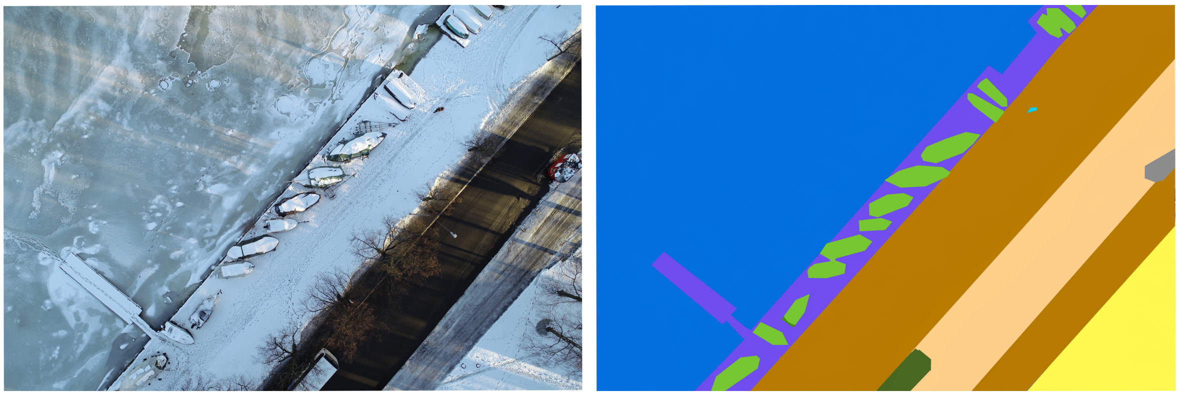


Figure 1.2: An example of image segmentation. Left: an orthographic picture of Kaisaniemi during winter. Right: a segmentation of the different objects.

A single image can also have multiple correct segmentations depending on semantic domains and desired granularity. For example, take an image of a countryside road. For a system that governs driving a car, interesting classes are the edges of the road, markings on the road, other vehicles, humans and animals. Meanwhile, the specific species of roadside flora can collectively be classified as *vegetation*. However, for an application designed to catalogue the different species found along the roads, differentiating between said species is important, while the markings on the road and road signs are irrelevant.

Typically, a semantic segmentation system attempts to predict a single class for a single pixel [71]. This differs from humans' ability to potentially assign multiple classes to single pixel in cases where that pixel consists of a translucent object, for example a glass. While this phenomenon is interesting, due to its philosophical nature it is outside of the scope for this thesis. However, it should be noted that a similar problem is encountered in the biomass estimation task, where images of lakes are in question. Here, the problem is as follows: given a plant which slowly submerges until it cannot be seen, which pixels should be considered as "water" and which "plant"? In this thesis, the answer is "the pixels which the human annotator annotates as plants are plants", and the potential resulting degradation of accuracy is accepted.

As this thesis is concerned with mapping surface areas, orthogonal aerial images of target locations are needed. To this end, drones are flown over the locations with cameras pointing directly downwards, resulting in images with perspectives similar to the left-hand image in Figure 1.2. These images are then fed to a model trained for semantic segmentation task, ideally resulting in a segmentation similar to the image on the right side.

In this thesis three different image processing methods appear. These are *image segmentation*, *image classification* and semantic segmentation. This is because semantic segmentation can be seen as the combination of image segmentation and image classification.

Image classification is a task where one or more labels are produced for the objects in the image. For example, the left image in Figure 1.2 could be labelled as ”terrain” or ”docks” in a image classification task, or given multiple labels, such as ”dock, boats, road”. The main difference to semantic segmentation is how this information is presented. Semantic segmentation aims to produce a pixelwise prediction, i.e. each pixel is assigned a class it belongs to, while image classification is concerned with associating the labels on the whole image, but is not interested in where in the image these labels are. Historically, this has caused regular classification to be more popular due to the smaller complexity of the model and the ease of producing annotated data. To train a model to classify an image to single class, pairs of (image, label) are needed. While not trivial, this dataset can be generated easier than that of semantic segmentation, where ideally each pixel in each image needs to be labelled correctly.

On the other hand, image segmentation can be defined as subdividing the image into regions that have inner cohesion and are distinguishable from their neighbours [25]. As such, it differs from semantic segmentation in that semantic segmentation stresses the importance of object detection and classification. The added requirement of having semantic interpretation of the segmentation has traditionally made semantic segmentation considerably harder task when compared to image segmentation. This is because usually the spatial relations of images do not yield hints to their semantic makeup. For example, a backside of a honey bee has a single semantic interpretation which is hard to capture by observing only the spatial dimensions of the image - namely the yellow-and-black stripes.

As mentioned previously, semantic segmentation can be seen as a combination of image classification and image segmentation. It attempts to divide the image into regions (segments) where the inner cohesion is achieved by classifying each segment into a singular class.

The contents of this thesis are as follows: in chapter 2, segmentation as a problem space is discussed and methods earlier than neural networks for both non-semantic and semantic segmentation are presented. Furthermore, related work for biomass estimation is discussed. In chapter 3, a brief introduction to feedforward and convolutional neural networks is given. Following that, in chapter 4 the thesis will present two central architectures for doing segmentation with neural networks: a fully-connected convolutional neural network and the U-net. Finally, a report on practical application of semantic segmentation to detect IAS as well as to estimate lake biomass is given in chapter 5. Here, the dataset is analysed and exact training methods, parameters and results given. Finally, chapter 6 includes conclusions and discussions for future research.

2. Related work

In this chapter, the philosophy of image segmentation is discussed. Furthermore, various traditional image segmentation as well as biomass estimation methods are presented.

Image segmentation is a step in an image processing pipeline [54]. It is used to simplify the image or capture regions of interest. As a problem space, image segmentation is concerned with dividing an image into regions based on defining features, for example contours or changes in colour palette [25]. The goal of image segmentation is to assign a region for each pixel in the original picture so that no regions overlap with each other, that each region is similar according to some metric, and no neighbouring regions are similar according to that metric.

Formally, image segmentation is defined as a set of regions $X_1, X_2, ..., X_n$ and uniformity predicate P, such that

$$\begin{aligned} \cup_{i=1}^{n} X_i &= X, \\ X_i \cap X_j &= \emptyset, i \neq j \\ P(X_k) &= TRUE, \\ P(X_k \cup X_l) &= FALSE \end{aligned}$$

where $i, j, k, l \in [1, n]$ and X_k and X_l are adjacent. The latter rule might benefit from elaboration: it simply states that the union of two adjacent regions must not be uniform. If their union would be uniform, they would naturally belong in the same region. Additionally, a simple example of uniformity predicate P in a monochromatic image with pixel brightness measured from 0 (white) to 255 (black) could be ”Pixel’s brightness is less than 127”.

The definition given above means that for a single image there are multiple different segmentations which can be considered valid. Furthermore, there are multiple approaches to image segmentation, such as characteristic feature thresholding or clustering, edge detection or region extraction [54]. None of these is the single correct algorithm, but each works well on a specific domain.

Traditionally, research has focused on methods that work on monochromatic images - that is, images that consist of single colour of varying intensity [11]. A classic setting is a grayscale image, for example a magnetic resonance image (MRI) of brain. However, the rising computational power has made the use of colour information practical. This broadens the possibilities and challenges for techniques by increasing the dimensions they can work in.

2.1 Non-semantic segmentation methods

As mentioned before, a plethora of image segmentation methods exists. While the focus of this thesis is in semantic segmentation done with neural networks, this chapter presents two traditional methods for context. The presented methods are thresholding and watershed.

2.1.1 Thresholding

Thresholding is primarily a monochromatic image segmentation technique, although variations for multichromatic scenarios do exist [11]. It has been used for example in medical imaging. Thresholding uses histogram of the image pixel luminosity (i.e. histogram depicting how much of certain brightness of grey exists in the image) to detect regions. Formally, thresholding an image consisting of pixels in locations (x, y) into m regions can be defined as follows [25]:

$$S(x, y) = k \Leftrightarrow T_{k-1} \leq f(x, y) < T_k, \tag{2.1}$$

where S is the region assignment function, f is the feature function (e.g. gray level function), m is the number of desired regions, $k = 0, 1, ..., m + 1$ and $T_0, ..., T_m$ are the threshold values for segments. Notice that k is required to get values from 0 to amount of regions + 1 because equation 2.1 requires us to define both the upper and lower limit to the region. Basically this means that each pixel is allocated into a region based on some of its properties; traditionally this property has been its position on a scale from white to black. As an example, suppose again that a monochromatic image with brightness values in $[0, 255]$ should be divided into two regions with brightness < 127 and brightness $>= 127$. The thresholds chosen would now be $k_0 = 0, k_1 = 127$ and $k_2 = 256$.

The problem then reduces to selecting the thresholds T. Myriad ways of expressing T exist. Generally [25], these can be divided to three classes. Thresholds that are expressed only through f are called global. These methods only use the values of x and y. If in addition to x and y the threshold uses data from the area around (x, y) it is called local, while dynamic thresholds use coordinate information of (x, y), the values at x, y and the values at the area around x, y. A simple global method for choosing

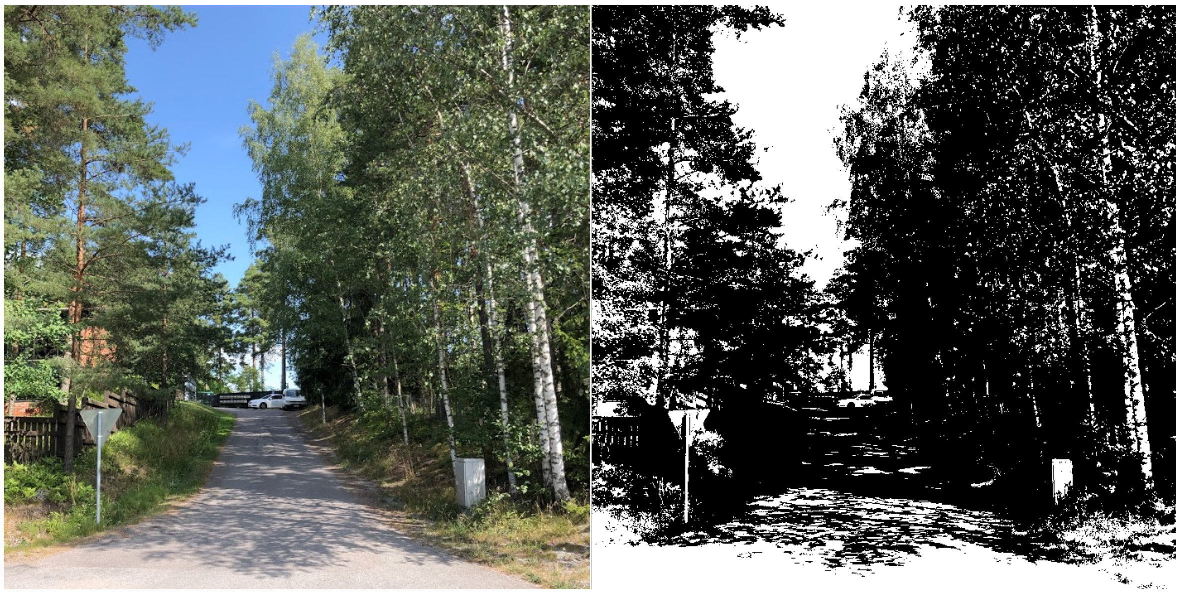
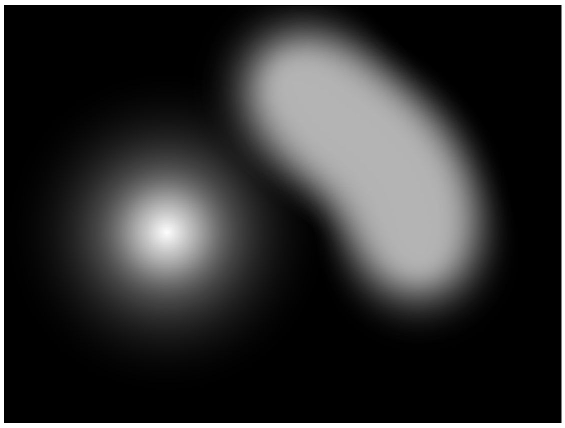


Figure 2.1: An example of thresholding with two classes. Left: the original picture of a street in Kaarina, Finland. Right: a segmentation into two classes of the original picture.

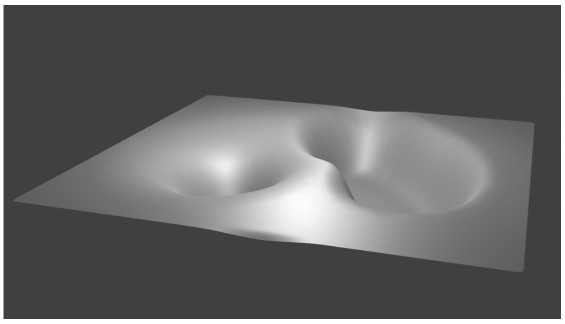
a threshold for monochromatic images is to create a histogram of colour luminosities and, identify the peaks and valleys, and set thresholds T to equal the valley values.

An example of thresholding can be seen in Figure 2.1, which depicts a street in Kaarina, Finland. The original image has been segmented into two classes (represented by black and white) by using GNU Image Manipulation Program's [68] thresholding function. This is the simplest thresholding, called bilevel thresholding [54]. Because the image is colourful instead of grayscale, pixel brightness is used to first map the image into a grayscale representation. This step is not depicted. A global function of pixel brightness was used, where pixel brightness took values from 0 (black) to 256 (white). Here, $T_0 = 0$, $T_1 = 127$ and $T_2 = 256$. Values 0 and 256 were selected because they are the lowest and highest possible values - otherwise some pixels would not have received a classification. The value 127 was achieved through trying out various options and selecting the one that looked good. Note that in reality, only T_1 is selected by the user in GNU Image Manipulation program. This is because practical applications necessarily do not need to concern themselves with theoretical soundness. Importantly, the Figure 2.1 reveals that while the segmentation is achieved with thresholding, it does not have semantic meaning. For example, both classes contain trees, road and parts of the fence. As such, the segmentation has little meaning and is hard to use as a part of larger computer vision pipeline. However, the technique can be used to perform meaningful tasks. For example, the technique has been used to generate segmentations in breast cancer images [33].

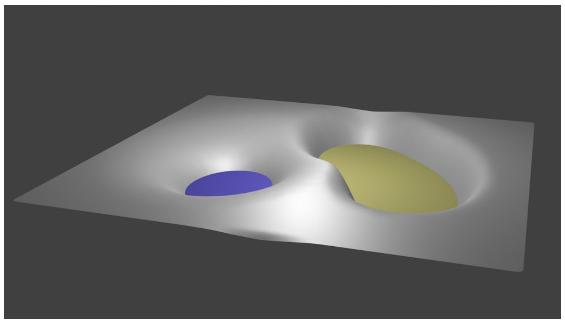
2.1.2 Watershed



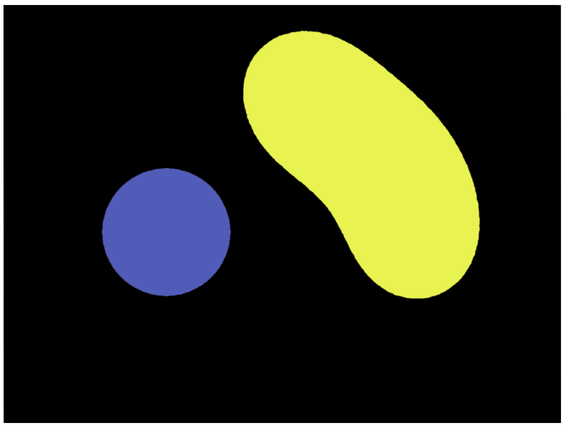
(a) Original grayscale image



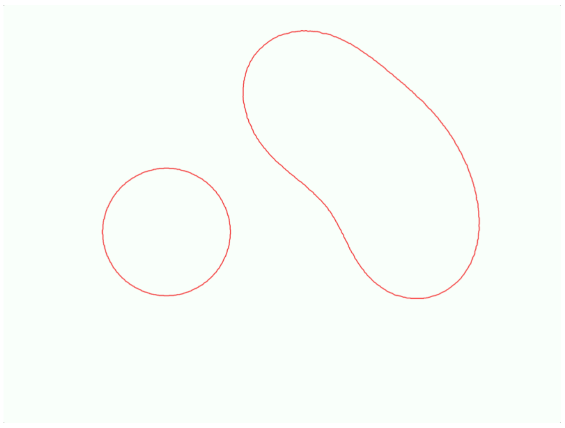
(b) Topographical interpretation of original grayscale image



(c) Beginning to fill the topography from the two basins



(d) Final filling of the image



(e) Final segmentation, or watershed lines

Figure 2.2: Example of watershed method on a simple greyscale image. Note that images b) and c) are metaphorical and included to make understanding of the watershed method easier. The actual watershed method does not produce 3D images.

Watershed methods share their namesake metaphor. The basic idea is likened to a geological watershed [54], which cause water to drain from high ground into surrounding lowlands (catchment basins). Given enough water (for example rainfall), these high ground formations would become a divisor between bodies of water, i.e. a watershed. Mathematically, any grayscale image can be considered a topographical surface, i.e. the darker the pixel is, the higher it is considered to be, or vice versa. Essentially, a greyscale image can be transformed into a heightmap. Then, the map is metaphorically filled with different, non-mixable colours. This filling can be thought of as finding all the local minimas of the landscape, and beginning to flood the area with different-coloured non-mixable liquids from them.

An example of the watershed method can be seen in Figure 2.2a. Here, a simple example with two main areas is given as an illustration on how the watershed algorithms work. Figure 2.2a denotes the original image. In Figure 2.2b a topographical interpretation of 2.2a can be seen. Figure 2.2c shows a metaphorical ”filling of basins” from local minimas, and figures 2.2d and 2.2e are produced by actually running a watershed algorithm found in OpenCV[70] package on image 2.2a. They show the segmentation and segmentation lines, respectively. Although the image seen in Figure 2.2d has been produced by the actual OpenCV implementation of watershed algorithm, it has been colour corrected in an image manipulation program to ensure colour consistency with the image in Figure 2.2c. As can be seen, the algorithms correctly identifies the two areas. However, it should be noted that this example was extremely simplified.

While rigorous mathematical definitions on watershed methods are available, this thesis will forego presenting them, as they are rather lengthy and not directly relevant to the subject matter at hand. An interested reader can find these in [7], [54] and [32].

The watershed method has been used in the field of medical imagining [32], steel imperfection detection [7] and bubble detection in radiographic plates. However, it suffers from various problems, such as oversegmentation (tending to create more segments than necessary, especially if there are multiple basins in the image), sensitivity to noise in the image, and poor detection of regions with low contrast or thin basins. For example, two similar coins next to each other can easily be classified as separate by humans, but prove difficult to separate with watershed because it relies purely on spatial information. Some of these problems can be alleviated. For example, oversegmentation can be improved by providing a separate marker function, which specifies which local minimas flood the landscapes and which do not. This results in a desired amount of classes with the additional requirement that the user should specify the marker function.

2.2 Semantic segmentation methods

The key difference between image segmentation and semantic segmentation is that semantic segmentation attempts to create regions with semantic meaning [71]. While the thresholding and watershed methods might result in a segmentation with semantic meaning, this should be considered more of an accident than a rule. The methods this thesis has presented up until this point have been satisfied when some mathematical property (e.g. all the pixels with RGB value of (148, 158, 0) or greater are classified as ”background”), has been filled. This leaves room for potential errors, such as the problem of segmenting a bee’s backside mentioned previously. Other examples of error states rising from the lack of semantics include, for example, a situation where the object is occluded [35] by another object. Such situations are fairly typical in the real world - an example could be a streetlamp in front of a car. Thresholding and watershed have no means to classify the two parts of the car as a single object which just happens to be behind another object. However, such a task is trivial for most humans. Another common example of failure cases are situations where objects in the foreground have textures similar to those found in the background, for example a soldier wearing camouflage. Yet another common flaw can be seen in OpenCV’s [3] watershed tutorial, where watershed algorithm tries to classify a collection of coins. While the image is almost correctly segmented, there are error cases where a segment has leaked from one coin to another because their colours has been close enough. Such an error would never be made by humans, because humans have an inner model of the concept of a ”coin”.

Successful semantic segmentation systems developed before the advent of neural network methods included *random forests*, *support vector machines*, *Markov random fields* or *conditional random fields* [71]. While it is trivial to claim that the traditional methods of watershed and thresholding do not build a semantic understanding of the image, it is more difficult to argue the same about the methods presented in this section. Typically, they are not framed as building a semantic understanding of objects in the image, but a well-trained distribution of probabilities. However, some of them use methods similar to convolutional neural networks. For example Shotton et al’s method [62] uses the convolution operation, which is the basic building block of convolutional neural networks. Convolutional neural networks in turn have been claimed to build an understanding of objects in the image [30]. While these questions of understanding are fascinating, they are out of scope for this thesis. As such, this thesis will adopt a practical stance: it can be argued that some of the methods presented here build a understanding of concepts. However, in the case of neural networks this claim is stronger. Although this understanding is weak and does not necessarily follow human logic, it helps neural networks achieve good results on image recognition and segmentation tasks.

In this section, the accuracies of reviewed methods are often discussed. However, earlier papers do not tend to specify how they define ”accuracy” - a good guess is they calculate how many pixels were given the correct class. The newer papers tend to use *intersection over union* (IOU, see equation 4.1). As such, these accuracies are not directly comparable. Furthermore, the datasets they use for validating their results differ. Therefore, it is difficult to have an honest comparison between all the methods without building a complicated map of competitions, their datasets and comparisons. Whenever the accuracy of the methods presented below are compared to convolutional neural networks, the comparison should be considered as a somewhat enlightened guess, not objective truth. U-net's (see chapter 4.3) IOU scores of 0.7756 and 0.9203 on two different datasets are used as a baseline.

Shotton et al [61] used random forests to produce a semantic segmentation. These are an ensemble of decision trees, where each node is a learned class distribution. Shotton et al. use a special case of a random forest: a semantic random forest. This is achieved by passing $d * d$ patches of images down a random forest, and using an image filter as a discriminant in each node. By applying said procedure on regions of image, they produce a bag of semantic textons. Shotton et al. report 42% - 66.9% accuracies on various datasets, a result which is eclipsed by those of modern CNN architectures.

Yang et al. [73] used bounding boxes produced by object detection as a guide for segmentation. Their method uses support vector machine-based detection of features. The features detected are appearance, depth ordering and labels of entities in the image. Objects detected by the bounding boxes are arranged in depth-wise layers so that the first layer contains the object which is closest to the camera, second the second closest, and so forth. This helps the final segmentation process. They came in 7th in Pascal VOC 2009 [20] and 2010 [21].

Zhang et al. [76] used a Hidden Markov Random Field (HMRF) combined with their expectation-maximization (EM) algorithm to produce a semantic segmentation on magnetic resonance images (MRI) of brains. Essentially, a Markov Random Field treats each pixel and its corresponding label as conditionally independent random variables given their local neighborhood, and a HMRFs state is unobservable. They learn the distribution $P(x, y)$ where x represents a pixel value and y corresponding label. This enables them to produce a segmentation that takes into account the context of a given pixel. The authors have not included any results from segmentation challenges, and themselves discuss the difficulties involved in choosing initial tissue parameters and classification, noting that this is a good step for future research.

Conditional random fields (CRFs) are similar to MRFs, but instead of learning the distribution $P(x, y)$, they learn the distribution $P(y|x)$, which greatly reduces the number of random variables in the model. Furthermore, no assumptions of the distribution of x has to be made. As with MRFs, x denotes a pixel value and y its label. Gonfaus et al. [29] presented usage of CRFs with a new potential function called the harmony potential. In short, a potential function make up a part of the calculation $P(y|x)$ in conditional random fields. Harmony potential uses a global random variable to penalize the per-pixel random variables when they try to take on values outside those allowed by the global variable. They achieved results similar to Shotton et al [61]. As such, they fall short of results achieved by neural networks. However, they did achieve first place in Pascal VOC 2010.

In addition, a CRF was used by Shotton et al. [62]. Their random field had four features, which were texture-layout, colour, location and edges. Of these, texture-layout turned out to be the most important. Texture-layout, in turn, performs a convolution operation (see Section 3.2 for more on the convolution operation) on the image. The convolution filters are learned as an ensemble of weak classifiers (ie. classifiers which correlate weakly with their predicted class). This process is known as boosting, and combined with convolutional operation it has similarities to the operation of convolutional neural networks, which are used in this thesis. Shotton et al. achieved a 73% accuracy on their validation set. They report biggest failure cases to be in classes with high visual variability and structured objects. Again, the accuracy seems to fall short of convolutional neural network's accuracy, although not by much. However, neural network methods' rise in popularity has resulted in various frameworks and off-the-shelf implementations, which makes their use easier.

2.3 Biomass estimation methods

Biomass estimation has been an active research field. However, there are multiple distinct ecological contexts, and the methods used in one are not necessarily applicable to another. Furthermore, meaningful methods vary depending on end goals. For example, there is a good amount of research into estimating lake cyanobacteria biomass, probably due to it being poisonous to humans. However, literature on calculating lake biomass based on flora is scarce. Here, few such methods - and some from closely related problem domains - are presented.

Lehmann et al. [44] presented a model which used lake *bathymetry* (lake floor topology), species composition and vegetation density to estimate the biomass of a mapped area. Their work was conducted on a 800m $*$ 800m area of Lake Geneva. Vegetation density and biomass were estimated through aerial photography and manual sampling, while the bathymetry was produced by using echo sounding, a type of sonar. Lehmann et al. calculate an estimate for the area by combining the bathymetry data with vegetation densities, distribution and biomass data. However, their method suffers from inaccuracies caused by successive calculations and is more suited towards qualitative than quantitative analysis. Furthermore, the system is more complex than one would hope.

Zhang [75] and Armstrong [6] used a multispectral camera to sample aquatic biomass. This technique relies on the reflectivity of submerged vegetation. Armstrong used GPS data to complement the multispectral image on biomass, while Zhang used a combination of principle component analysis to extract meaningful components of multispectral bands. Both used linear regression to estimate the biomass. Their results were promising, but the method used is not applicable to lakes over 3m deep. This would make the method inapplicable to many Finnish lakes. Furthermore, their method relies on mounting a multispectral camera on a boat, which is unwieldy in places with large amount of small lakes (such as Finland), and slow compared to using an unmanned aircraft system.

Additionally, a method based on satellite data has been used by Jachowski et al. [38]. They built a support vector machine for estimating mangrove biomass which achieved a correlation coefficient of 0.81. Their method used the elevation as well as the blue, green, red and near infrared bands of the satellite image. While their study site was a river mouth leading to ocean, and mangrove tends to behave in a different manner to lake flora, the results are impressive. However, it is unclear if their method can distinguish between different species of fauna.

Zweig et al. [77] used an unmanned aircraft system to take orthogonal images of wetlands. They then used The Feature Analyst [65] add-on for ArcGIS [19] to classify the imagery. The Feature Analyst [53] is a commercial product which uses an ensemble learning method with neural network, decision tree, clustering and bayesian learning components to extract features and recognize objects from orthogonal images. Using this combination, Zweig et al. achieved a 69% accuracy when attempting to distinguish between nine different classes, consisting of four different types of slough, three different types of sawgrass, emergent vegetation, and trees. When the classes were combined to a total of three classes, consisting of slough, emergent vegetation and sawgrass, the accuracy jumped to 91%. The results provide optimistic precedent for the work conducted in this thesis. However, they are from a slightly different ecological domain and the classification used is of higher abstraction. For example, this thesis is interested in classifying, among others, yellow water lilies, which are a member of the emergent class in Zweig et al.'s work. Moreover, the Feature Analyst add-on was not supplied for this thesis.

In this chapter, this thesis has given an overview of traditional image segmentation methods, non-neural network semantic segmentation methods as well as biomass estimation methods. Next, neural networks will be introduced. Following the breakthroughs made during the last few years by deep neural networks in image classification tasks, similar methodologies were utilized for segmentation tasks [10]. These approaches generally try to understand what is in the image, and where it is. Next, this thesis presents the general philosophy of neural networks and describes their structure. Special attention is paid to convolutional neural networks.

3. Neural networks

This chapter introduces neural networks in the context of computer vision. First, the philosophy of neural networks is discussed from both cognitive science and artificial intelligence viewpoints. Then, classical dense feedforward networks, activation functions and network training are examined. Finally, this chapter presents the theory behind convolutional neural networks.

Neural networks are a mode of computing that is based on connectionism [26], a cognitive science movement which tries to explain intelligence by borrowing ideas from the structure of the human brain. On a basic level, human cognition is formed by networks of neurons. A single neuron can have multiple inputs and multiple outputs. The neuron receives impulses from other neurons through its inputs. If a neuron receives a substantial enough impulse, it activates and sends the signal to all of its outputs, which in turn are the inputs to other neurons.

From artificial intelligence viewpoint, neural network models as discussed in this thesis are a method of supervised machine learning [31]. This has two implications: first, being a machine learning method implies that the system does not contain logic for how the model should operate, but it is given a set of instructions on how to learn the patterns required to succeed in a given task. Second, in supervised learning, the model is taught through examples. Each example consists of an (*input*, *output*)-tuple. For each input, the agent tries to guess the correct output. If the model outputs an incorrect prediction, it will be adjusted slightly in order to get better predictions. This mode of learning is contrasted by unsupervised learning, where the model is given only inputs but no right answers, and reinforcement learning where the model is given a problem space and is rewarded according to its performance in that space [31].

The canonical example of a neural network is the MNIST digit classifier. Here, the input is a 64x64 image of a handwritten digit from MNIST dataset, examples of which can be seen in Figure 3.1. The output is the network's guess on which number 0-9 a single image represents. While learning to describe written digits is in itself an achievement, this is nowadays considered an introduction to the field of neural networks.

Figure 3.1: 16 * 10 example inputs for MNIST classifier [39].

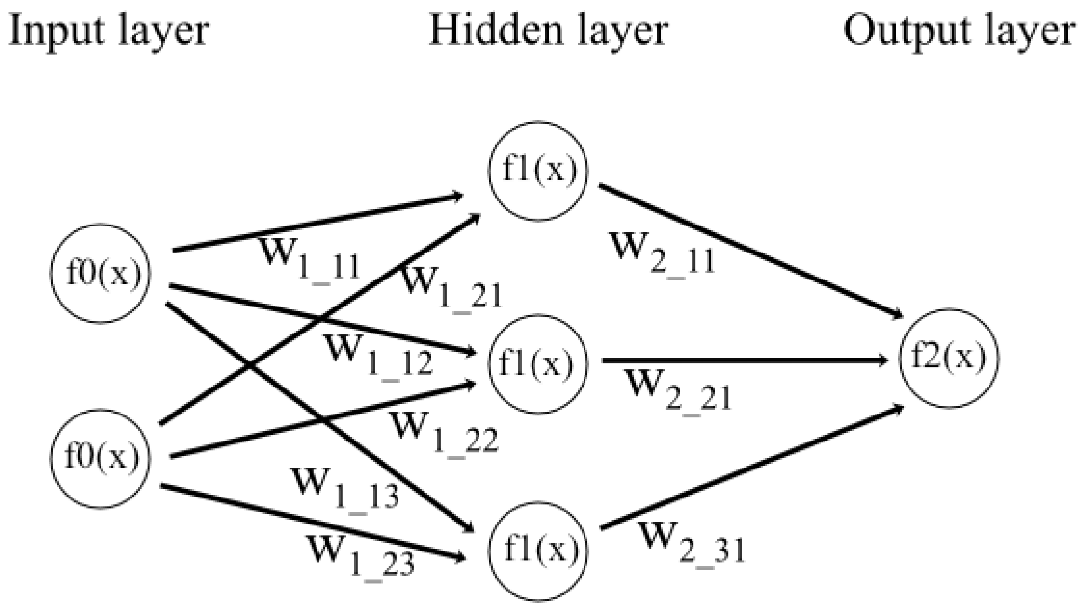


Figure 3.2: An example of dense feedforward network

3.1 Structure of Neural Networks

This section discusses the structure and workings of neural networks. First, the architecture of standard feedforward neural networks is presented. Then, activation functions are introduced. Finally, the thesis presents gradient descent as a way to update the network.

A neural network consists of neurons (nodes) arranged in layers [31]. Canonically, the first layer is called the *input layer*, the last layer the *output layer* and the layers between *hidden layers*. Each neuron has an activation function. Furthermore, each neuron has a set of incoming connections from and a set of outgoing connections to other neurons. A bias term is added to the incoming connections if a neuron; in the context of this thesis, it can be thought of as an additional incoming connection. Each connection has a weight associated with it. A neuron calculates its output by calculating the dot product of the values produced by incoming connections and the weights of incoming connections, taking a sum of the result and running the sum through the neuron's activation function.

An example image of a neural network is shown in Figure 3.2. Seen here is a high-level abstraction of a simple fully connected feedforward network with one input layer with two input neurons, one hidden layer with three neurons and an output layer with one neuron. This thesis will denote layers as $l^i = (l^1, l^2, ..., l^n)$, where n is the number of the layer, l^1 is the input layer and l^n is the output layer.

The set of inputs to the network is denoted as $\boldsymbol{x}$. Each individual input to the network is considered to be a vector $\bar{x} = (x_1, x_2, ..., x_n)$, and the output of the network a value y, with corresponding set of outputs $\boldsymbol{y}$. These are not found in Figure 3.2.

Each connection from a neuron to another has an attached weight. These are marked as $\mathbf{w}_{i\_j}$ in Figure 3.2, where i denotes the layer and j specifies which neurons are connected. Whenever the neuron receives an input, this input is multiplied by the weights (see equation 3.2). Bias terms are not depicted in Figure 3.2, but are marked in equation 3.2 as b_i. Bias terms work similar to normal inputs to neurons, and they have a weight associated with them.

After the input has been multiplied by the weights, the end result will be summed up and passed to the neuron's activation function, marked as $f(x)$ in Figure 3.2 . While activation functions in a layer tend to be the same function, they technically are not required to. The activation determines how strong impulse will be sent to all the neurons connected to this particular neuron. A classical choice for activation function is the logistic function, given in equation 3.3 and depicted in Figure 3.3. However, other activation functions, such as the Rectified Linear Unit (Figure 3.5 and equation 3.5) and tanh (Figure 3.4 and equation 3.4), exist.

As implied in the second paragraph of this chapter, a neural network needs a minimum of three layers: the input layer, which is responsible for loading the input to the network, a hidden layer, and an output layer, which is responsible for outputting the network's guess to user.

3.1.1 Dense feedforward neural networks

A *dense feedforward* network is a classical case of a neural network. Its definition is a combination of dense network and feedforward network [31]. In a dense network, each of the neurons in layer l^i is connected to each neuron in layer l^{i+1}, with the exception of the output layer, which is not connected to any other layer except the layer l^{n-1}. In a feedforward network, the computation only goes through the network in one direction. Essentially, this means that the computation flows from input layer to the first hidden layer, then in order through the rest of the hidden layers, until it reaches the output layer, which outputs the prediction. As such, it can be considered a directed acyclic graph - vis-à-vis a recurrent neural network [31], which contains cycles.

Altogether, the computation done by dense feedforward neural networks can be expressed as

$$y = f_1(f_2(...f_n(\bar{x})...)), \tag{3.1}$$

where y is the prediction generated by the network, $\bar{x}$ is the input vector and f_i corresponds to computation done by layer i, and is expressed as

$$f_i = a_i(\sum(\boldsymbol{w}_i(x_i^\frown b_i))), \tag{3.2}$$

where a_i is the activation function for layer i, $\boldsymbol{w_i}$ is the weight matrix and $x_i^\frown b_i$ is the input vector concatenated with a bias term.

As an example, the aforementioned MNIST classifier can be implemented as a dense feedforward network. Such models can perform admirably well, with a reported 0.35% error rate on a large model [14]. However, even relatively small, two-layer models have been reported [74] to achieve up to 0.7% error rates.

3.1.2 Activation functions

A number of activation functions exist [31]. This thesis presents three: logistic, tanh, and ReLU, all of which are non-linear. Using non-linear activation functions is crucial, because it enables the network to learn non-linear relationships. However, more complex activation functions result in more complex derivatives. As such, minimally non-linear functions are preferred as activation functions.

The formula for logistic activation function can be seen in equation 3.3, and it is plotted in Figure 3.3. It is closely related to tanh activation, which is given in equation 3.4, and plotted in Figure 3.4. Both logistic and tanh are examples of sigmoid functions, and both of them work well when x is near 0, but are slow to learn the further away from 0 x gets. This is due to the derivative of both these functions being very small when x is very large or very small.

$$logistic(x) = \frac{1}{1+e^{-x}}, \tag{3.3}$$

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \tag{3.4}$$

Rectified Linear Units, or ReLUs, have become the preferred activation function whenever they can be used. The formula for ReLU can be seen in equation 3.5, and plotted in Figure 3.5.

$$ReLU(x) = max(0, x) \tag{3.5}$$

As can be seen, ReLU introduces a minimal amount of non-linearity. This makes it easy to calculate the derivative of the loss function (also known as the gradient) which is discussed in chapter 3.1.3. The network used in this thesis uses ReLUs as activation function in all but the last layer, which uses a logistic activation.

Activation functions

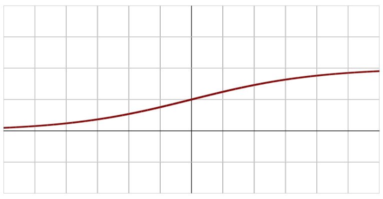


Figure 3.3: Logistic

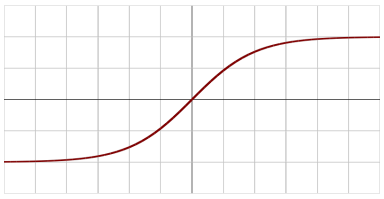


Figure 3.4: Tanh

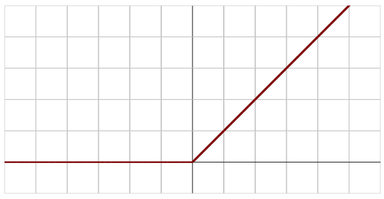


Figure 3.5: ReLU

All figures by Laughsinthestocks [43]

3.1.3 Training neural networks

As stated before, neural networks as discussed in this thesis are a supervised learning method [31]. As such, the training process involves a training dataset of (input, output)-tuples. On a high level, their training consist of initializing a network with random weights and then repeating the following procedure: first, the network tries to guess a correct output for given input. Then, if the guess is incorrect, the network weights are adjusted slightly. Finally, the network tries to guess the correct output for the next input, and so forth.

To measure the correctness of an output predicted by a network, a loss function is needed. There are multiple different cost functions, but a simple one is Mean Squared Error (MSE) [31], defined as

$$L(y, \hat{y}) = \frac{1}{2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.6}$$

where (y_i) is the correct output from training data, $\hat{y}_i$ is the network prediction and n is the amount of training samples. However, MSE is typically used with regression tasks, and this thesis is concerned with classification. As such, binary cross entropy [49], which is better suited for classification tasks, is used in this work. The equation for binary cross entropy is

$$L(y, \hat{y}) = -(y * log(\hat{y}_i) + (1 - y) * log(1 - \hat{y}_i)), \tag{3.7}$$

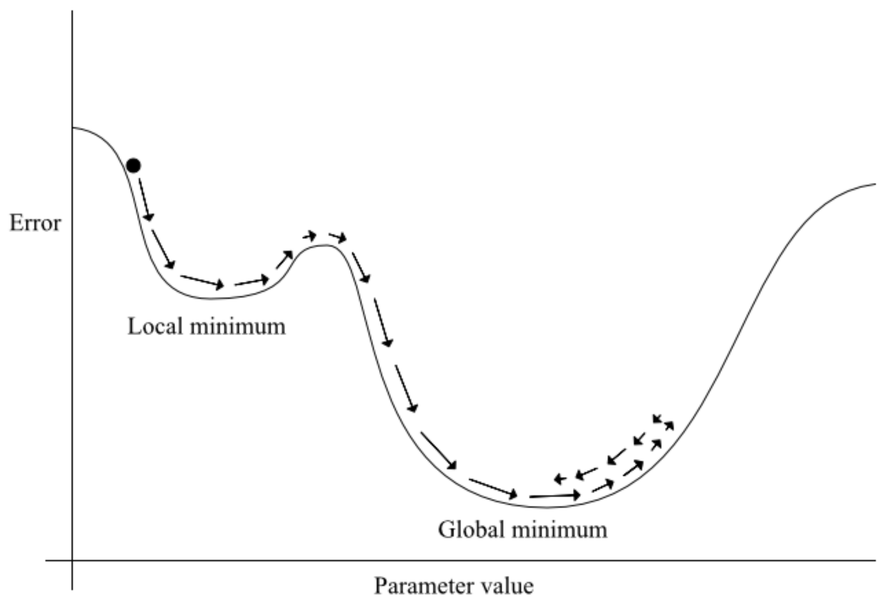


Figure 3.6: An example of finding global minimum through gradient descent.

where again y is the correct output and $\hat{y}$ is the model prediction.

Optimization

Once the error has been calculated, it is propagated backwards through the network [31]. On a high level of abstraction, the adjustment generated by this propagation can be formulated as

$$\Theta' = \Theta - \alpha \nabla_{\Theta} J(\Theta), \tag{3.8}$$

where Θ' is the next position, Θ is the current position, α is the *learning rate* (a small number), and $\nabla_x J(\Theta)$ is the vector for the adjustment. For a more in-depth presentation of these topics, see resources by Goodfellow et al. [31] or Nielsen [49].

In each neuron, a partial derivative with regards to each of its inputs and error is calculated. These partial gradients make up the vector $\nabla_x J(\Theta)$. The weight matrix and bias members are then adjusted according to the partial derivative and the learning rate, and the error is propagated further backwards through the network. This method is known as *gradient descent*, as calculating the derivatives results in a gradient of the cost function and the objective is to find the global minimum of the gradient; ie. the objective is to descend along the gradient to the global minimum. Note, however, that finding the global minimum nor a local minima of the gradient is not guaranteed by using gradient descent. A simple example of gradient descent can be seen in Figure 3.6. Here, y-axis ("Cost") is a function of x-axis ("Parameter value"). The black dot depicts the current situation, and the arrows the future steps of algorithm. The numerical evaluation of the gradient is made efficient by the use of backpropagation algorithm.

The learning rate is simply a hyperparameter dictating how large steps the network should take when adjusting the weights. In Figure 3.6, learning rate is depicted by the size of an single arrow. Large learning rates lead to the algorithm potentially finding a minimum faster, but may result in the minimum being inaccessible due to a valley in the gradient. In this situation, the learning will ”bounce” back and forth between the sides of the valley. In Figure 3.6, this could mean having a learning rate large enough that the steps taken (arrow size) would be long enough so that the training would bounce from one side of the global minimum valley to another. However, too small learning rate can get stuck in a local minimum, or cause the network training to slow down. To combat these problems, *momentum* can be used. Essentially, adding momentum increases learning rate when the gradient is steep or has recently been steep, and decreases learning rate when the gradient is gentle. In Figure 3.6, this can be seen as longer arrows during descent and shorter during ascent.

While gradient descent is a classical method for optimizing the network, more advanced methods exist. In particular, the work in this thesis uses an algorithm called Adam (Adaptive moment estimation) [41] to optimize the network. Adam is an extension on stochastic gradient descent, which in itself is a modification of gradient descent [31]. Gradient descent has a time complexity of $O(m)$, where m is the number of training examples. This m can become quite large, as large datasets are a requirement for proper learning in neural networks. Put together, these two conditions can make training times with regular gradient descent prohibitively long. In stochastic gradient descent, the gradient of the cost function is calculated by using an uniformly drawn subset of the data. This results in an estimate of the gradient of the whole dataset. However, the subset - often called a minibatch - is considerably smaller and does not grow if more data is added, which relaxes the computational time requirements.

In addition to minibatches, Adam combines the advantages of two previous algorithms: AdaGrad’s [18] ability to work on sparse gradients (gradients which contain very little information on how to adjust the parameters; imagine a gradient with very flat topology) and RMSProp’s [28] ability to work on moving objectives. Adam achieves this by using parameter (neuron)-specific learning rates. The adaption of the learning rates are based on moving average and uncentered variance of the data.

By repeating the forward propagation (network guessing a prediction for given input) and backwards propagation (adjusting weights and bias terms according to the derivative of the cost function) over and over, the network aims to minimize the result of the cost function [31]. Given properly chosen architecture, enough data and time, this results in the network learning how to solve the given problem.

There are various methods on how to enhance the training results of neural networks [31]. In this work, one basic enhancement is used: *dropout*. Essentially, dropout is a method for neural network regularization. As a highly robust and complex systems, neural networks have a tendency for *overfitting*. Put simply, overfitting means that the network has learned the peculiarities of its dataset too well, and does not behave well on a general case anymore. Regularization methods are aimed to combat overfitting. Dropout aims to achieve this by randomly shutting down a described proportion of neurons in the network. This forces each neuron to be able to work in a more generalized environment, because there are no guarantees it will receive inputs from the same set of neurons as before, nor can it rely on exactly the same set of future inputs on receiving its own signal. The benefits of dropout versus other regularization methods is that it is computationally cheap, applies to most of neural network models and doesn't interfere with the practicalities of training setup.

3.2 Convolutional Neural Networks

Convolutional neural networks, or ConvNets, or CNNs, are a special form of neural networks [31]. They are inspired by the research into how the mammalian visual cortex processes images, especially by the V1 part of the cortex. The crux of this research is that mammalian brains tend to respond the strongest to specific arrangements of light, such as horizontal bars. The convolution operation of CNNs take their inspiration from the simple cells of the primary visual cortex. These perform an operation that can be approximated as a linear function on their input. The *max-pooling operations*, in turn, are inspired by the complex cells in the primary visual cortex. These cells are similar to the simple cells with one exception: they aren't concerned with small shifts of the position or lighting of the detected feature. However, the max-pooling operations only concern positional, not illumination changes.

In the mammalian brain, the signals sent by the eyes are passed through various layers until they eventually hit the "gradmother cells". These are cells specialized in recognizing a concept, such as "grandmother" or "Emma Watson" - whether it is encountered as an image, audio, text, or so forth. This process results in a mammalian brain transforming the visual stimulus into a mental concept. In a philosophical sense, this process is what convolutional neural networks in image recognition tasks try to mimic - however, the models used are extremely simple compared to the mammalian visual cortex.

There are also differences between CNNs and mammalian vision. The human eye is very low resolution, except for the center of attention. The illusion of high resolution is created by *saccades*, ie. the eye moving around in fast bursts, and the subconscious rendering a single larger high-resolution mental image from these smaller images. However, CNNs can see the whole image in high-resolution in one go. Furthermore, human cognition uses all senses available, while CNNs are purely a visual tool. Third, the model used by convolutional neural networks is a highly simplified version of a brain, which has intricacies such as higher-level feedback and different activation/pooling methods.

From a practical standpoint, convolutional neural networks can be used whenever the data has spatial dependencies, ie. when it can be arranged in a n-dimensional grid. For example, a 2D image is a two-dimensional grid of pixels, or sound can be represented as a one-dimensional grid of pitch, sampled at specific intervals. Whenever a network has one or more *convolutional layer*, it is called a convolutional network. Their main difference to the dense feedforward networks is that instead of using matrix multiplication on all previous layer outputs, they use a convolution operation. The convolution operation is simply a sum of elementwise multiplication between two matrices. When performing convolution, the first of these matrices is a subsection of the input layer, and the second is the convolution *kernel*. The kernel K contains the kernel weights, and has to have same amount of dimensions as the input layer because due to the desire to do matrix multiplication between the two matrices. When the network is taught, it is these kernel weights that are being adjusted. In this thesis, the kernel has two dimensions because it is concerned with images, which are two-dimensional objects. The subsection of input layer is calculated by sliding a *convolution window* over the input layer. The size of this window is equal to the kernel size, and in each step of the convolution it creates a submatrix of input layer by selecting the contents inside the window. Because this process only gives a result for a small submatrix in the output layer, the convolution window is then moved to cover a different part of the input layer. A classical choice for this move is to move the window forward by its own size, ie. so that each element in the input layer is inside the convolution window in exactly one step. The size of this movement is called the *stride* of the convolution window. Essentially, for one-dimensional cases, the value of output layer element l_x^{i+1} can be calculated as

$$l_x^{i+1} = a(sum(I_{x*s} \circ K)),$$

where l_x^{i+1} denotes the x:th element of output layer, I_{x*s} denotes the submatrix of the input layer created by selecting input layer values from position x * *stride* to x * *stride* + *window size*, K is the kernel, $\circ$ is the elementwise multiplication operator, *sum* refers to calculating sum of matrix elements and a is a activation function. In multidimensional cases extra markup is required to denote the location of convolution window, but here the one-dimensional case is presented for simplicity. A classical choice for activation function in convolutional layers is the rectified linear unit (eq 3.5).

Examples of convolutional operations can be seen in figures 3.7 and 3.8. In Figure 3.7, a single step in the convolutional layer is shown. Here, a 3x3 kernel is applied into a single window in the input layer, resulting in a 1x1 output on the output layer. In Figure 3.8, the result of applying a 2x2 kernel with stride 2 on a 3x3 input is shown. The output can be seen on the right side of the figure. Note how the output layer is smaller than the input layer. If this is undesirable, the output layer can be padded to avoid layer shrinking. This padding can be achieved through addition of pixels around the image.

As a sidenote, in image context - which is of interest to this thesis - the kernel transformations can be thought of as image filters in the same sens as in photo editing software. For example blurring or detecting edges in input images can be expressed as a convolutional operation. Thus, the kernels are often called filters.

In dense networks, each node connects to every node in the previous layer. This means that even relatively simple networks often have a large number of parameters, which leads to long training times. Convolutional layers only train the kernel weights, which means that there are less parameters to train. This leads to a significantly smaller parameter size for convolutional layers. However, it is common for ConvNets to train multiple different kernels for a single layer.

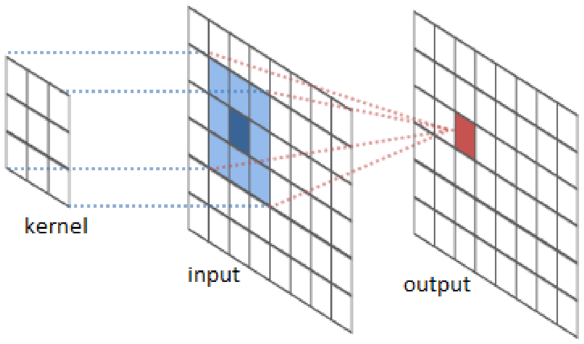
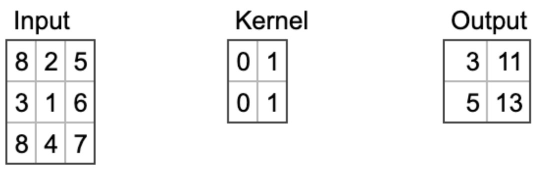


Figure 3.7: An example of kernel applied to input [36]


Input

8	2	5
3	1	6
8	4	7

Kernel

0	1
0	1

Output

3	11
5	13

Figure 3.8: A numerical example of a 2x2 kernel applied to layer.

Typically in convolutional neural networks, a convolution layer is paired with a pooling layer. A pooling layer slides a window over the input grid and calculates a summary statistic - such as max value or average - on it. Similar to a convolutional

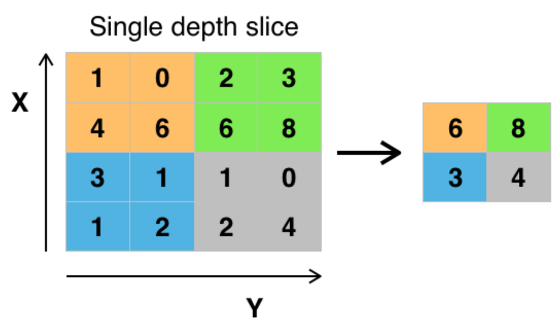


Figure 3.9: An example of 2x2, 2-stride max pooling [5].

kernel, a pooling operation has a window size and a stride.

An example on max pooling can be seen in Figure 3.9. Here, 2x2 max pooling with stride of 2 is applied to a 4x4 input, resulting in a 2x2 output. Thus, the window is first placed to the top-left (orange) quadrant, and the maximum value (hence, max-pooling) out of the four is calculated. This value (6) is then inserted to the top-left cell of the result matrix. Then, due to stride two, the window is moved two steps to the right, which lands it in the quadrant coloured green. This process is repeated until the input matrix has been gone through. The pooling layers have the effects of making the network less vulnerable to noise in the data and downsampling the problem space, which means that smaller layers are required in later parts of the network.

The main advantage that ConvNets have over dense feedforward networks is that they require less parameters to train. A dense feedforward network would use one parameter in each layer for each pixel in the image. However, ConvNets' window size govern their parameter size - for example, a 2x2 window results in $2 * 2 = 4$ parameters for a single layer. Furthermore, because the pooling layers shrink images, they are more invariant towards small variance. ConvNets can also handle variable input sizes. In addition to these theoretical features, ConvNets have been shown to perform well on, for example, image classification [42] and image segmentation [10].

In this chapter, this thesis has presented neural networks. It introduced the classical dense feedforward network architecture as well as the convolutional neural network architecture. It then presented a comparison between the two. Furthermore, different activation functions and optimization strategies were discussed. In addition to traditional neurons, convolution operation as well as pooling operations were introduced. In the next chapter, the thesis returns into the domain problem of doing image segmentation through convolutional neural networks.

4. Segmentation with neural networks

Semantic segmentation done by neural networks is a well-researched problem with multiple published papers. In this chapter, three architectures pertaining to doing semantic segmentation with convolutional neural networks will be presented. These are the fully convolutional neural network by Long et al. [46], the Mask R-CNN architecture by He et al. [34] and the U-net architecture by Ronneberger et al. [60].

4.1 Fully convolutional network

Long et al. [46] introduced fully convolutional networks (FCNs, networks with only convolutional layers and pooling operations) as a method to do semantic segmentation with pixel-to-pixel resolution. The paper presents a transfer-learned, fully convolutional network coupled with skip architecture. Long et al.'s network accomplishes pixel-level segmentation by taking an existing image classification network, such as AlexNet [42], and modifying it. The modifications include interpreting all dense layers as convolutional layers, as well as replacing the final layer with upsampling. Interpreting dense layers as convolutional layers is done by considering each dense layer as a convolutional layer with kernel size equal to image size. Furthermore, the final prediction layer of the network is discarded, and a convolutional layer is added to produce predictions for the layer. This prediction is then upsampled through deconvolution, which is convolution operation applied backwards: a single pixel is multiplied by window values, and these values are the upsampled pixel values.

Furthermore, Long et al. make use of a skip architecture. Skip architecture means that layers have connections to layers farther down the network than just the next layer. Skip architecture helps in producing fine-grained predictions by letting the network remember details about the fine-detail layers at the first levels of computation.

Long et al. report training time of up to four days when converting from existing classification network, however this was in 2015 and on a single ”typical” GPU, so smaller numbers could be expected on modern environments. The actual GPU used is not reported. Their model does inference relatively fast, in 100ms, again with a single GPU.

Long et al. [46] achieved good results with their architecture, which are displayed in Figure 4.1. Here, VOC2011 [22] and VOC2012 [23] test sets refer to Pascal VOC datasets, which is an image dataset of 20 mundane classes such as ”cats” and ”airplanes”. Mean IU score compares the model prediction to the ground truth annotation and reduces score for both false positive and false negative values. Hence, bigger score in IU is better. Long et al.'s results were, at the time, better than the competition's, and their model's inference time was faster.

Table 4.1: Results reported by Long et al. [46] on the semantic segmentation performance on PASCAL VOC. FCN-8 is their network, while SDS and R-CNN are previous well-performing networks

	mean IU VOC2011 test	mean IU VOC2012 test	inference time
R-CNN (2014)	47.9	-	-
SDS (2014)	52.6	51.6	50s
FNC-8s (2014)	67.5	67.2	100ms

4.2 Mask R-CNN

Mask R-CNN, introduced by He et al. [34], uses a multi-tiered architecture to produce both object instance segmentation as well as bounding boxes. Object instance segmentation is slightly different problem than semantic segmentation: in semantic segmentation, every instance of ”car” belongs to the segment corresponding to the class ”car”. In instance segmentation, each instance of ”car” belongs to its own segment. However, the method is more recent and could be used to produce a semantic segmentation, so it is introduced here.

Mask R-CNN consist of four parts: a feature extractor, a region proposal network which scans the image for objects, a region classifier and a mask network. The features can be extracted by a standard convolutional neural network. However, He et al. suggest the use of *feature pyramid networks* [45]. Feature pyramid networks consist of three parts: bottom-up pathway, top-down pathway and lateral connections. Bottom-up pathway is a series of convolutional operations, generating smaller-resolution features from the original image. Top-down pathway takes the result of the top-down pathway and generates higher-resolution features from it with the help of lateral connections and the convolution operation.

The region proposal network [58] uses a convolutional neural network to generate regions of interest. These regions are generated by the use of anchor boxes. Essentially,

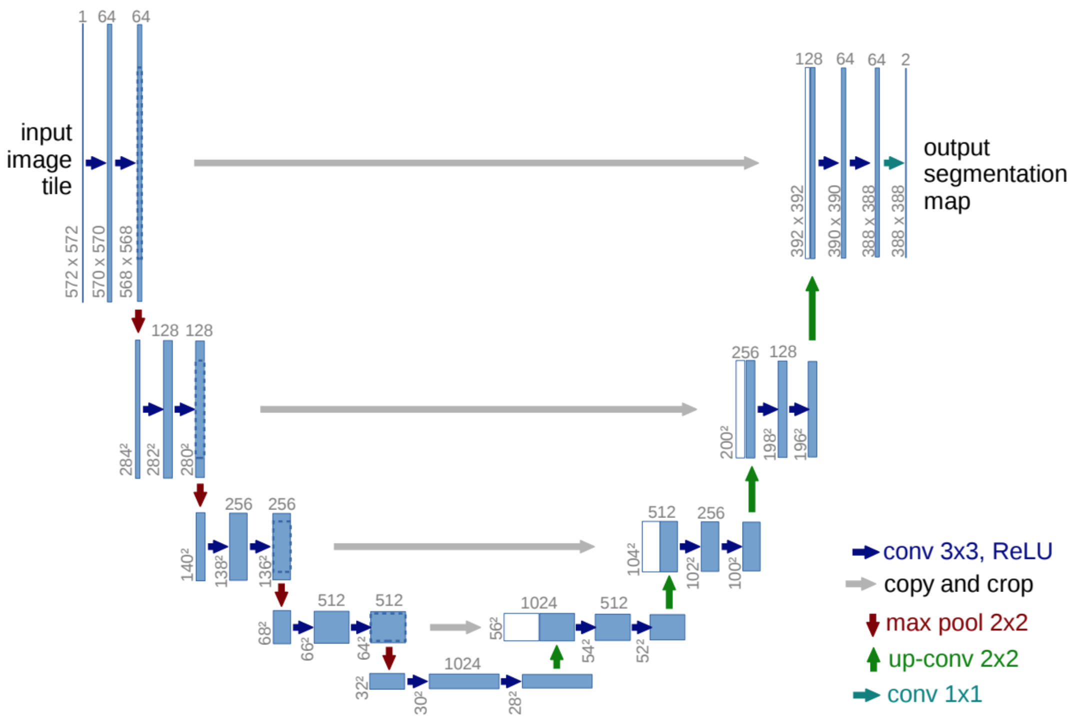


Figure 4.1: U-net architecture [60]. The blue boxes are layers, the number above them depicts the number of specified layer and the pair of numbers to the left of the box the size of the layer. Arrows depict operations, such as 3x3 convolution (blue), max pool (red), concatenation(grey), upwards convolution (green) and 1x1 convolution (teal).

a set of boxes are distributed in the image, each box is checked, and if it contains an object, it is then adjusted to better fit the object. To speed up the computation, these regions are then pooled. The pooled regions are then passed to the region classifier, which refines the bounding box and assigns a label to it via the use of convolutional neural networks. A FCN [46] (see Section 4.1) is run parallel to the region classifier to produce the segmentation on the images.

Mask R-CNN took 32 hours to train on a COCO train set and an 8-GPU environment, and does inference in 210ms on a Nvidia Tesla M40 GPU [51]. It is safe to say that Mask R-CNN is a heavier model than the FCN, but the train and the inference times will not be a hindrance for using the model. Similar to VOC challenges [20][21][22][23], COCO segmentation challenge [15][16] has images containing everyday objects from 90 classes. He et al. achieve good results on segmentation, reporting beating winners of 2015 and 2016 COCO segmentation challenges.

4.3 U-net

U-net, by Ronneberger et al. [60] is an improvement on Long et al.'s work presented above. U-net improves on Long et al. by implementing multiple deconvolutions (resulting in a the implementation's namesake U-shaped network) and increasing the amount of features used in skip connections. These deconvolution layers are the steps that separate U-net from traditional CNNs such as the one presented by Long et al. Their function is to create a high-resolution map based on the lower resolution maps produced by the previous layers of deconvolutions. The architecture is displayed in detail in Figure 4.1. It can be seen that in the left or deconvolution part of the network there are repeated applications of two 3x3 unpadded convolutions followed by 2x2 stride 2 max pool operation. The same is true for all but the final convolution operations on the upsampling or right part of the network, with max pool replaced by deconvolution layers. Finally, the last layer is a 1x1 convolution which predicts a class for each pixel in the image. Ronneberger et al. use ReLU (see eq. 3.5) as their activation function.

Ronneberger et al. use weighted loss function to force their network to learn cell borders. Essentially, they have created a loss map for their training data which punishes mistakes in classifying cell borders more than otherwise. In this thesis, no weighted loss functions were used due to the fact that there is no need to be able to distinguish between single instances of classes. As this thesis is mainly interested in how much target classes there exists in the pictures, the amount of work required to produce the weighted loss maps is not justifiable.

Ronneberger et al. stress the importance of data augmentation: that is, producing more training data by applying transformations to original training dataset in order to increase the amount of training dataset. They also note that this works well in their domain (medical imaging), because realistic-looking transformations to their medical data are relatively easy. These transformations include shifting, rotating and deforming the image as well as varying the gray levels. However, augmenting data by style transfer has also been shown to work well on image classification [27]. As such, it might be safe to assume that data augmentation could work well for other segmentation tasks besides medical imaging.

Ronneberger et al. report training times of 10 hours on a single 6GB Nvidia Titan GPU [52]. They do not provide inference times. Ronneberger et al. claim good results, their U-net architecture achieving significantly better results (table 4.2) in ISBI Cell Tracking Challenge [37] than the next-best try. The IOU (intersect over union) metric used in the competition is defined as follows:

$$IOU = \frac{A \cap B}{A \cup B}, \qquad (4.1)$$

where A is a set of the predicted pixels for a class and the B is the set of actual pixels belonging to that class. Here the maximum score is 1 and the minimum is 0. Essentially, IOU is a metric of how well the predicted area overlaps with the actual area. U-net's results in EM segmentation challenge (table 4.3) are good as well, with the model only losing to competition in on case - DIVE-SCI achieved a better Rand error value. Rand error is calculated [57] as

$$1 - \frac{a+b}{\binom{n}{2}}, \tag{4.2}$$

where a is the number of pair of pixels in the image which are correctly classified to the same class in both the prediction and ground truth, and b is the number of pixel pairs in the image which were correctly classified differently. The authors explain their loss as the difference between their more general model taking on an algorithm specifically designed for the dataset.

Table 4.2: U-net IOU results on ISBI Cell Tracking Challenge 2015, two different image datasets [60]

Name	PhC-U373	DIC-HeLa
IMCB-SG (2014)	0.2669	0.2935
KTH-SE (2014)	0.7953	0.4607
HOUS-US (2014)	0.5325	-
second-best[sic] (2015)	0.83	0.46
U-net (2015)	0.9203	0.7756

Table 4.3: U-net results on EM segmentation challenge [60]. Lower is better.

Rank	Group name	Warping Error	Rand error	Pixel error
	** human values **	0.000005	0.0021	0.0010
1	u-net	0.000353	0.0382	0.0611
2	DIVE-SCI	0.000355	0.0305	0.0584
3	IDSIA [1]	0.000420	0.0504	0.0613
4	DIVE	0.000430	0.0545	0.0582
...				
10	IDSIA-SCI	0.000653	0.0189	0.1027

In this chapter, three methods for doing semantic segmentation with convolutional neural networks were given. These were the fully convolutional neural network by Long et al. [46], the Mask R-CNN by [34] and the U-net, which is an improvement on U-net, by Ronneberger et al. While these networks do not directly deal with the problem domain of biomass estimation or recognizing alien invasive species, their results are promising. It is intuitively hard to see clear differences in, for example, cell detection versus marine plant detection. As such, this thesis will next present applying U-net architecture to the two problem domains: discovering alien invasive species and biomass estimation. The U-net was chosen as it was deemed to potentially achieve a good result with a simpler architecture when compared to Mask R-CNN. Furthermore, in the context of this thesis, semantic segmentation is enough; object instance segmentation is not needed.

5. Using U-net to segment drone images

In the previous chapter, an overview of semantic segmentation was given. Furthermore, U-net was introduced as a valid choice for network architecture for semantic segmentation. In this chapter, an implementation of U-net for the Finnish Environment Institute (SYKE) is presented. This chapter begins by broadly describing the specifics of the context and the approach taken to solve the problems of biomass estimation and invasive alien species detection. It continues by giving a description of the dataset used, discusses the potential problems of the data, as well as presents learning points for future work. It then goes over the model training step, from image preprocessing to the specifics of the training process. Finally, results are given.

As described in Chapter 1, the Finnish Environment Institute has two goals in mind in the context of this project: to monitor the amounts of invasive plant species found in Finland, and to estimate lake biomass. Both of these missions have traditionally required SYKE to monitor large areas. The naive solutions to these problems are described in Chapter 1. These approaches have two downsides: scaling and inefficient use of resources. If the government wants to monitor larger areas for biomass or IAS, traditional methods require more personnel in a linear fashion. The personnel doing this kind of a job typically have a university degree, quickly making the cost of the monitoring task prohibitive. Both of these activities tie up expert resources to a fairly trivial task. Thus, a more efficient method for solving the two problems is needed.

To be able to achieve this goal, this thesis explores the usage of semantic segmentation produced by an U-net in combination of images taken by drones. To produce the required dataset, SYKE has purchased drone photograph services from BVDrone [8] which photographed image datasets from both land and lake areas. These drones take pictures of the respective areas, which are then fed into a neural network. The neural network should produce a mask based on existence of each invasive species. In this thesis, a U-net (see chapter 4) will be used to produce a semantic segmentation of the images, which can be then used to estimate lake biomass as well as to locate alien invasive species. Next, the specifics of the used model as well as the details of the

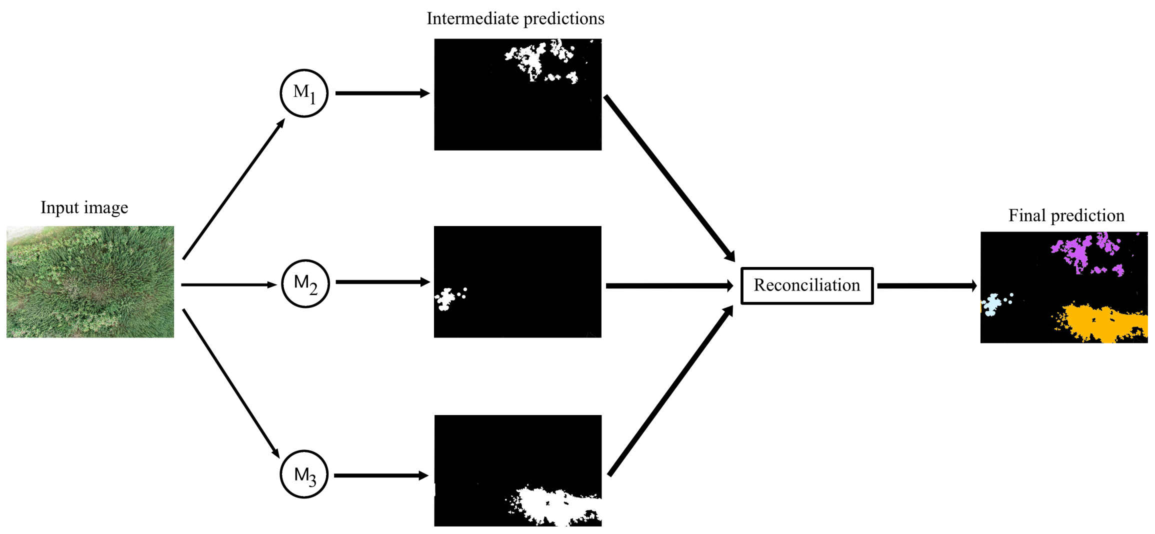


Figure 5.1: Monoclass model prediction pipeline

training process will be discussed.

There are two possible ways to approach building the models. The first is to train a model to produce multiclass predictions. The second is to train one model for each class (ie. a binary classifier), run prediction task for single image through each of the models, and then use a reconciliation function to produce a final prediction. This thesis uses the second method, as this allows each model to concentrate on one specific class, making learning easier.

In the multiclass prediction task, the model outputs a probability distribution over the classes for each pixel. The prediction image is then produced by mapping through each pixel, selecting the RGB value of class with highest probability. In the training phase, conversion to RGB can be omitted by precomputing one-hot vectors out of ground truth images.

The second method is to train a model for each class in the dataset. This method is depicted in Figure 5.1. In this approach, prediction is done by running an input image through each model, which produces n predictions, where n is the number of classes. Each of these predictions is a grayscale image. The colour black corresponds to probability of 0 for a given pixel to belong to the class predicted by the model. Correspondingly the colour white corresponds to a probability of 1. Each pixel then contains the probability of that pixel to belong to the class predicted by the class's model. Once n predictions of an image have been produced (i.e. each model has produced a prediction), the predictions are reconciled by a reconciliation function. Typically, the reconciliation function goes through the n intermediary predictions and chooses for each pixel in the final prediction the colour of the most probable class (i.e.

	Profile 1	Profile 2	Profile 3
Focal length	8.8mm	8.8mm	8.8mm
Exposure	1/100s	1/1000s	1/160
Focal ratio	f/4	f/2.8	f/4.5
Resolution	5472x3648	5472x3648	5472x3648
Image format	PNG	PNG	PNG
Colour space	RGB	RGB	RGB
Colour profile	sRGB IEC61966-2.1	sRGB IEC61966-2.1	sRGB IEC61966-2.1

Table 5.1: Camera setting profiles for drone cameras

the class whose model has scored highest) from the prediction images. Imagewise, this would mean the class whose model produced the whitest pixel in the intermediary predictions. In this thesis, binary classifiers were used due to skewness of the datasets (see Figures 5.2 through 5.17).

5.1 Methodology

As mentioned before, the models were trained with datasets provided by SYKE. In this section, the dataset and its problems are discussed. Then, a description of data preprocessing as well as training steps are given.

5.1.1 Data

The dataset consists of three different categories of data. The first are the actual images taken by drones. Whenever ”input images” are mentioned in this chapter, it is a reference to these imges taken by a drone. The second are ground truth images created by experts at SYKE based on the drone images. These will be known as ”ground truth images”. The third are the category files for each ground truth image. The category files contain a mapping from colour to the class for given image, as the colour used for a single class might vary from image to image.

The dataset used for biomass estimation task consists of three separate DJI Phantom 4 Pro drone runs over Onkivesi lake in Finland, while the dataset for invasive alien species dataset has been collected from two runs near Hollola, Finland and one run at Ojaküla, Estonia. Two different profiles for the camera were used. These profiles are listed in table 5.1. Two of the Onkivesi runs used profile 1 and one used profile 2. Ojaküla run used profile 2. One of the Hollola runs used profile 2 while the other used profile 3.

The runs used different flying altitudes. Initially this was to figure out the best altitude for prediction task. Flying higher would be faster as the camera could see more at once, but at higher altitudes the details of single plants would become hard to distinguish, even to the point where human experts would not be sure which plants they were looking at. In the end, all altitudes where human experts were able to make positive identifications were used to create a more robust model.

In both of the tasks the datasets are heavily skewed towards ”out of research interest”-class. Class distributions in the datasets are given in Figures 5.2 through 5.17. Furthermore, class distributions are given in tables 5.2 for biomass estimation data and 5.3 for alien invasive species data. For example, table 5.2 tells us that of the total 19448 images in biomass estimation task, *Nuphar lutea* appears in 2404, or 12.37% of the images. Furthermore, as can be seen in Figure 5.10, the distribution of images with *Nuphar lutea* is highly skewed: even when *Nuphar lutea* appears in a image, it is probable that it covers a small portion of said image. The Y axis in the figure displays a number of teaching images (512x512 tiles of the original), and the X-axis indicates how many percent of that image had *Nuphar lutea* in it. Images without *Nuphar lutea* are around two orders of magnitude more common than those that have at least 1% of *Nuphar lutea* in them. Because of this restriction, single-class models trained on the whole dataset tended to learn to predict that there were no instances of their class anywhere in the pictures. Thus, a restricted dataset had to be used. This dataset was created by examining the original dataset and keeping any 512x512 tiles which had at least one pixel of target class in it.

The dataset has following limitations. First of all, having only images which include at least a little bit of the target class in the image raised the worry that the model could become biased. However, due to the abundance of examples of ”not-in-class” in the training set, this was not considered a serious issue. As can be seen from Figures 5.2 through 5.17, even the images which contain target class typically have a good amount of examples of ”not-in-class”. Indeed, when the *Nuphar lutea* model was trained with around 5% of the training set consisting of entirely out-of-research-interest images, the model training would often fail catastrophically and halt in an early stop just after a couple of epochs of training. Second, the dataset is already small (223 full images), and this problem is further exacerbated by cutting the dataset to include only tiles with the target class in them. For example, while the total number of images in biomass estimation dataset is 19448 (512x512 pixel tiles), only 2434 (12,5%) of them were images which had at least one pixel classified as *Nuphar lutea.* The amount of images used to train the models would further drop due to the usage of 3-fold cross-validation: out of the 2434, $\frac{2}{3} = 1622$ would be used to train the model, while the rest were used for evaluation. Yet a third potential problem is the annotation of the data. The dataset was annotated by experts at SYKE, who have done mostly admirable

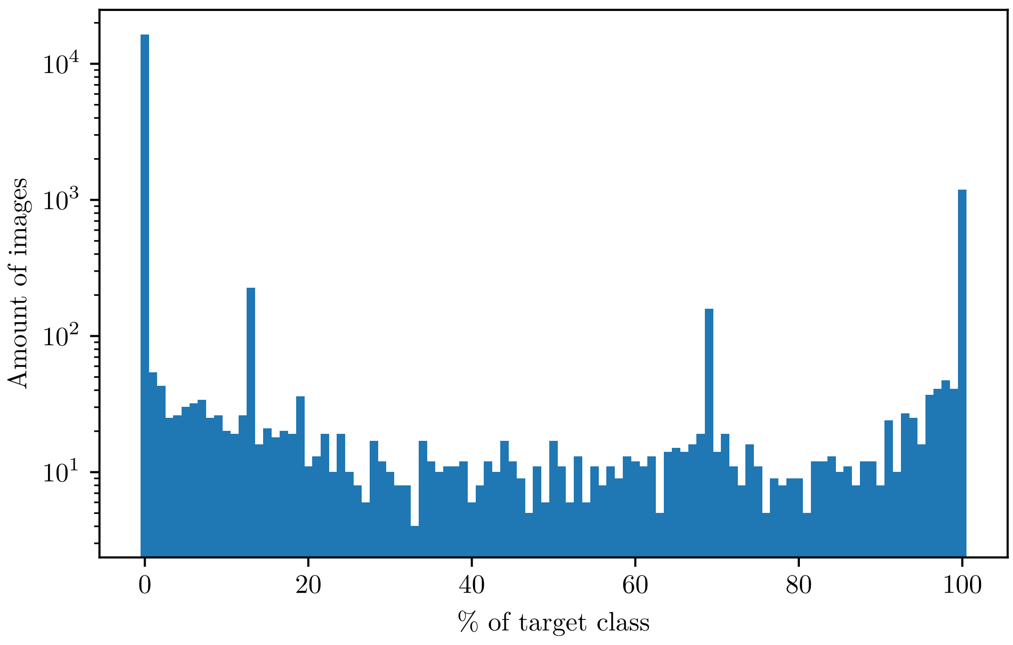


Figure 5.2: Distribution of *Phragmites australis* in the dataset (log scale)

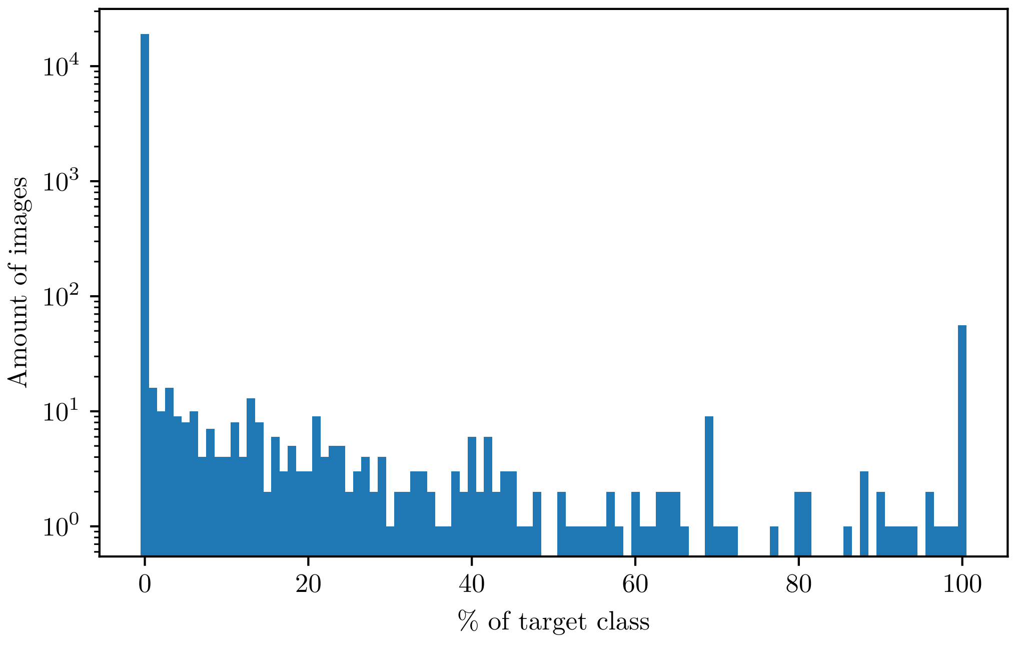


Figure 5.3: Distribution of *Scripus* in the dataset (log scale)

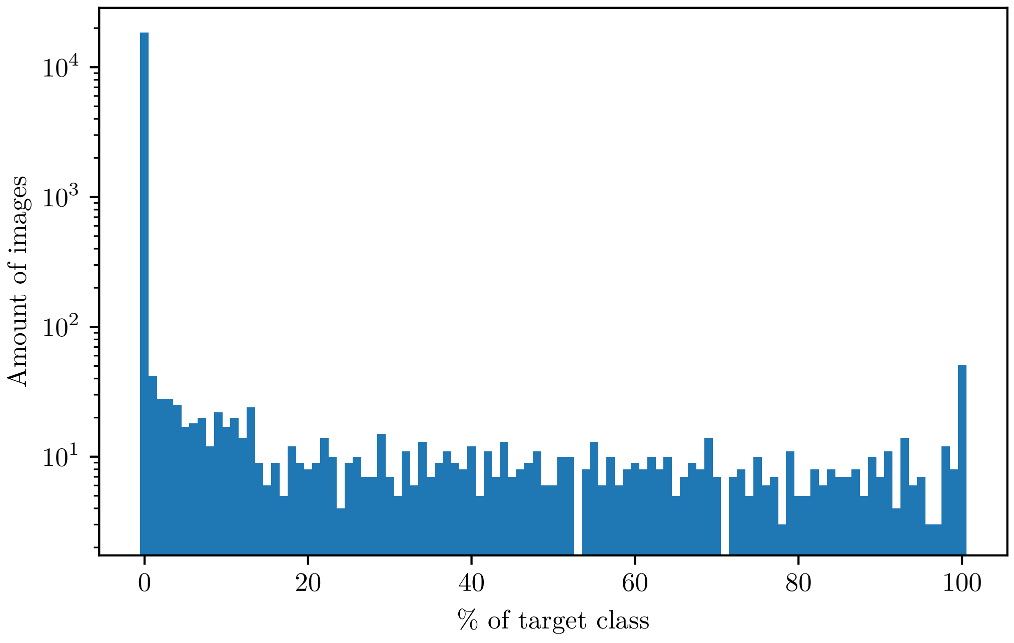


Figure 5.4: Distribution of *Sagittaria natans* in the dataset (log scale)

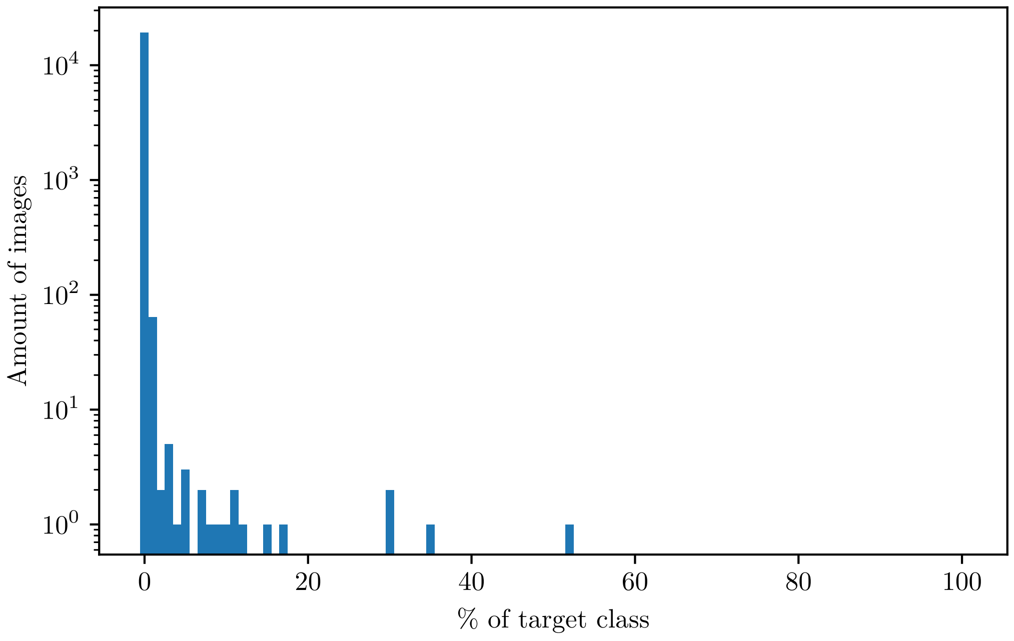


Figure 5.5: Distribution of *Hydrocharis morsus-ranae* in the dataset (log scale)

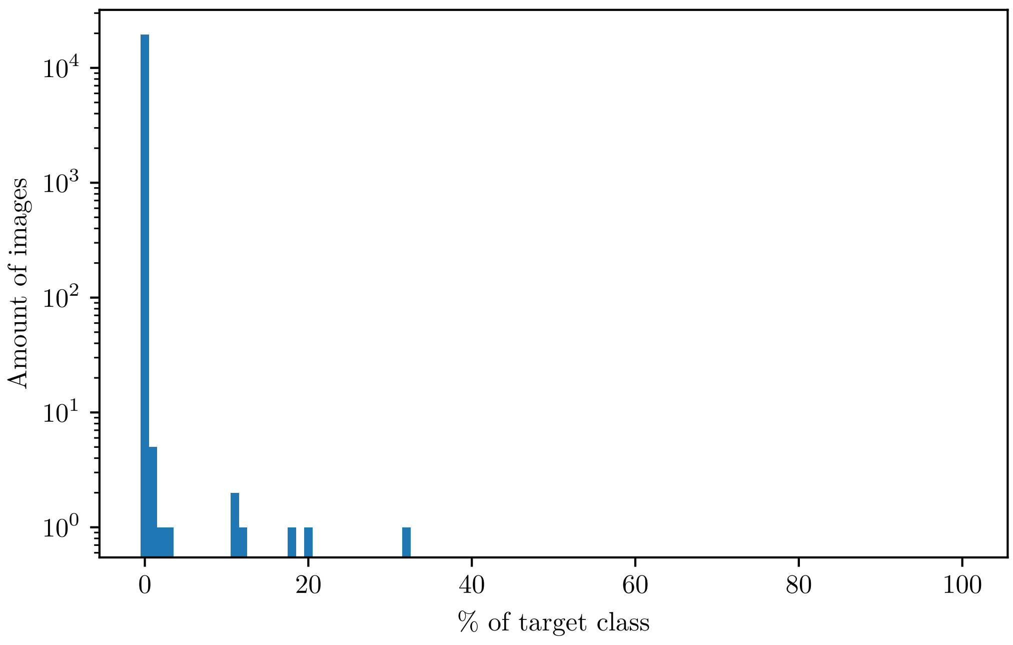


Figure 5.6: Distribution of *Sagittaria sagittifolia* in the dataset (log scale)

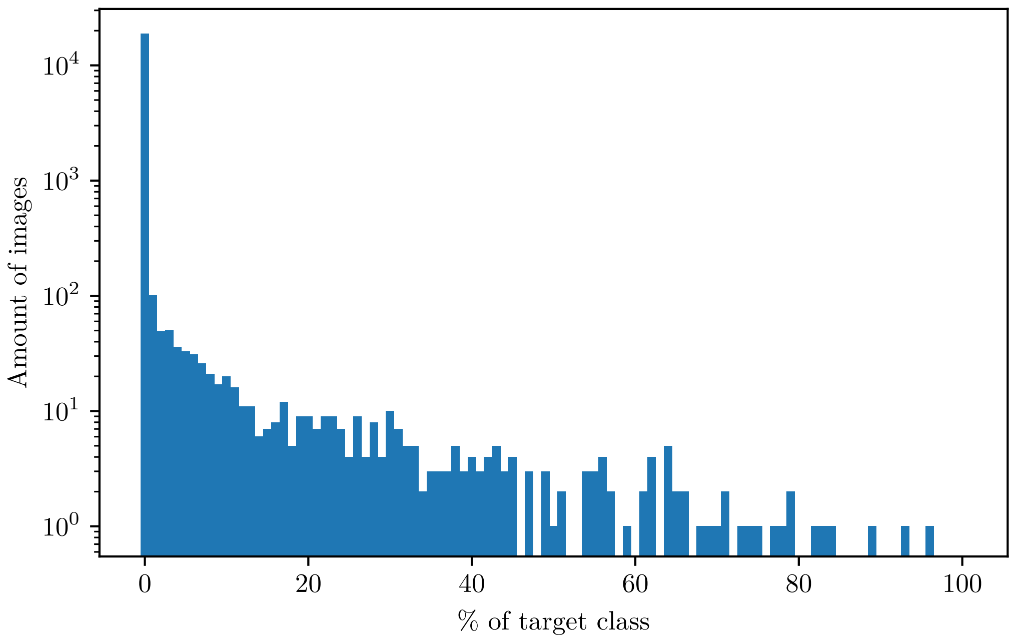


Figure 5.7: Distribution of *Sparganium emersum* in the dataset (log scale)

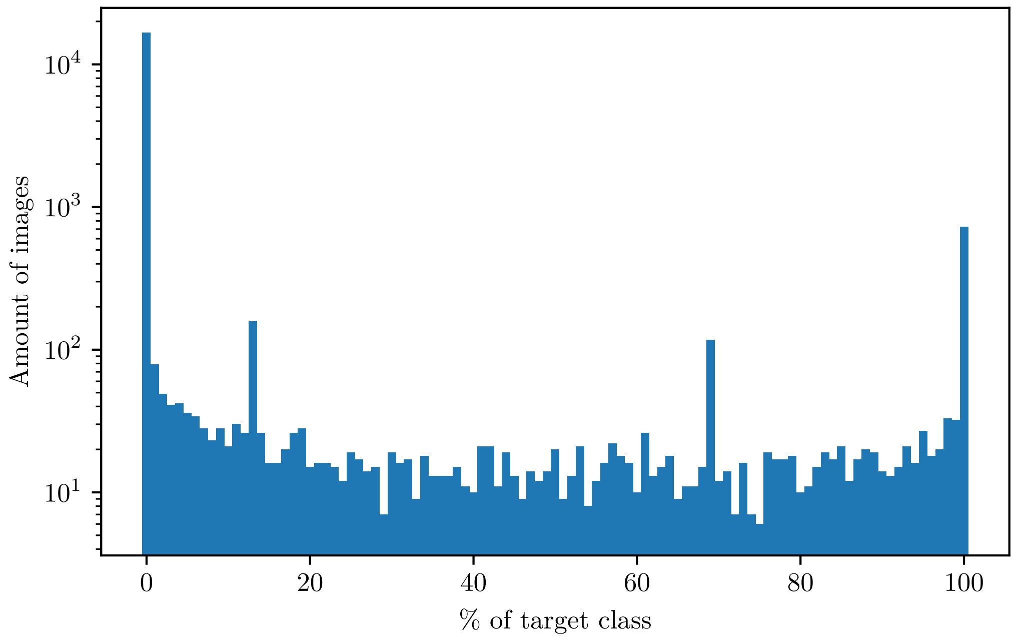


Figure 5.8: Distribution of *Sparganium gramineum* in the dataset (log scale)

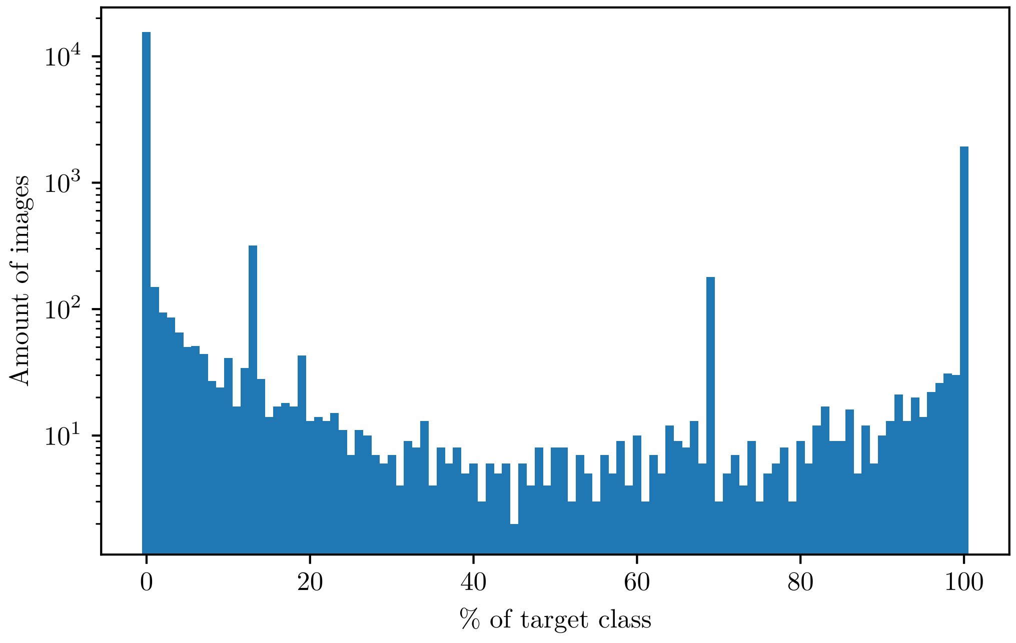


Figure 5.9: Distribution of *Potamogeton natans* in the dataset (log scale)

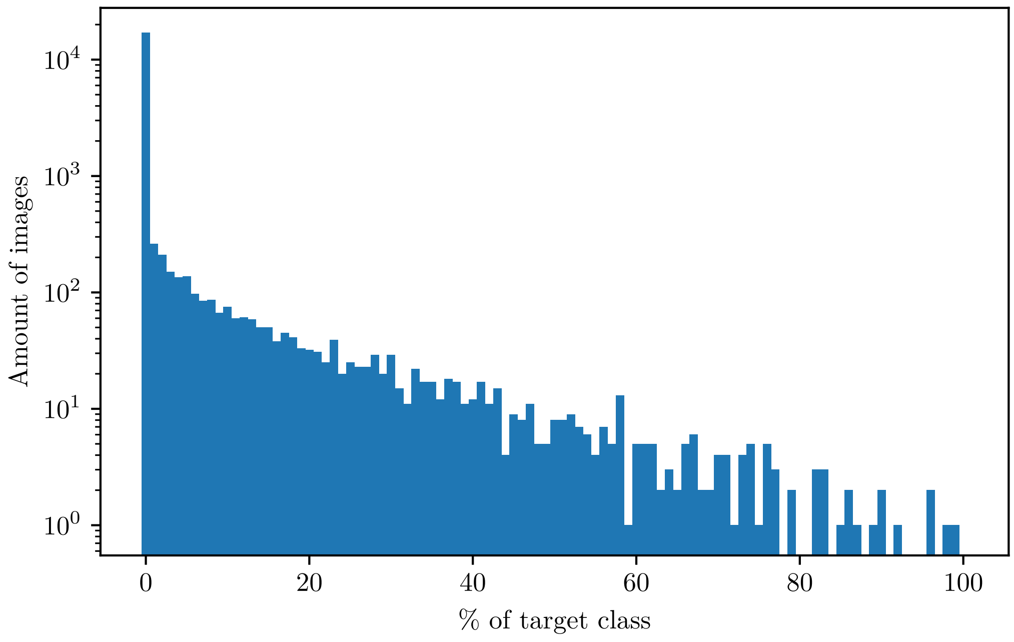


Figure 5.10: Distribution of *Nuphar lutea* in the dataset (log scale)

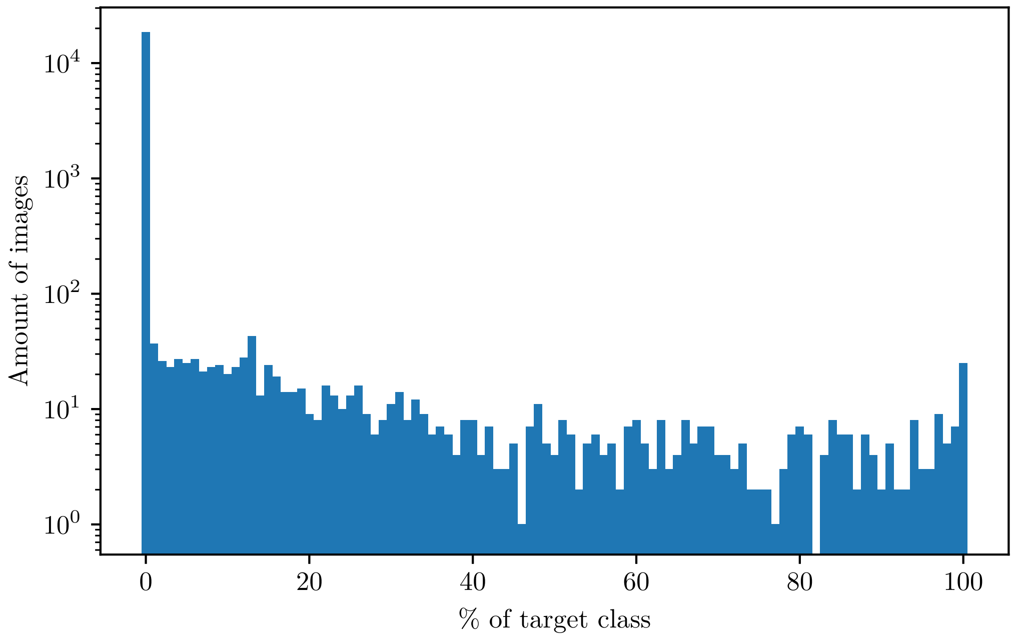


Figure 5.11: Distribution of *Persicaria amphibia* in the dataset (log scale)

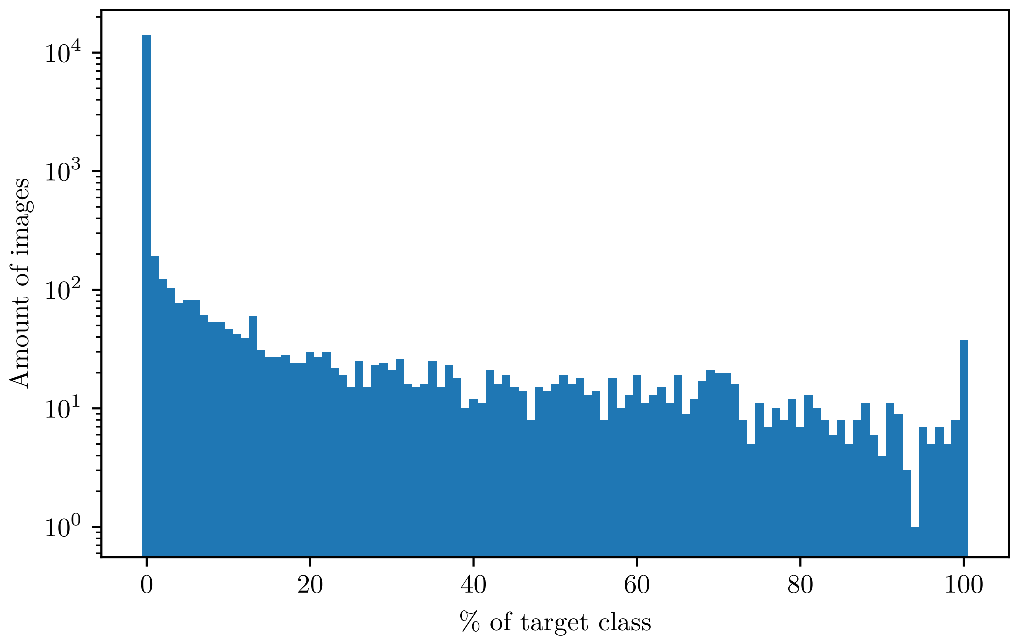


Figure 5.12: Distribution of *Impatiens glandulifera* in the dataset (log scale)

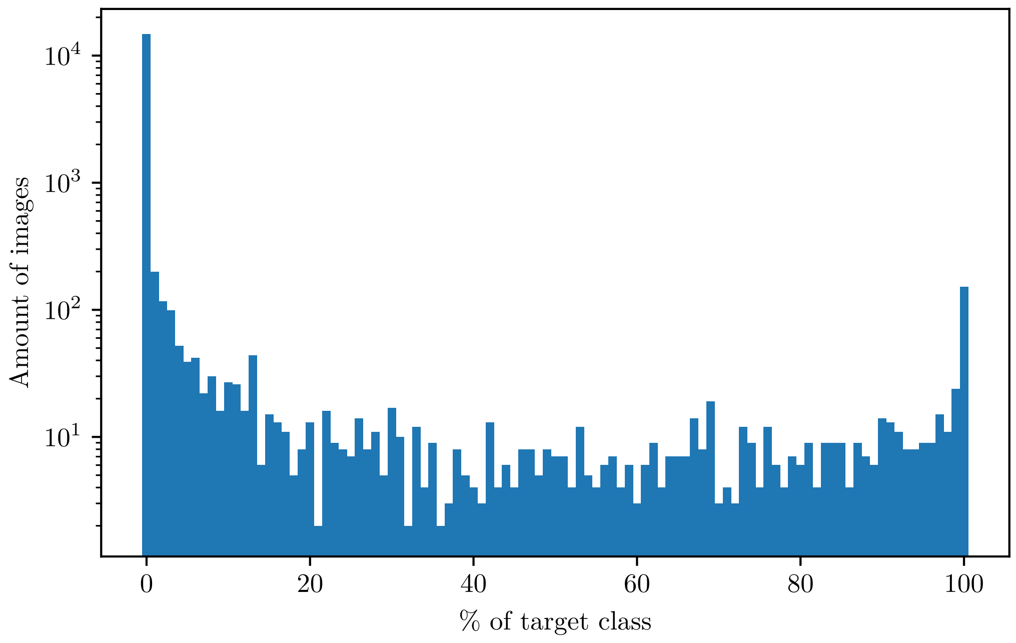


Figure 5.13: Distribution of *Heracleum persicum* in the dataset (log scale)

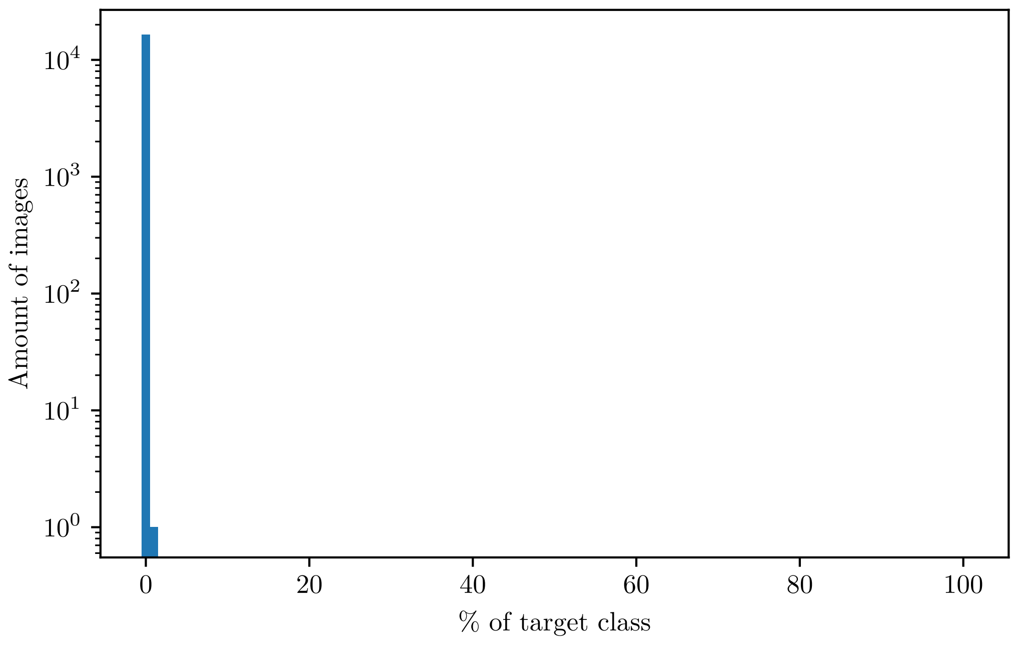


Figure 5.14: Distribution of *Solidago canadensis* in the dataset (log scale)

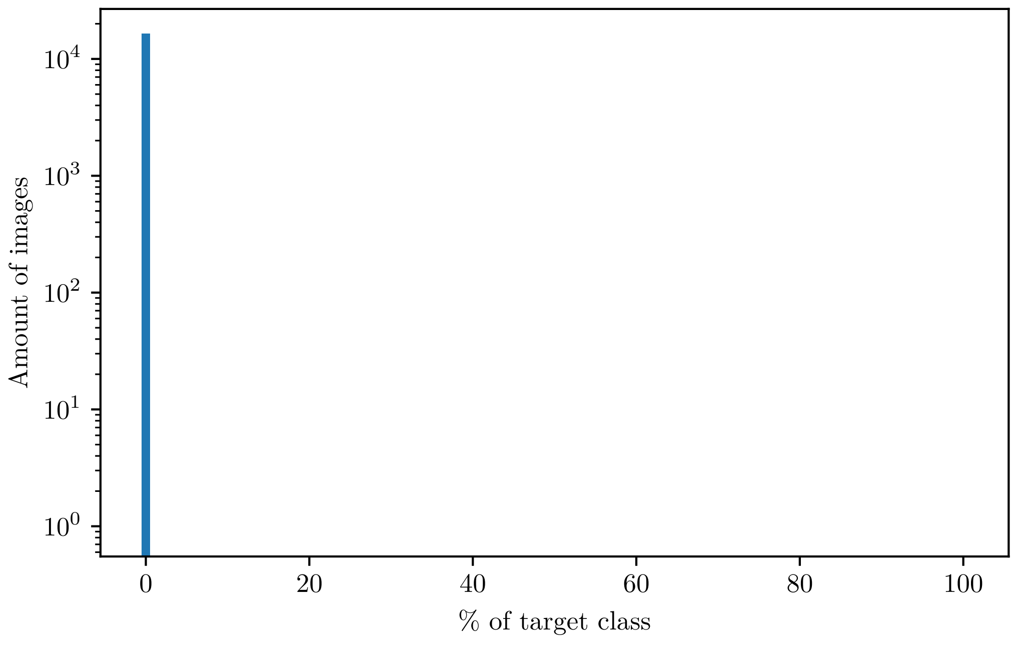


Figure 5.15: Distribution of *Solidago virgaurea* in the dataset (log scale)

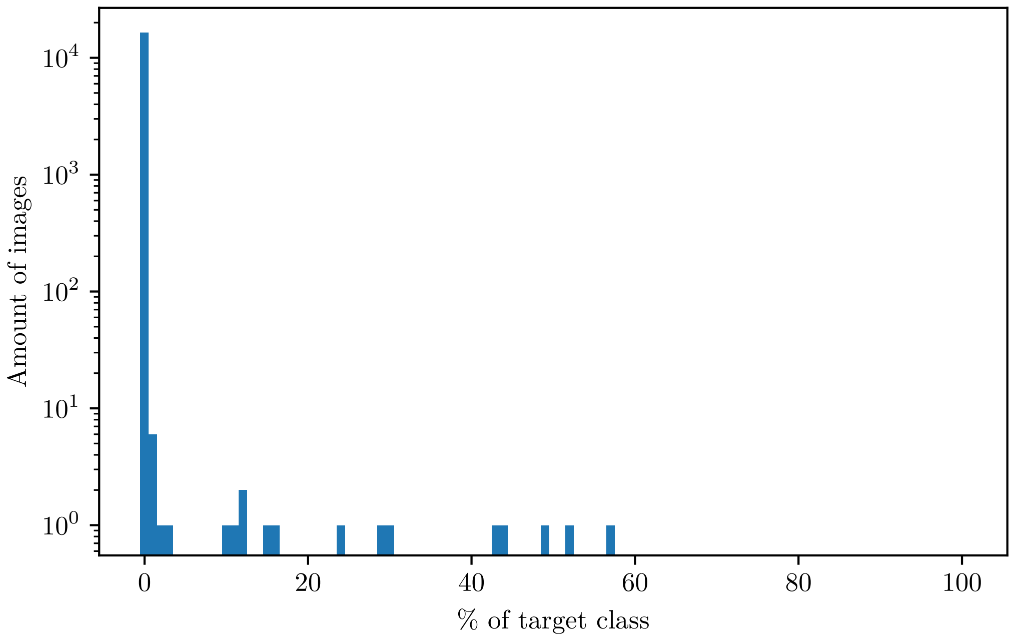


Figure 5.16: Distribution of *Rosa rugosa* in the dataset (log scale)

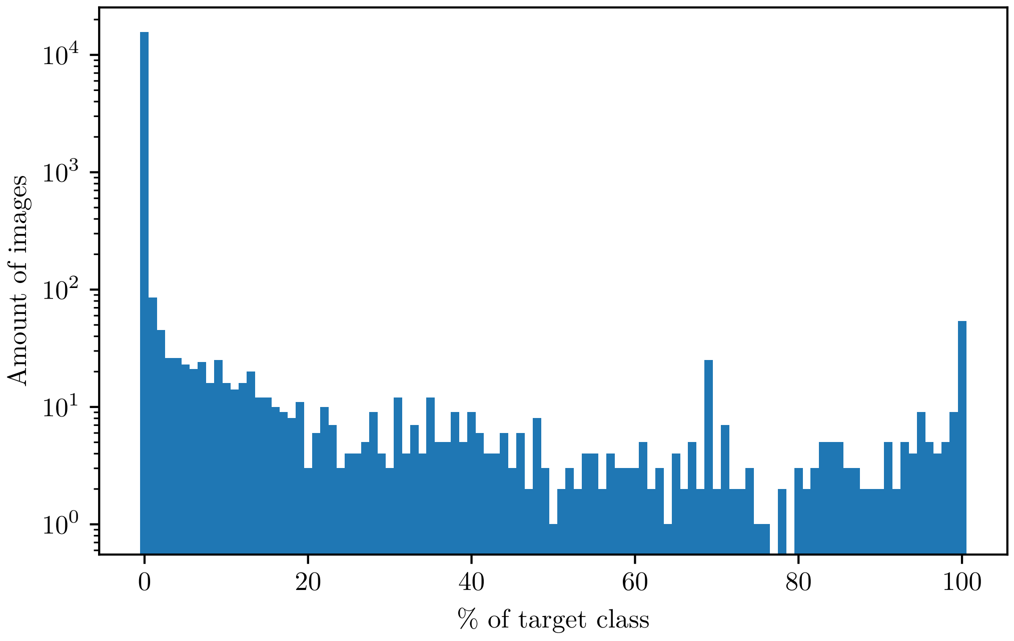


Figure 5.17: Distribution of *Lupinus polyphyllus* in the dataset (log scale)

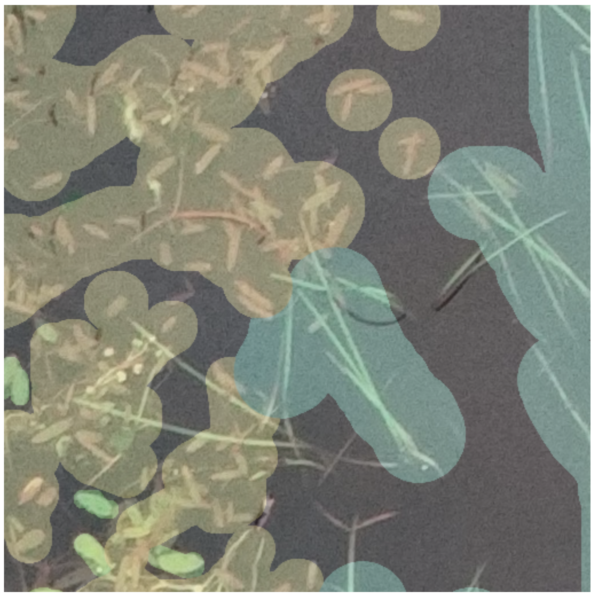


Figure 5.18: Example of a vague annotation. Some of the plants belonging to the *Sparganium gramineum* (cyan) are actually marked as belonging to the *Sagittaria natans* (yellow) class in the bottom-left quadrant, and both classes contain pixels which should be classified as ”water” or ”out of research interest”.

job, but at times human fatigue can clearly be seen in the annotations which have been created with a rather large brush. An example of this can be seen in Figure 5.18. Furthermore, the dataset contains images where class exemplars have not been annotated, as can be seen in Figure 5.19. The two problems mentioned above result in noisy class distribution in training set.

Furthermore, it is sometimes hard to explicitly say where a plant starts, especially when we consider the stem. Taking aerial photos case results in an image where the plant’s stem slowly fades out of the image as it goes deeper and doesn’t reflect enough light for human eye, or indeed the drone’s camera, to see it.

The dataset is also temporally limited, which is a problem because Finland has four seasons. While it wouldn’t make sense to include images from winter (due to freezing of the lakes), the dataset might benefit from having spring - summer - fall variance in it. Some classes, such as *Persicaria amphibia*, also have blooms, which greatly alter the classification task. It is naturally easier to classify them when they are blooming. Despite these shortcomings, the dataset was considered good enough to train some of the models.

5.1.2 Qualitative analysis of the dataset

To get a better picture of the problems mentioned above and depicted in figures 5.18 and 5.19, a qualitative analysis for the dataset was conducted. The analysis consisted

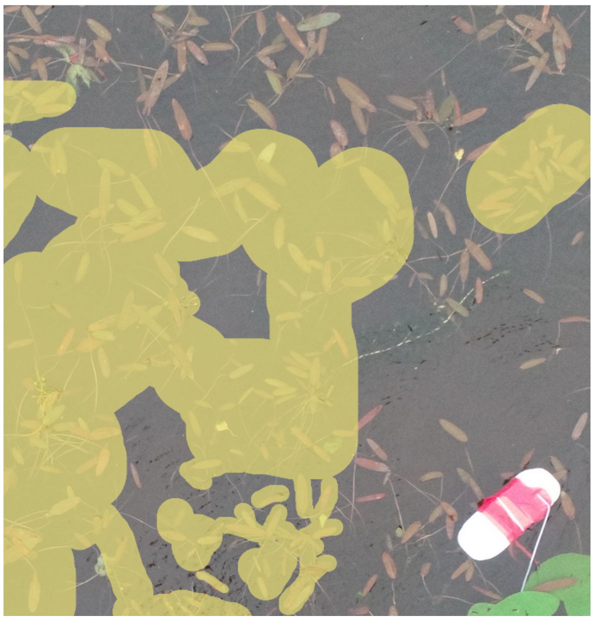


Figure 5.19: Example of an incomplete annotation. As can be seen from the image, some parts of the *Sagittaria natans* (yellow) class have not been annotated at all, while other parts have been annotated with a large brush.

	Amount	Percent
Potamogeton natans	3919	20.16
Phragmites australis	3069	15.79
Sparganium gramineum	2760	14.2
Nuphar lutea	2404	12.37
Sagittaria natans	1019	5.24
Persicaria amphibia	942	4.85
Sparganium emersum	656	3.37
Scirpus	331	1.7
Hydrocharis morsus-ranae	84	0.43
Sagittaria Sagittifolia	8	0.04
Total	19448	100

Table 5.2: Absolute number and proportion of 512x512 tiles with given class in biomass estimation dataset. Note that the values do not sum up to 19448 images or 100%

	Amount	Percent
Impatiens glandulifera	2302	11.84
Heracleum persicum	1555	8.0
Lupinus polyphyllus	820	4.22
Reynoutria japonica	146	0.75
Rosa Rugosa	17	0.09
Solidago canadensis	0	0
Solidago virgaurea	0	0
Total	16401	100

Table 5.3: Absolute number and proportion of 512x512 tiles with given class in alien invasive species dataset. Note that the values do not sum up to 16401 images or 100%

of going through each annotation and deciding if the annotation was done well or if it had some problems from machine learning point of view. Three categories of mistakes were flagged: broad strokes, wrong category and absence of annotation. ”Broad brush” means a situation where the annotation is mostly correct, but due to usage of a large brush to paint the area, the annotation is inaccurate. ”Wrong category” means a situation where member of a class has been painted with the colour of another class, and ”absence of annotation” means a situation where member of a class has not been painted. It should be noted that the above flags are not mutually exclusive: indeed, ”wrong category” is often a result of using too broad brush. If none of these errors were present, the annotation was flagged as good.

It should be noted that correct data annotation is a notoriously difficult problem, and the data in this thesis exacerbates the problem. The main problem is that the dataset contains multiple large pictures with plenitude of small details, which can be frustrating for humans to annotate accurately. The problems outlined below are a result of resource limits (e.g. work time spent on the annotation job), and as noted in Section 5.2, the problems do not seem to greatly affect the end results. However, there were prevalent problems in the dataset, which should be addressed when discussing the results of this thesis. Summaries of these problems are given in tables 5.4 and 5.5. Note that the amounts of problems do not sum up to total number of images. This is because some images had multiple problems, and others had no annotations at all. The latter was not considered a ”good” annotation, but simply skipped over as it would be excluded from the training data and as such would not affect the training process. Thus, for example dataset 3 in table 5.4 has 110 images, but the amount of annotation grades do not reach 110.

Invasive alien species data had three datasets. Their problem cases are presented

Table 5.4: Description of IAS dataset image problems

	Set 1	Set 2	Set 3
Total number of images	30	52	110
Broad brush	18	1	89
Wrong class	1	0	1
Absence of annotation	6	1	2
Good	7	50	9

in table 5.4. Most of the problems revolve around using a broad brush, with small amount of target class members missing annotations and a few cases where a class was incorrectly annotated. Notably, the second dataset has really good annotations. However, the images in this dataset contain only small amounts of target classes. This reflects a larger trend in all of the images: those with small amounts of target classes were annotated exceptionally well, while those with medium to large amounts of target classes were annotated using broader brushstrokes. Furthermore, the only target class to appear in this dataset is *Rosa rugosa*.

In the first and third sets of the alien invasive species dataset the most prevalent problem is using a broad brush. Especially the first dataset is exceptionally hard to annotate even for a human equipped with extreme patience due to the abundance of flora. This makes it difficult to spot the members of the target classes and especially annotate them precisely. Furthermore, the height differentials create lots of shadows which obscure some parts of the images, adding further complications.

On the biomass estimation side of the dataset, problems are more prevalent. The problems are listed in Table 5.5. Again, the problems are concentrated to the broad brush-category with a few problems in other categories. This is probably due to the nature of the dataset: for example, *Sparganium gramineum* and *Sparganium emersum* are long, thin plants which are hard to annotate in an exact manner with the circle brush that was available to the annotators. The broad brush strokes in these annotations result in a class that has high amount of water pixels in them. Similarily, *Sagittaria natans* is a small plant, but tends to form colonies, which makes exact annotation an extremely time-consuming process. Due to *Sagittaria natans* often coexisting among other target classes, the broad brush strokes creates annotations which include large amounts of members of other classes in them. Especially in set 1 (Table 5.5), these broad strokes led to other classes of errors.

In hindsight, biomass estimation annotation would probably have benefited from using a colour range selection tool. Due to the background consistently being of different colour (ie. different colours of water), the annotations could have been created

	Set 1	Set 2	Set 3
Total number of images	104	63	56
Broad brush	112	42	50
Wrong class	26	0	0
Absence of annotation	29	3	0
Good	2	12	0

Table 5.5: Description of BE dataset image problems

by first selecting all green parts of the image, and then labelling those parts. It is also interesting to speculate on the effect of drone flying altitude on annotations: biomass estimation dataset 1 happens to be the one flown at 10 meters and contains more errors than sets 2 and 3, which were flown at 5 meters. Set 1 also contains considerably more errors, even when taking into account the fact that it has approximately twice the number of images in it. The intuitive reason for this would be a frustrated annotator rushing the annotation process due to the sheer amount of work going into the annotation of a single image. A possible experiment to test this hypothesis would be to offer two groups of annotators the same image dataset, but one would be tiled into many smaller images, while one would contain large images, and see which set has more accurate annotations.

While this audition has found some problems in the dataset, it was conducted by a person who is not an expert in biology. Thus, there is a high probability of both false negatives and false positives in ”wrong class” or ”absence of annotation” categories. However, the broad brush-category is easily identified even by people who are not experts in biology. Furthermore, as can be seen in the results, it seems that the annotations were good enough for the classes that had enough data.

5.1.3 Preprosessing

Some preprocessing for the input images is required to begin the training pipeline. To begin the preprocessing, both the input, as well as the ground truth images are tiled into a 512 x 512 pixel tilemap. The size of 512 x 512 has no special meaning; it was chosen arbitrarily. Because every image does not have dimensions which would be a multiple of 512, extra black pixels are added to the right and bottom part of the tiles when necessary. Due to the sparsity of the target classes in the dataset (see figures 5.2 through 5.17), especially when training single-class models, the next step of data preprocessing is to select which tiles are used.

To select the dataset used for single-class-model training, a Python script goes through each ground truth image and checks if any pixel of that image contains that class in question. Based on the script, the dataset is then divided into images that contain the class in questions and those that do not. Different mixes between the two categories were tried, but in the end best training results were achieved when the training set consisted only of images in ”contains the class”-category. Elaboration on this can be found in Section 5.1.1.

5.1.4 Training

Layer	Type	Output Shape	Connected to
$input_1$	InputLayer	(None, 512, 512, 3)	
$conv2d_1$	Conv2D	(None, 512, 512, 64)	$input_1$
$conv2d_2$	Conv2D	(None, 512, 512, 64)	$conv2d_1$
max_pooling2d$_1$	MaxPooling2D	(None, 256, 256, 64)	$conv2d_2$
$conv2d_3$	Conv2D	(None, 256, 256, 128)	max_pooling2d$_1$
$conv2d_4$	Conv2D	(None, 256, 256, 128)	$conv2d_3$
max_pooling2d$_2$	MaxPooling2D	(None, 128, 128, 128)	$conv2d_4$
$conv2d_5$	Conv2D	(None, 128, 128, 256)	max_pooling2d$_2$
$conv2d_6$	Conv2D	(None, 128, 128, 256)	$conv2d_5$
max_pooling2d$_3$	MaxPooling2D	(None, 64, 64, 256)	$conv2d_6$
$conv2d_7$	Conv2D	(None, 64, 64, 512)	max_pooling2d$_3$
$conv2d_8$	Conv2D	(None, 64, 64, 512)	$conv2d_7$
$dropout_1$	Dropout	(None, 64, 64, 512)	$conv2d_8$
max_pooling2d$_4$	MaxPooling2D	(None, 32, 32, 512)	$dropout_1$
$conv2d_9$	Conv2D	(None, 32, 32, 1024)	max_pooling2d$_4$
$conv2d_{10}$	Conv2D	(None, 32, 32, 1024)	$conv2d_9$
$dropout_2$	Dropout	(None, 32, 32, 1024)	$conv2d_{10}$
up_sampling2d$_1$	UpSampling2D	(None, 64, 64, 1024)	$dropout_2$
$conv2d_{11}$	Conv2D	(None, 64, 64, 512)	up_sampling2d$_1$
$concatenate_1$	Concatenate	(None, 64, 64, 1024)	$dropout_1$, $conv2d_{11}$
$conv2d_{12}$	Conv2D	(None, 64, 64, 512)	$concatenate_1$
$conv2d_{13}$	Conv2D	(None, 64, 64, 512)	$conv2d_{12}$
up_sampling2d$_2$	UpSampling2D	(None, 128, 128, 512)	$conv2d_{13}$
$conv2d_{14}$	Conv2D	(None, 128, 128, 256)	up_sampling2d$_2$
$concatenate_2$	Concatenate	(None, 128, 128, 512)	$conv2d_6$, $conv2d_{14}$
$conv2d_{15}$	Conv2D	(None, 128, 128, 256)	$concatenate_2$
$conv2d_{16}$	Conv2D	(None, 128, 128, 256)	$conv2d_{15}$

up_sampling2d$_3$	UpSampling2D	(None, 256, 256, 256)	conv2d$_{16}$
conv2d$_{17}$	Conv2D	(None, 256, 256, 128)	up_sampling2d$_3$
concatenate$_3$	Concatenate	(None, 256, 256, 256)	conv2d$_4$, conv2d$_{17}$
conv2d$_{18}$	Conv2D	(None, 256, 256, 128)	concatenate$_3$
conv2d$_{19}$	Conv2D	(None, 256, 256, 128)	conv2d$_{18}$
up_sampling2d$_4$	UpSampling2D	(None, 512, 512, 128)	conv2d$_{19}$
conv2d$_{20}$	Conv2D	(None, 512, 512, 64)	up_sampling2d$_4$
concatenate$_4$	Concatenate	(None, 512, 512, 128)	conv2d$_2$, conv2d$_{20}$
conv2d$_{21}$	Conv2D	(None, 512, 512, 64)	concatenate$_4$
conv2d$_{22}$	Conv2D	(None, 512, 512, 64)	conv2d$_{21}$
conv2d$_{23}$	Conv2D	(None, 512, 512, 2)	conv2d$_{22}$
conv2d$_{24}$	Conv2D	(None, 512, 512, 1)	conv2d$_{23}$

Table 5.6: Network architecture

The architecture for the experiments was taken from Ronneberger et al. [60] and is described in Table 5.6. Note that the concatenation layers are connected to two other layers - these are the skip connections. ReLU activations were used on every layer except the last, which used a sigmoid activation. In addition, Adam was used as optimizer, loss was calculated with binary crossentropy (Equation 3.7) and keras accuracy metric (Equation 5.1) was used to calculate training accuracy during training, with IOU scores calculated for each model at the end of the training phase. In addition to Figure 4.1, dropout was added to the end of the downsampling part and the beginning of the upsampling part of the U-net.

AWS SageMaker [4] was used for the training process. The training is done with three-fold cross-validation technique, which separates the training data into three parts. The same model is then trained three times (”over three folds”), using two thirds of the data as training set and one third as a evaluation set. The parts which are in the training set and in the evaluation set change for each fold. Cross-validation folds are generated by Scikit-learn’s [56] K-Fold class. The model itself is compiled with Keras [13] using a TensorFlow [1] backend. In the training phase, input images are shown to the model in batches. An image generator is used to generate the batches for training. The images as well as ground truth images are read from disk by Pillow [24] and then translated into Numpy [50] arrays.

The datasets are augmented during batch generation. The augmentations are applied randomly to 75% of the training images, and they are computed online (as opposed to precomputing the augmentations). Furthermore, the parameters of the augmentations are computed randomly for each batch. This leads to the situation where the same input image could have multiple different augmentations applied to it in different epochs, or it could be presented without augmentations in one epoch, yet appear augmented in another. The augmentations used for input images in are rotation, shearing (shifting one corner of the image while the rest remain stationary), flipping, changing the hue and adding noise to the images. Ground truth images use only distortive augmentations (i.https://hangouts.google.com/call/FiH7Xymzcr4KYIPvPBoFAAEIe. rotation, shearing and flipping). It should be noted that the same distortions are used for a (input, ground truth)-image pair. Imgaug [2] is used to generate the augmentations, with different spatial distortions applied both to the actual images and ground truth images, but colour-affecting distortions only to the actual images.

When training single-class models, the ground truth images need to be cleaned to contain only a boolean map of (”belongs to class”, ”does not belong to class”). This is done separately for each ground truth image to enable usage of distortion augmentations mentioned above. Technically it would be possible to calculate the maps before training, which would save some time during training, but it drastically limits the augmentation possibilities of the dataset and requires more hard drive space. Because the datasets were already small, and hard drive space on cloud computing systems are expensive, doing single-class mapping from general ground truth images to specific class ground truth images was preferred.

Finally, the input images are normalized. Because Pillow reads the images as 8-bit RGB triplets, each pixel in the original input image are described as a triplet (0-255, 0-255, 0-255). Each of the values is divided by 255 to normalize the values to a range [0,1]. The single-class ground truth image described above already includes only values between [0, 1], so further normalization was not necessary.

During the training, the default Keras accuracy function [69] (equation 5.1) was used to measure the accuracy of the model.

$$accuracy(y_{true}, y_{pred}) = mean(equal(y_{true}, round(y_{pred}))), \tag{5.1}$$

where y_{true} is a vector of ground truth values, y_{pred} is the vector of network guesses, *mean* is a standard averaging function, $round(x)$ is a standard elementwise rounding function and *equal* is an elementwise equals function:

$$equal(x, y) = \begin{cases} 1 \text{ if } x = y, \\ 0 \text{ if } x \neq y \end{cases} \tag{5.2}$$

To calculate the loss, binary crossentropy (equation 3.7) was used.

Training for biomass estimation task was done over 25 epochs, each of which goes through the whole dataset once. For alien invasive species task 30 epochs were used because during biomass estimation task it became apparent that some models could have benefitted from longer training times. However, a TensorFlow EarlyStopping module was used. If the test set loss metric plateaus for three consecutive epochs (five for alien invasive species task), the training is considered to have reached a minimum and is halted for said fold. Furthermore, the TensorFlow ModelCheckpoint module, which is used for saving the model, saves the model at the end of an epoch, but only if it has scored better on the test set loss metric than the previous saved model.

After each fold, IOU scores (see 4.1) are calculated for the model. This is achieved by running the model against every image in the test set, calculating the IOU for the prediction, and saving all of the IOU:s in a list. However, the model produces values between (0, 1), inclusive, and IOU requires us to supply a true/false map of the image. Thus, a threshold is required. If a pixel's value is above this threshold, we consider this pixel to belong to the class. Otherwise, we consider it to not belong to the class. While relatively simple, this method makes the threshold a parameter in how successful our model is considered to be. Essentially, the same model can get different IOU scores depending on which threshold was chosen.

5.2 Results

This section presents the results of training a U-net model described in the beginning of chapter 5 on the datasets described in Section 5.1.1. IOUs as well as training metrics are provided and discussed. Due to the skewness of datasets each trained model will be discussed separately. It should be noted that models for *Solidago canadensis* and *Solidago virgaurea* were not trained because the dataset did not contain any examples for these.

Due to the small amount of data, it was decided that 3-fold cross validation IOUs were a suitable metric for measuring model success. This doesn't yet provide a statistically sound proof of model's accuracy, but enables making good, informed estimates.

Furthermore, some example predictions generated by the models are shown in figures 5.38a through 5.40b. The figures depict a full-size image taken from *Nuphar lutea* test set and run against the *Nuphar lutea* model. The predictions have been post-processed in Affinity Photo [47] software to compensate for the distortion produced by scaling down a full-sized prediction (around 5472x3648 pixels) image into 480x349 pixel image. The correction was done by applying a level adjustment layer, increasing the black level to 60% for each image. This results in dim colours becoming more vibrant

	Avg IOU
Potamogeton natans	0.856
Phragmites australis	0.814
Sparganium gramineum	0.742
Nuphar lutea	0.625
Sagittaria natans	0.445
Scripus	0.444
Persicaria amphibia	0.353
Sparganium emersum	0.113
Sagittaria sagittifolia	0.039
Hydrocharis morsus-ranae	0.018

Table 5.7: Average of IOU scores for biomass estimation models. 1 is the best possible score, 0 worst.

while not overexposing light colours. Notably all of the predictions display seams around the 512x512 pixel tiles. This is a known feature of U-net architecture, and can be corrected by using overlapping tiling strategy as presented by Ronneberger et al [60]. This process is depicted in Figure 5.37. Essentially, for each prediction, a padding in each direction is used. The window is also moved $2 * \text{padding size}$ less between predictions, which results in the areas inside the paddings being used in prediction twice. For example, in 5.37, the light red square corresponds to the a tile given to the model for prediction, and bright red to the area that is used of the prediction. Similarly, the light blue corresponds to the next tile given to the model for prediction, and the dark blue to the area that ends up being used from that prediction. Using such procedure for a whole input image should cut the number of seam lines in prediction images. However, it was not implemented in this thesis due to it not being a finalised product.

The example prediction in Figure 5.38a depicts a situation where the input image contains a cluster of target class (*Nuphar lutea*), a cluster containing both *Nuphar lutea* and other flora as well as a small cluster of other flora. As can be seen from 5.38b, the model has fairly good results on this image. Some out-of-class flora have triggered a small response (probably due to them being also green), and some members of *Nuphar lutea* - especially those whose colouring is yellowish - trigger weaker response than a truly robust model would, but in general the model behaves well. Similar success can be seen in figures 5.39a and 5.39b, where input contains only *Nuphar lutea*. Figure 5.40a contains an input without any *Nuphar lutea*, but some non-target-class. As can be seen in Figure 5.40b, the right-hand-side non-target-class flora has triggered a small response, but one not on par with actual *Nuphar lutea*.

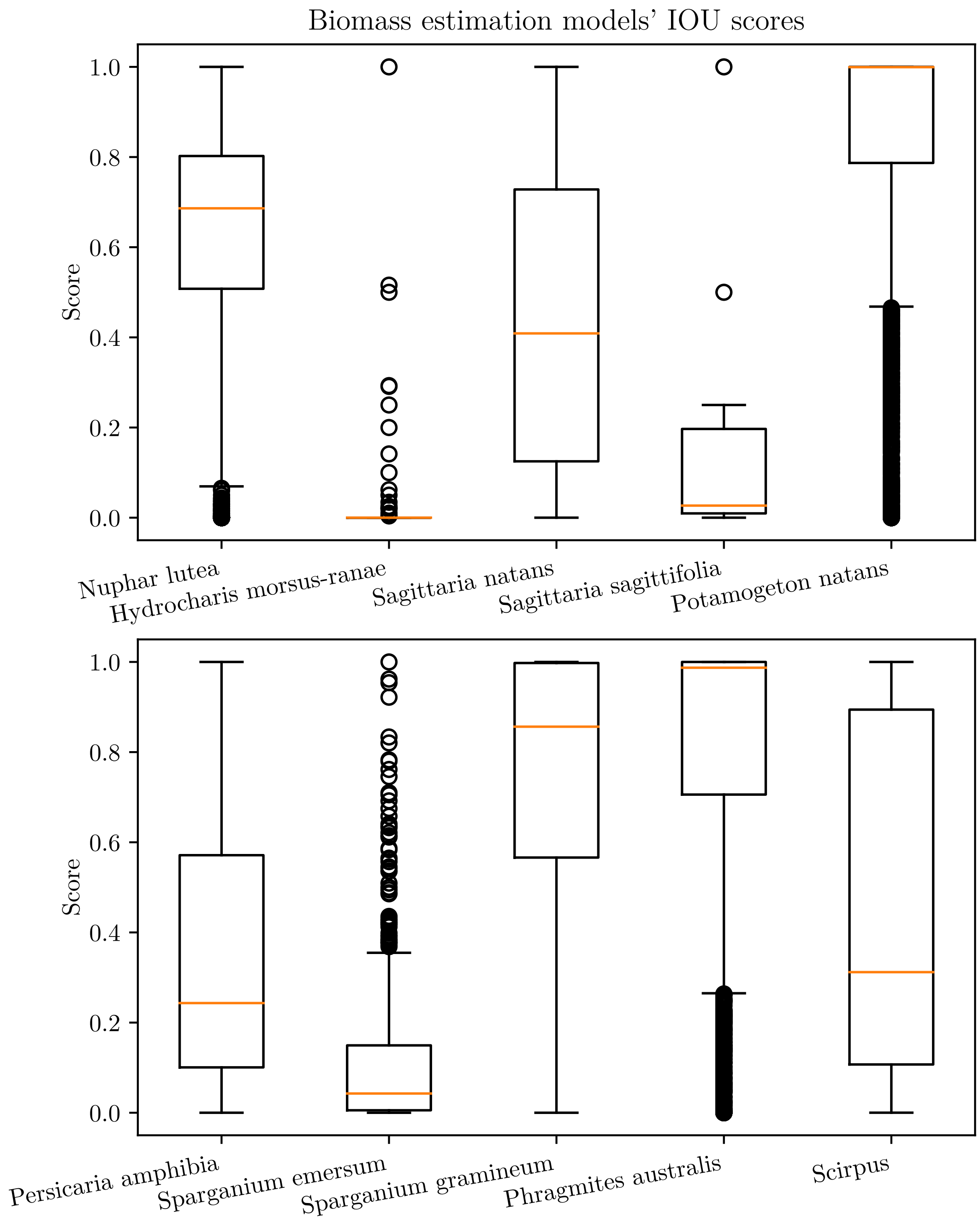


Figure 5.20: IOU scores for biomass estimation task.

5.2.1 Biomass estimation

The average IOUs for biomass estimation task are listed in table 5.7. As can be seen, the results vary quite significantly from good (*Potamogeton natan*'s 0.856) to poor

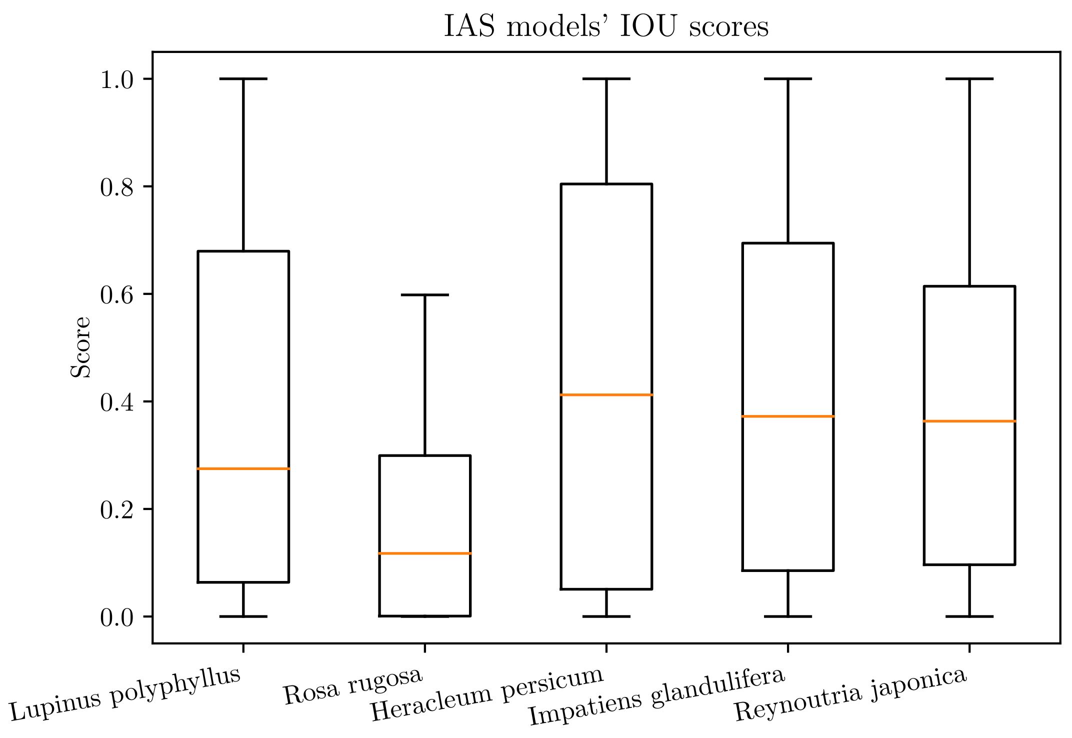


Figure 5.21: IOU scores for invasive alien species task.

(*Hydrocharis morsus-ranae*'s 0.018). Boxplots for IOUs with 95% confidence intervals are reported in Figure 5.20. These reflect the results in table 5.7. However, it is clear from the figure that each model has images it thrives on as well as images it has very clear problems with. It is probable that the good results the poor models get in some images result from input image overlapping. In this context, overlapping means that there are some images in the dataset which have overlapping parts. If one of such images ends up in the training set and the other in evaluation set, it is possible for the model to have seen a part - albeit from a slightly different angle - of the evaluation image during training, which would explain some of the high scores scored by poor

	Avg IOU
Heracleum persicum	0.447
Reynoutria japonica	0.409
Impatiens glandulifera	0.407
Lupinus polyphyllus	0.374
Rosa rugosa	0.245

Table 5.8: Average of IOU scores for invasive alien species models. 1 is the best possible score, 0 worst.

models.

Training metrics for biomass estimation models are given in Figure 5.22 through 5.31. As explained in Section 5.1.4, each model in biomass estimation task was trained with three folds for 25 epochs with an early stop if the evaluation loss metric would not improve during the last three epochs. As with the IOUs, the results vary. *Nuphar lutea* and *Potamogeton natans* exhibit somewhat well-behaved training patterns in each fold: they reach small losses and high accuracy with a logarithmic curve. However, the models achieve good results suspiciously quickly: even the first epoch has relatively high accuracy (> 0.6, >0.8 for *Nuphar lutea*). This might be explained by the problem being relatively easy to begin with, as in biomass estimation task a good first guess is to guess everything with correct colour to belong to these classes, and the shapes of different classes being fairly distinct. However, this hypothesis was not tested.

On the other hand, *Sagittaria natans*, *Scripus*, *Sparganium emersum*, *Sagittaria sagittifolia* and *Hydrocharis morsus-ranae* models have clear difficulties in converging to a solution, probably due to small sample size. *Phragmites australis* has two folds where the model has made almost no progress for a long time before quickly finding a sweet spot and one fold where the spot was found quickly enough to make one suspect some sort of problem in the training phase. *Phragmites australis* is also an example of training which could have benefitted from more training: the two folds where training continued for 25 epochs have clearly not yet converged. *Sparganium gramineum* has a similar training pattern where two folds quickly get to high accuracy and small loss on both training and validation sets, while the third struggles to learn the necessary features.

5.2.2 Invasive alien species

In general, U-net didn't fare as well in alien invasive species task as in biomass estimation. The IOUs for AIS task can be found in Figure 5.21 and Table 5.8. The results are much more uniform compared to biomass estimation task, with the best average IOU scored by *Heracleum persicum* model (0.447) and the worst by *Rosa rugosa* (0.245). However, the *Heracleum persicum* result is comparable to *Sagittaria natans* and *Potamogeton natans* on biomass estimation task - neither of which behave very well. Boxplots for IOUs with 95% confidence interval can be found in Figure 5.21. As with biomass estimation task, some good results with generally poorly behaving models could be explained by overlapping input images.

Training metrics for alien invasive species task are given in Figures 5.32 through 5.36. Unlike in biomass estimation task, alien invasive species models were given 30 epochs with early stopping after no improvement in evaluation loss for five consecutive epochs. Out of these training sessions, *Rosa rugosa* had no success whatsoever, probably due to small training size (n = 17). *Lupinus polyphyllus* has one somewhat promising fold (number two), but two which failed to learn significant features - again, probably due to small sample size (n=820). However, this model looks like it is on a verge of becoming good, and could've benefitted from better annotations, longer training time or more data. Same holds for *Heracleum persicum* and *Impatiens glandulifera.* For *Reynoutria japonica* the training loss and evaluation loss graphs look good if somewhat slow to learn, but the accuracy graphs behave erratically. The conclusion is that a small training set size (n=146) prevents the model from learning proper features.

In conclusion, the models behave well on select biomass estimation species, mainly *Potamogeton natans*, *Phragmites australis* and *Sparganium gramineum.* None of the alien invasive species models yield useful predictions. These results are probably a combination of a few variables. Firstly, the base problem in biomass estimation is easier. In this task, the probability for given pixel to be a plant versus water is highly conditional on its colour, which provides a good guess in the first epochs. Easy base features would also explain the high starting accuracies. However, in alien invasive species task the class colour does not necessarily differ from out-of-class colour; the grass next to *Reynoutria japonica* is still green. Second, the dataset contains more examples of biomass estimation task images versus alien invasive species task images. There are about as many examples of *Impatiens glandulifera* in the alien invasive species dataset (2302 images, largest dataset in IAS task) as there are *Nuphar lutea* in the biomass estimation dataset (2404 images, fourth-largest dataset in biomass estimation task).

In this chapter, details on using the U-net architecture on finding segmentations for drone images were given. The chapter begun with a description of the general parameters of the test setup. It then presented the dataset used, as well as discussed the problems in the dataset, such as missing or incorrect annotations. Potential ideas on how to achieve better annotations in the future were presented. Then, a description on how the models were trained and how well the training process went was given. Overall, it looks like training went well for a few models, but most would require more data to be useful.

Next, conclusions to this thesis are given.

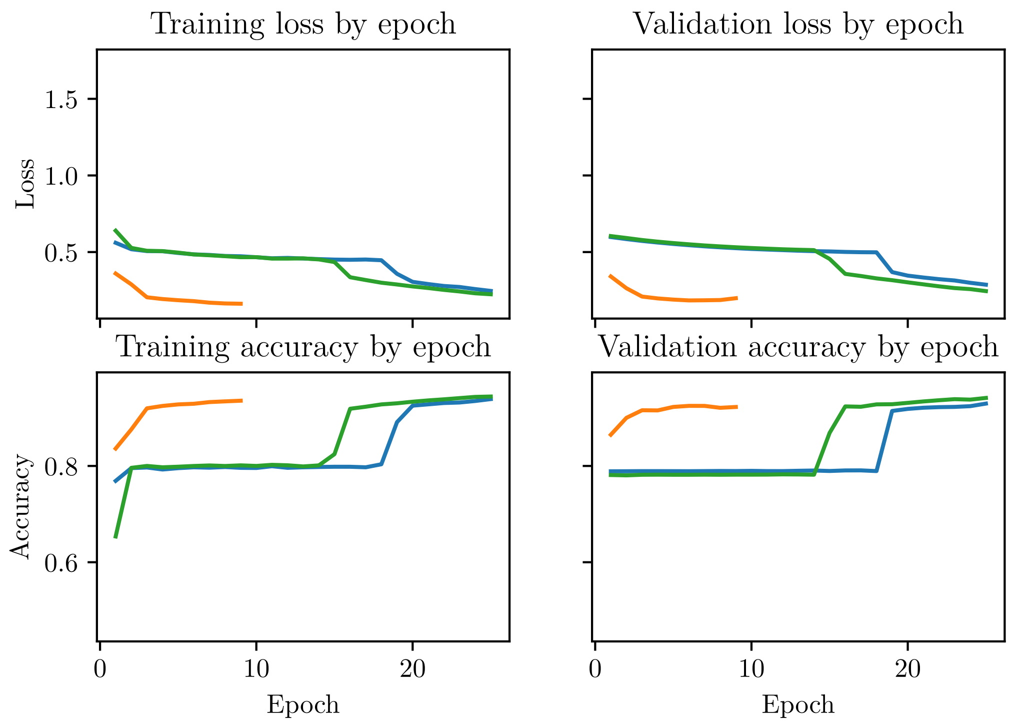


Figure 5.22: Training metrics for *Phragmites australis* (training set size = 3069)

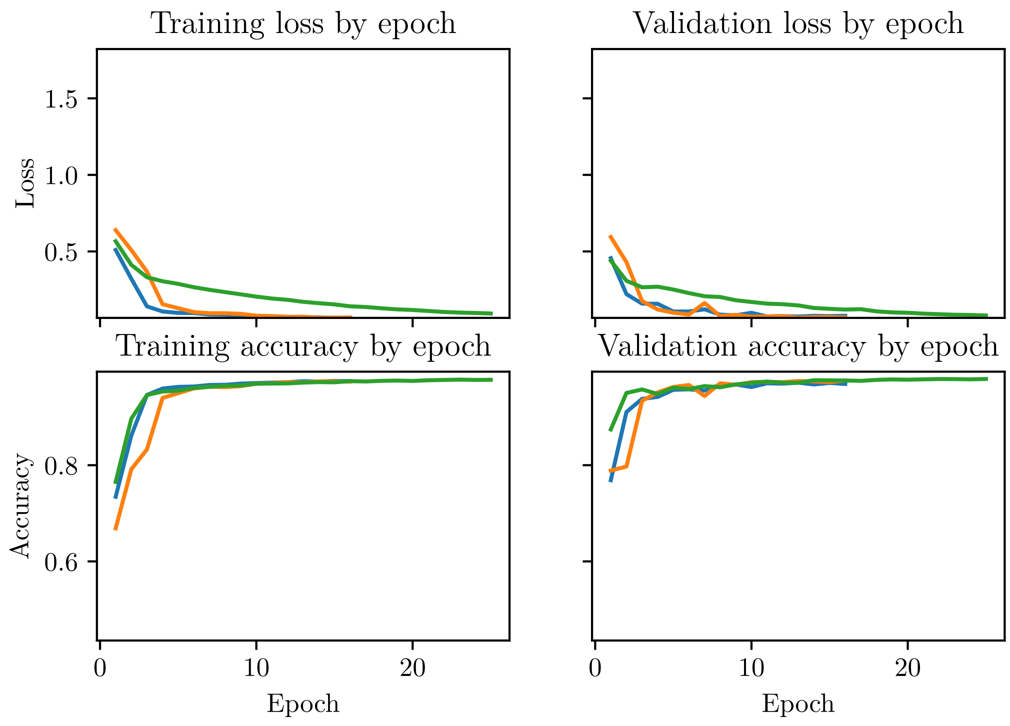


Figure 5.23: Training metrics for *Potamogeton natans* (training set size = 3919)

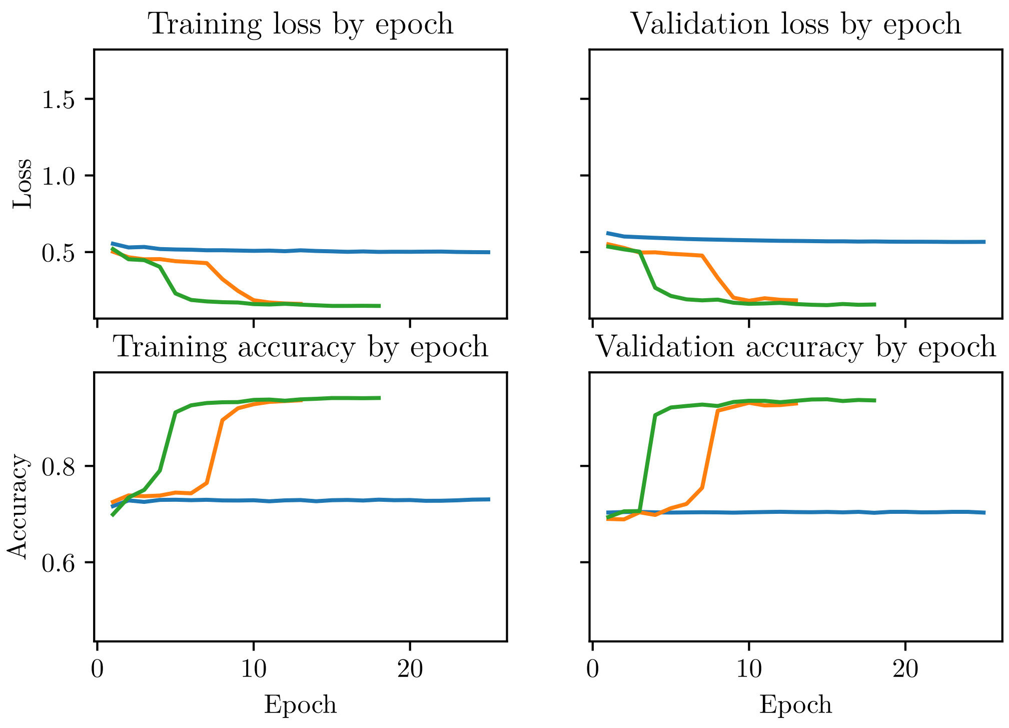


Figure 5.24: Training metrics for *Sparganium gramineum* (training set size = 2760)

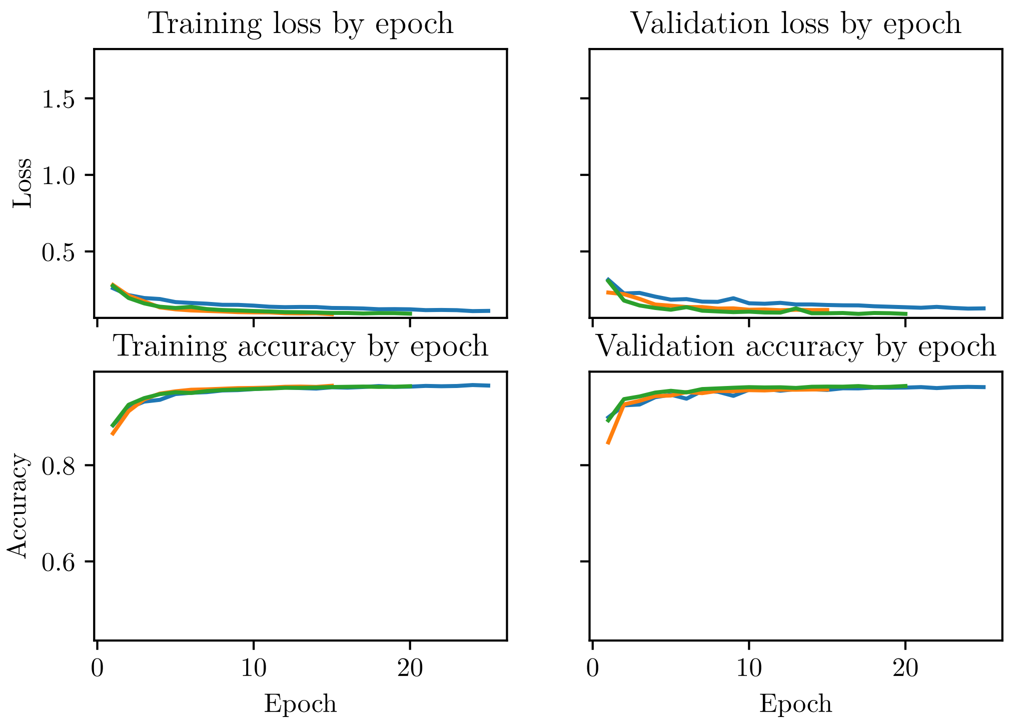


Figure 5.25: Training metrics for *Nuphar lutea* (training set size = 2404)

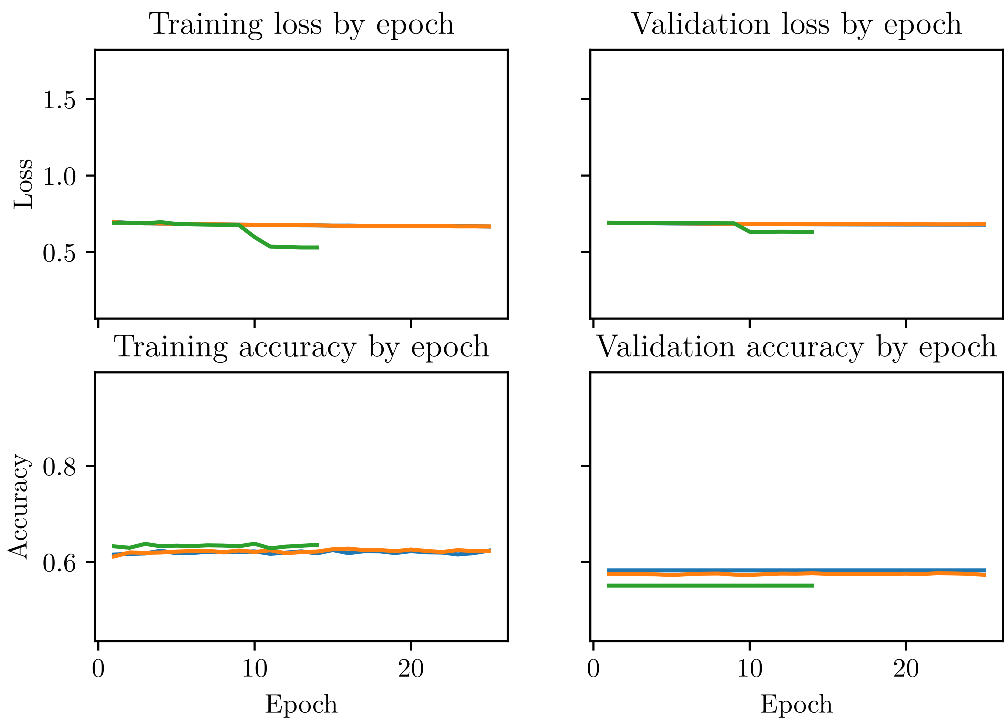


Figure 5.26: Training metrics for *Sagittaria natans* (training set size = 1019)

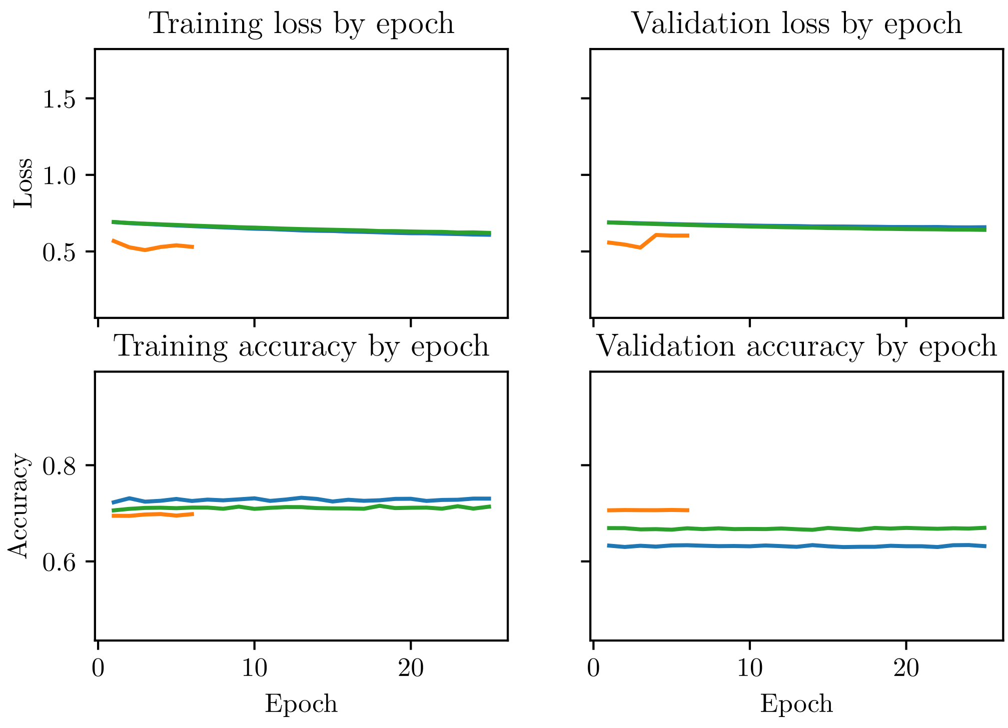


Figure 5.27: Training metrics for *Persicaria amphibia* (training set size = 942)

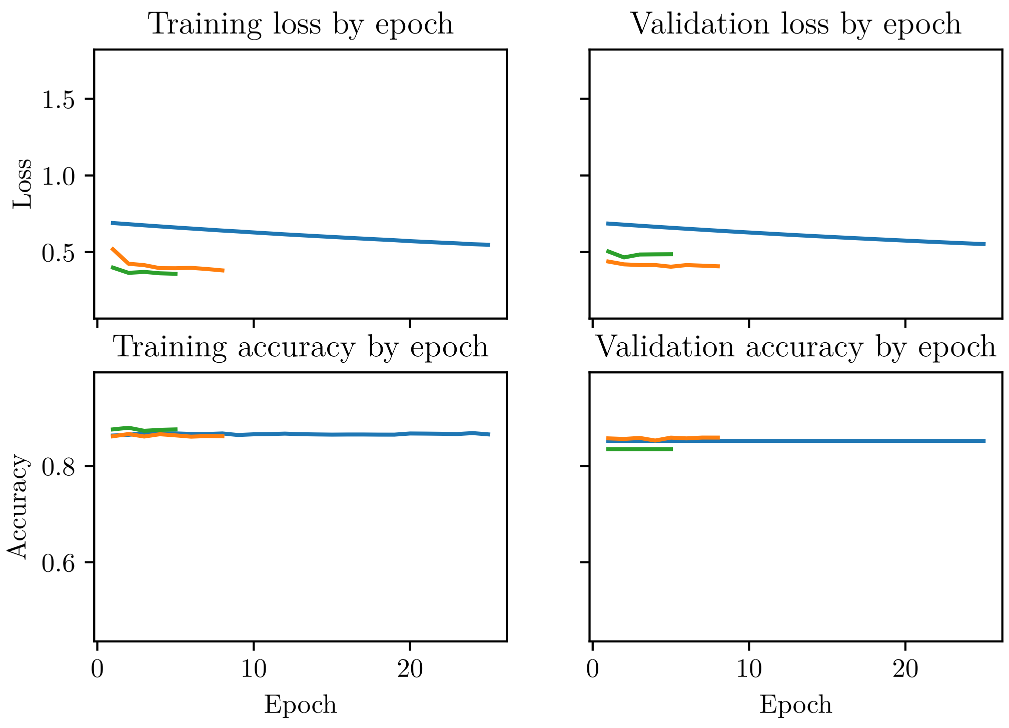


Figure 5.28: Training metrics for *Sparganium emersum* (training set size = 656)

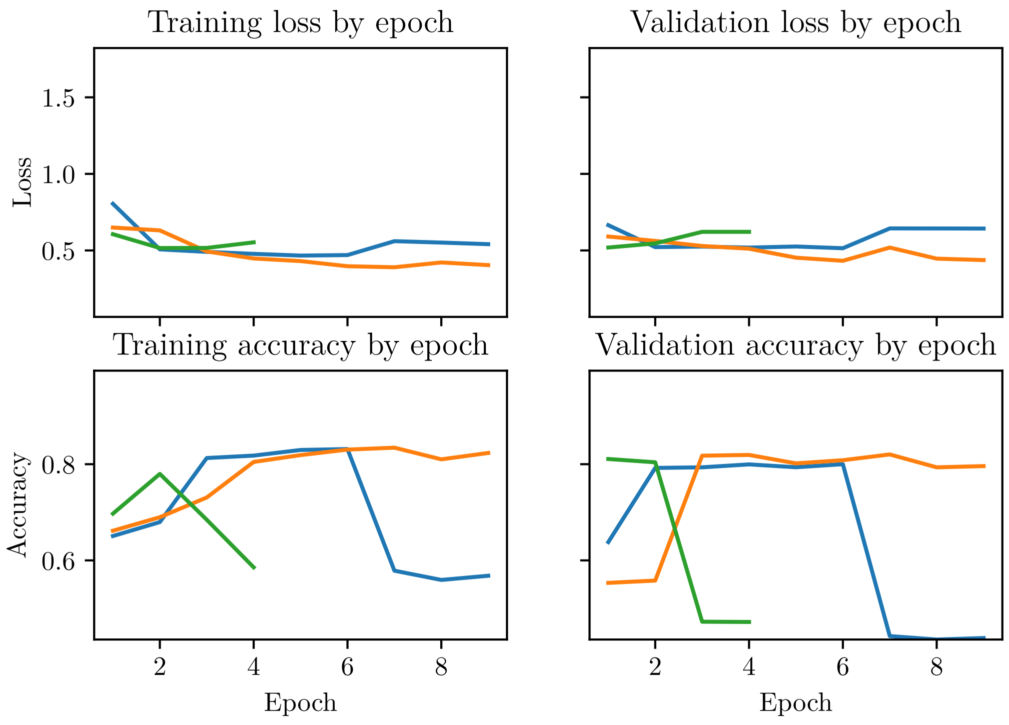


Figure 5.29: Training metrics for *Potamogeton natans* (training set size = 331)

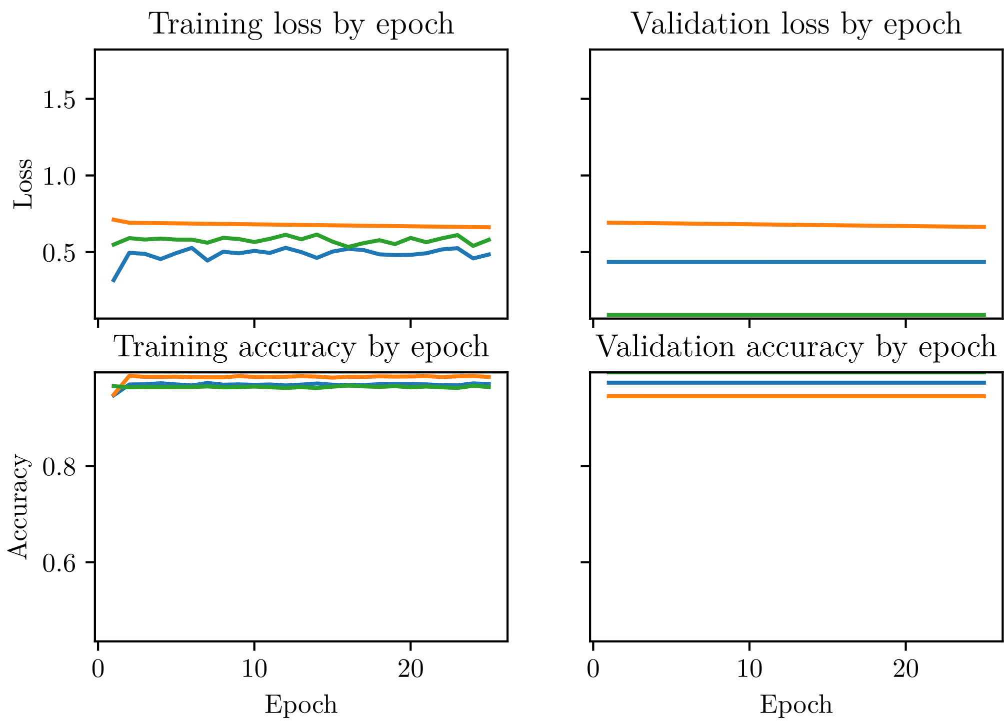


Figure 5.30: Training metrics for *Hydrocharis morsus-ranae* (training set size = 84)

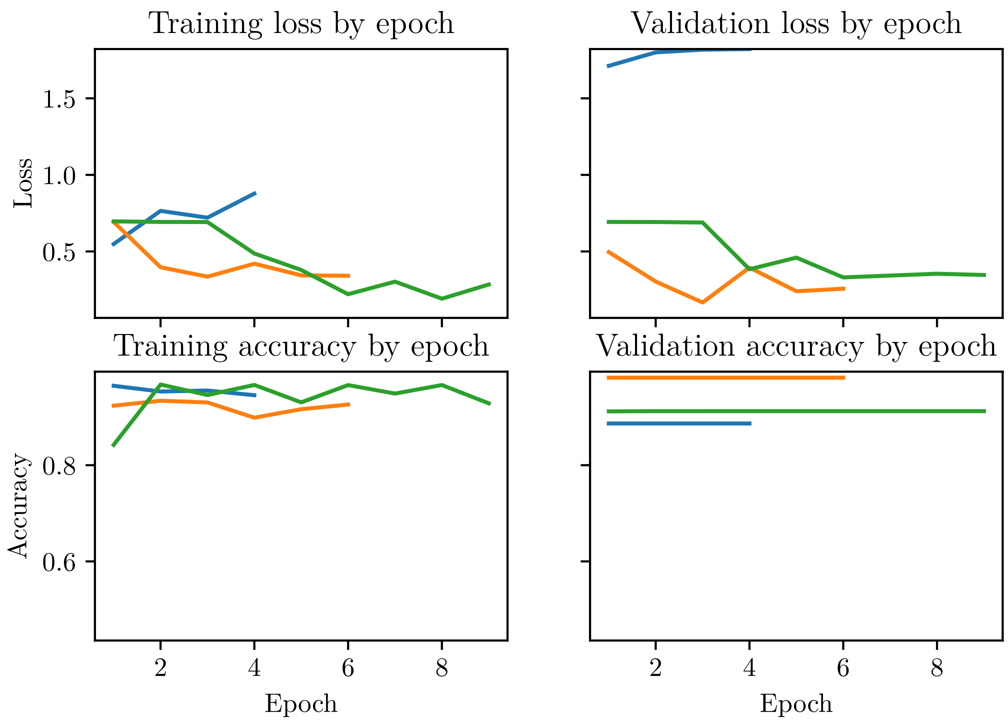


Figure 5.31: Training metrics for *Sagittaria Sagittifolia* (training set size = 8)

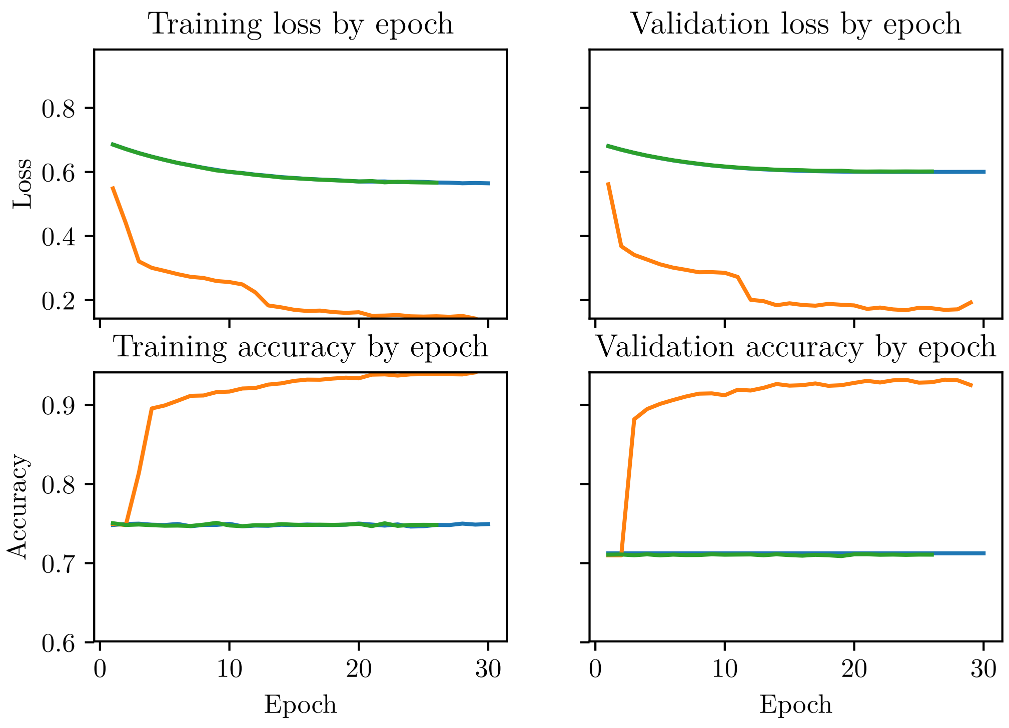


Figure 5.32: Training metrics for *Impatiens glandulifera* (training set size = 2302)

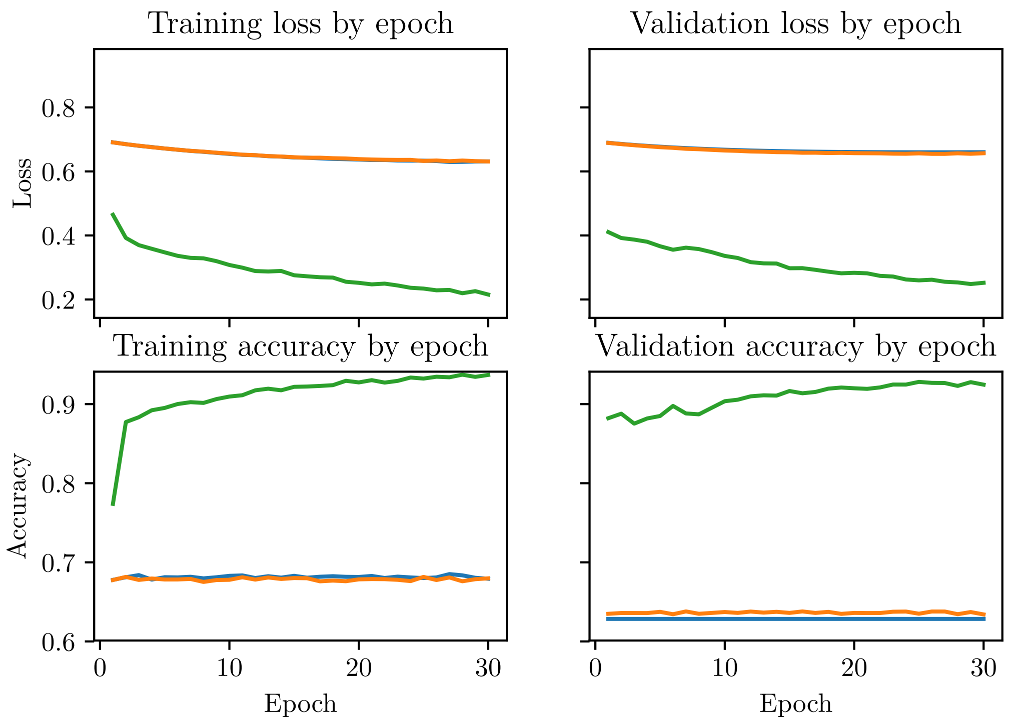


Figure 5.33: Training metrics for *Heracleum persicum* (training set size = 1555)

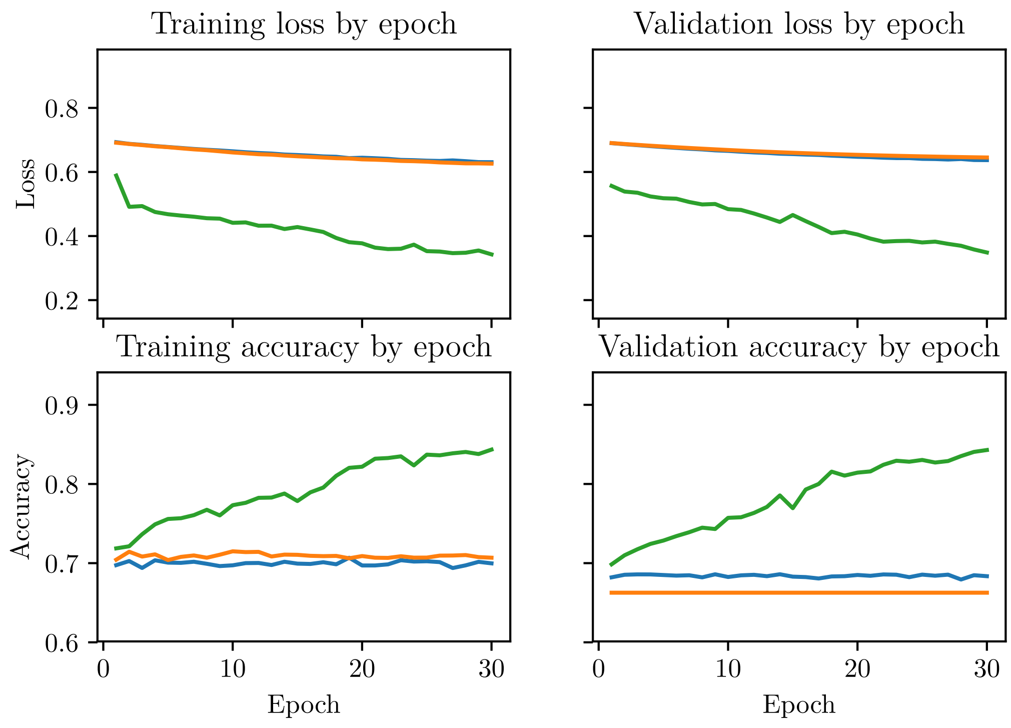


Figure 5.34: Training metrics for *Lupinus polyphyllus* (training set size = 820)

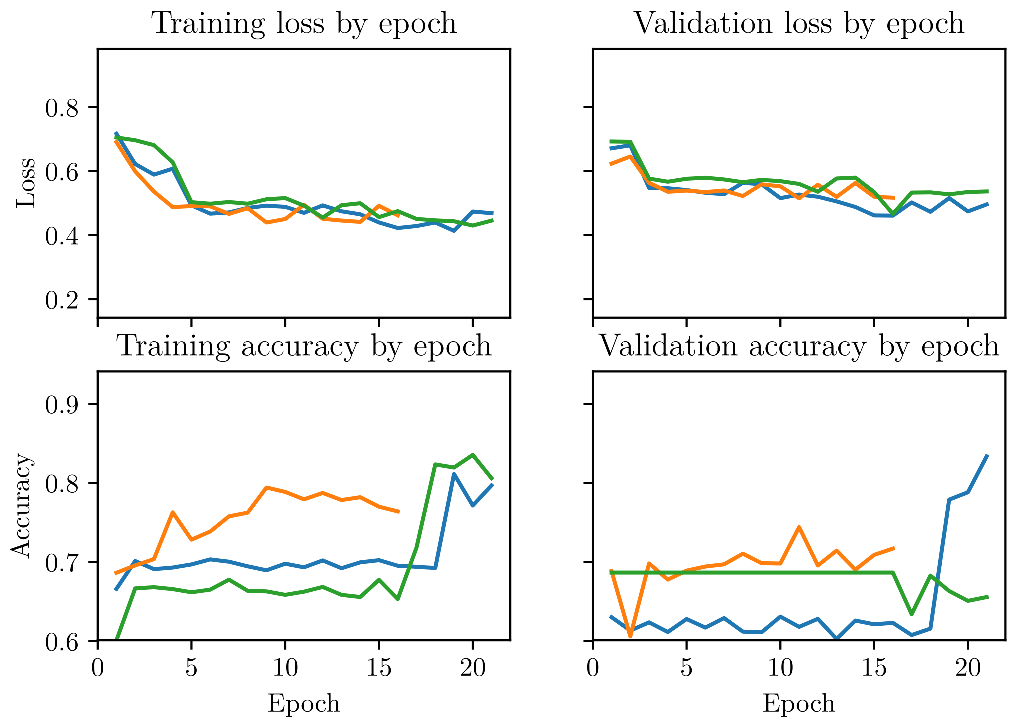


Figure 5.35: Training metrics for Reynoutica japanica (training set size = 146)

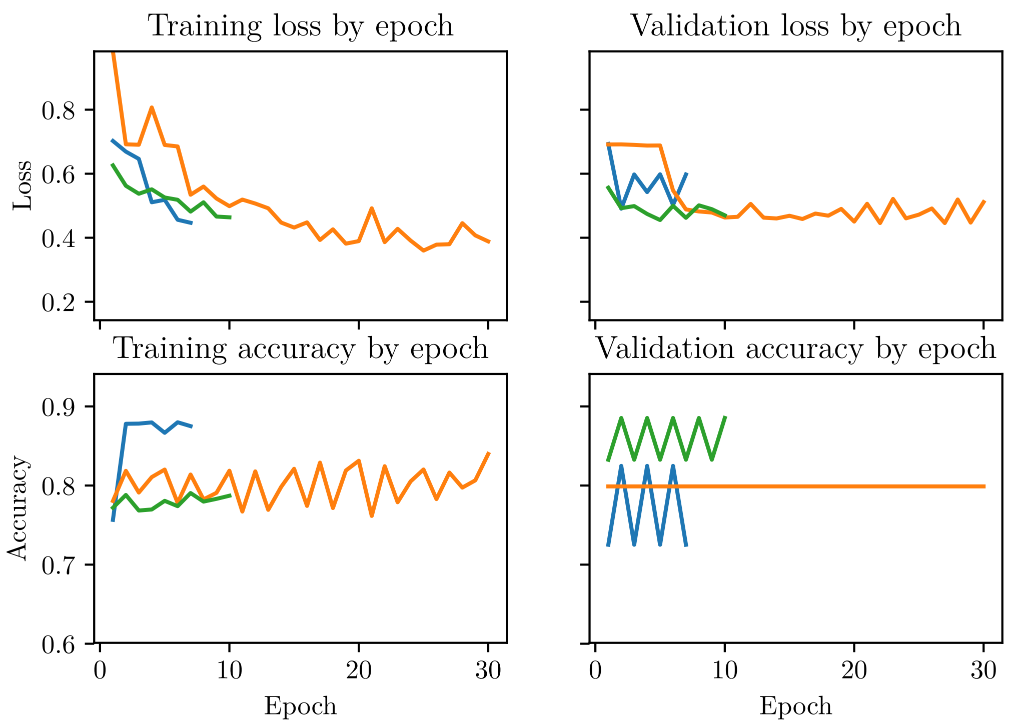


Figure 5.36: Training metrics for *Rosa rugosa* (training set size = 17)

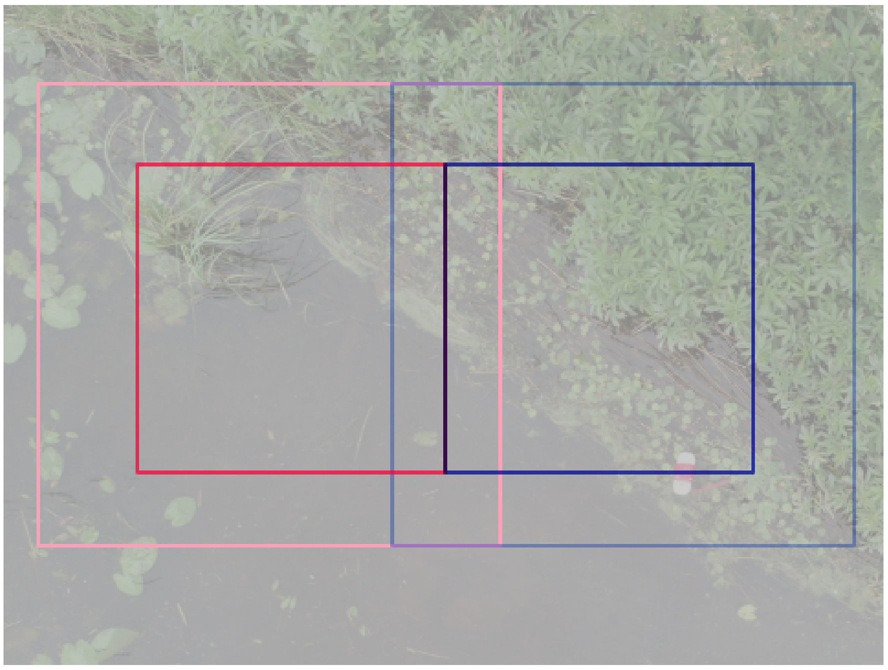
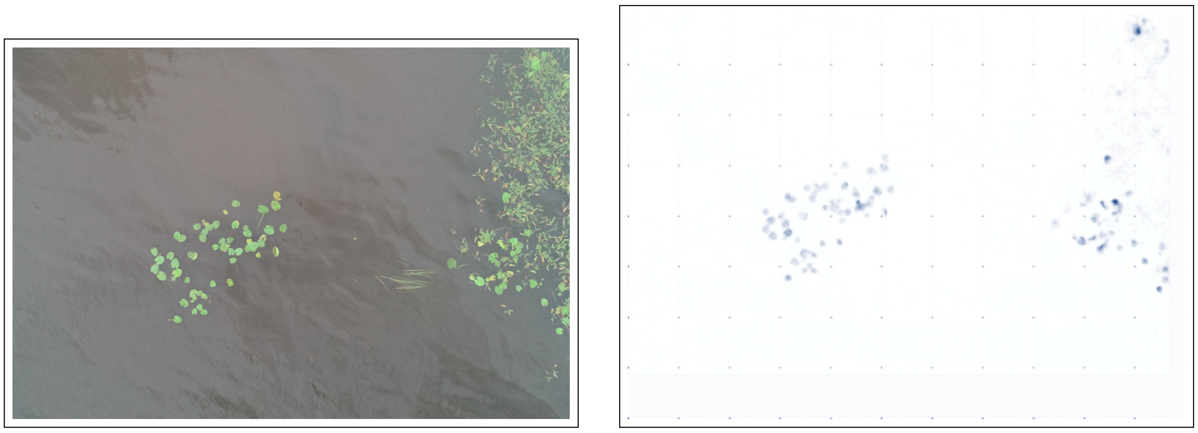
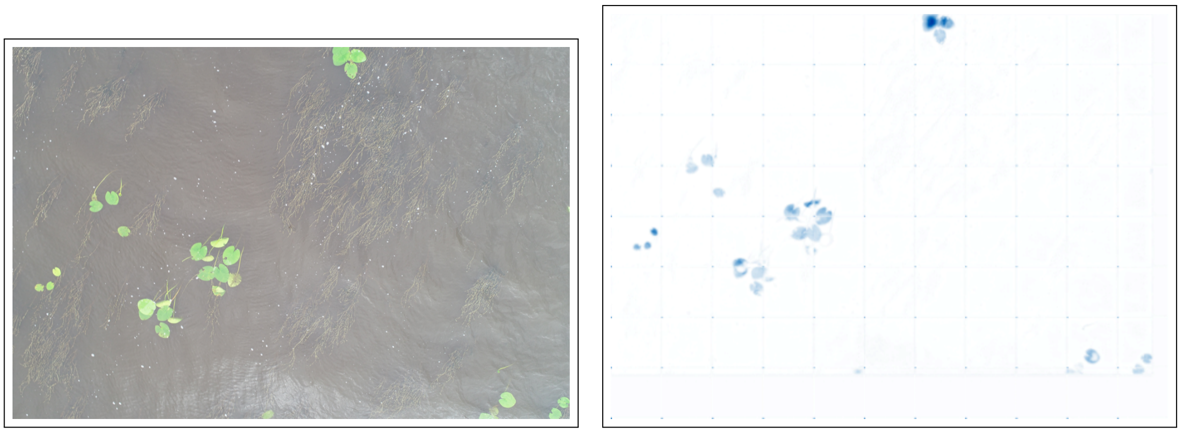


Figure 5.37: Overlapping tiles strategy for seamless predictions



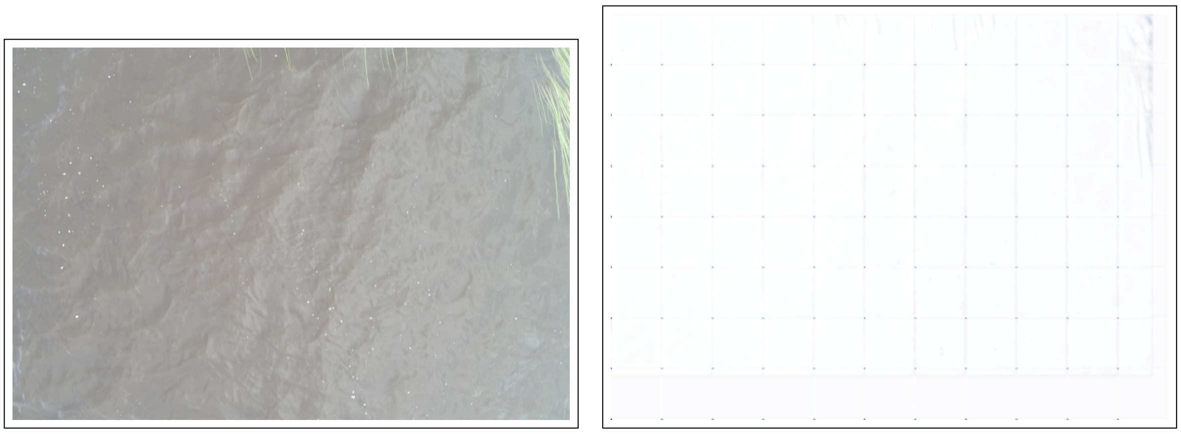
(a) Original image **(b)** Prediction generated by U-net

Figure 5.38: Example image 1 has a cluster of water lilies left from the center and a cluster of water lilies surrounded by other flora on the right side of the image



(a) Original image **(b)** Prediction generated by U-net

Figure 5.39: Example image 2 has a few clusters of water lilies as well as some *Sparganium gramineum*.



(a) Original image **(b)** Prediction generated by U-net

Figure 5.40: Example image 3 has no water lilies, but does include a small cluster of *Sparganium gramineum* in the top-right corner.

6. Conclusions

This thesis has examined U-net as a part of a method to solve the problem of scaling surveillance of wildlife. This problem relates to two goals that SYKE has: to detect invasive alien species, and to estimate lake biomass. If machine vision could be harnessed, a drone could be flown over required areas and results be achieved with considerably fewer human resources. The resulting images could then be fed to a semantic segmentation model in order to know how much biomass or alien invasive species there are in the image. This thesis presented these objectives in the introduction, which was followed by a historical perspective on image segmentation.

Neural networks, and more specifically U-net were chosen as a technique due to promising results they have yielded in other areas of computer vision. To give the reader a better insight into the selected method, the anatomy of a dense feedforward network and a convolutional network were given in chapter 3. Convolutional neural networks, inspired by the mammalian visual cortex, have enjoyed tremendous success in image classification and segmentation tasks. Out of these networks, the U-net architecture [60] was selected as a prime candidate for accomplishing the task at hand. After presenting the related work, this thesis moved from theory into practice by presenting the training environment. A U-net model was trained for each class of interest.

6.1 On the relationship between model success and dataset size

It is a well-known adage in machine learning that more data is better. This thesis is no exception; as can be seen from 6.1, the relation between the IOU score (IAS models in yellow, biomass estimation in purple) and the dataset size is almost linear. The linear dependency is quite visible just by looking at the image, but to be more robust, a Tikhonov regularization with two degrees of freedom was applied, resulting in the green parabolic line in the figure. The coefficients for the estimator's 0th, 1st and 2nd degree members are 0, $2.33111815 * 10^{-4}$ and $-1.35940254 * 10^{-8}$ respectively, which further support the almost linear dependency interpretation. The parabolic nature of

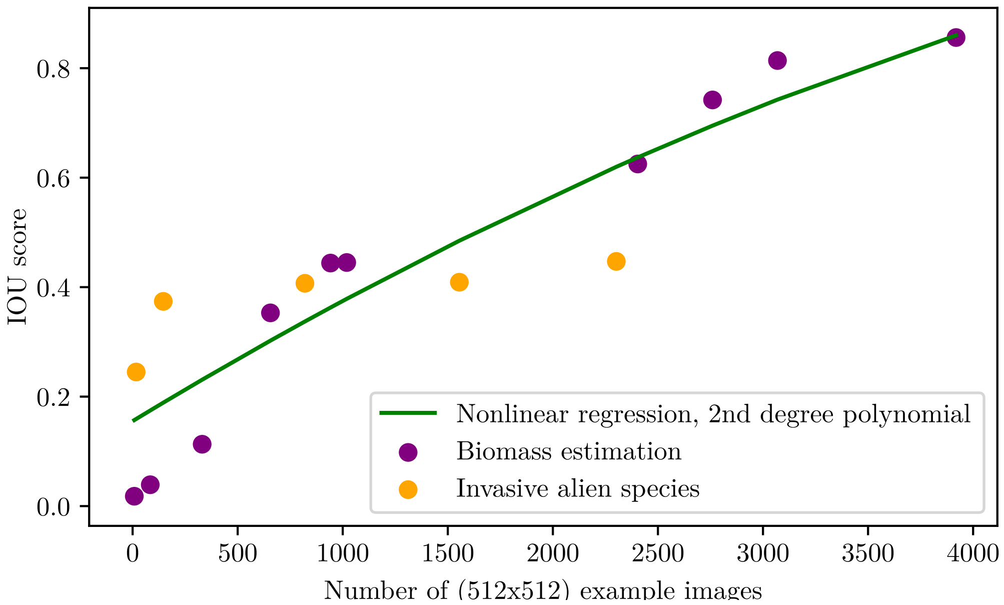


Figure 6.1: Model IOU scores by sample size

the curve is justified, however. Let us consider the two scenarios: in the first scenario, no data is given to the model during training. This model will essentially guess a class for each pixel. Given that we use a binary classifier it will, on average, guess that a pixel belongs to the target class around 50% of the time. Thus, the average union of model prediction and ground truth will consist of more than 50% of the pixels in an image - 50% from prediction, and some number of pixels from the ground truth. However, the intersection will be considerably smaller - around 50% of the pixels in ground truth. For example, if a ground truth image of size $512px * 512px$ would have a $100px * 100px$ area of target class in it, and the model would have a uniform guess, the IOU score should be approximately

$$\frac{100 * 100 * 0.5}{100 * 100 + 412 * 412 * 0.5} \approx 0.053.$$

Let us then consider the second scenario, where we have the perfect model. This model would have an IOU score of one - the maximum IOU allows. As such, the curve has an absolute plateau at 1. It is also believable that giving more data to an almost perfect model is less effective than giving more data to a bad model; a good model already knows the basic shapes and needs to learn the weird situations, which is hard. As such, the idea of smooth parabolic climb to the plateau is justified.

One can also notice from Figure 6.1 that the models in invasive alien species task do not seem to enjoy similar gains as the ones in biomass estimation. Due to the problems of data listed in Section 5.1.1, one should be careful when drawing any conclusions from this. Invasive alien species task didn't have nearly the amount of data seen in the best cases of biomass estimation tasks, which further exacerbates the problem. However, it is also possible that the invasive alien species task is more difficult for the U-net than biomass estimation. In biomass estimation, the single instances of target classes are usually a separate colour compared to their background (ie. green on a grey/blue/brown lake) and have clearly distinguishable shapes. In invasive alien species task, the target class might well share a colour between its background (ie. green leaves of *Rosa rugosa* and the grass next to it).

Yet a third interpretation of Figure 6.1 is that the problems regarding the dataset listed in Section 5.1.1 might not be as critical as first thought. Even with somewhat problematic annotations, the models with good amount of examples enjoyed good results. The annotations do still create a little bit of tension in calculating the IOU scores, however: even if the model would have learned to avoid lake pixels next to target class, the IOU calculation would punish it for not also colouring said pixels. In the general picture, this does not seem to matter much.

As a conclusion, neural networks with U-net architecture still seem a promising method for achieving the two given tasks. Especially the top three models in biomass estimation task seem to yield reasonable results. However, this thesis could not yet demonstrate the usability of machine learning techniques beyond a reasonable doubt. To make this claim, more data is needed. By looking at the Figure 6.1, it looks like 3000-4000 512x512 tiles per class would be an ideal target for future data gathering. If this data is collected, it would be a good idea to provide the annotators with more robust methods of annotating the data. One such tool would be to pre-select colour ranges (typically green) in biomass estimation task, which would result in more accurate annotations and less frustrating annotation process for the annotators. Furthermore, it would interesting to research annotation fatigue and ways to combat it; annotation is a tedious, repetitive task for a human to do, but vital for machine learning tasks. One such research setting was introduced towards the end of Section 5.1.1. Essentially, the research question would consider creating annotations similar to the ones seen in this thesis. The hypothesis to be tested would be that presenting annotators with multiple smaller images yields better annotations than presenting few smaller images due to annotators not feeling overwhelmed by a single task. Initially, this hypothesis seems to agree with results in motivational psychology. However, it would also be interesting to see how fast the annotation process is done. It could be that the annotators who are given the big pictures annotate more images in a given time period, and if they annotate the images well enough, the vague annotations might overcome the few well-done annotations by sheer volume.

References

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

[2] alejo. imgaug. https://github.com/aleju/imgaug. Version 0.4.0.

[3] Alexander Mordvintsev & Abid K. Image segmentation with watershed algorithm. https://docs.opencv.org/master/d3/db4/tutorial_py_watershed.html. Retrieved: 2020-02-13.

[4] Amazon.com. Aws sagemaker. https://aws.amazon.com/sagemaker/. Retrieved 31.7.2020.

[5] Aphex34. Max pooling with a 2x2 filter and stride = 2. https://en.wikipedia.org/wiki/Convolutional_neural_network#/media/File:Max_pooling.png. Retrieved 17.03.2019.

[6] R. A. ARMSTRONG. Remote sensing of submerged vegetation canopies for biomass estimation. *International Journal of Remote Sensing*, 14(3):621–627, 1993.

[7] S. Beucher and F. Meyer. The morphological approach to segmentation: the watershed transformation. *Mathematical morphology in image processing*, 34:433–481, 1993.

[8] BVDrone Oy. BVDrone Oy. https://www.bvdrone.com/. Retrieved 31.7.2020.

[9] P. Cassey, T. M. Blackburn, R. P. Duncan, and S. L. Chown. Concerning invasive species: Reply to Brown and Sax. *Austral Ecology*, 30(4):475–480, 2005.

[10] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. 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, 2018.

[11] H.-D. Cheng, X. H. Jiang, Y. Sun, and J. Wang. Color image segmentation: advances and prospects. *Pattern recognition*, 34(12):2259–2281, 2001.

[12] M. F. Chislock, E. Doster, R. A. Zitomer, and A. E. Wilson. Eutrophication: Causes, consequences, and controls in aquatic ecosystems. *Nature Education Knowledge*, 4(4):10, 2013.

[13] F. Chollet et al. Keras. https://keras.io, 2015. Version 2.2.4.

[14] D. Cirecsan, U. Meier, L. M. Gambardella, and J. Schmidhuber. Deep big simple neural nets excel on hand-written digit recognition. *arXiv: 1003.0358 v1*, 2010.

[15] COCO Consortium. Coco 2015 object detection task. https://cocodataset.org/#detection-2015. Retrieved 31.7.2020.

[16] COCO Consortium. Coco 2016 object detection task. https://cocodataset.org/#detection-2016. Retrieved 31.7.2020.

[17] R. I. Colautti and H. J. MacIsaac. A neutral terminology to define 'invasive' species. *Diversity and Distributions*, 10(2):135–141, 2004.

[18] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[19] Environmental Systems Research Institute. Arcgis. https://www.arcgis.com/index.html.

[20] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2009 (VOC2009) Results. http://www.pascal-network.org/challenges/VOC/voc2009/workshop/index.html. Retrieved 31.7.2020.

[21] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2010 (VOC2010) Results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html. Retrieved 31.7.2020.

[22] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html. Retrieved 31.7.2020.

[23] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. Retrieved 31.7.2020.

[24] A. C. Fredrik Lundh et al. Pillow. https://python-pillow.org/. Version 6.0.0.

[25] K.-S. Fu and J. Mui. A survey on image segmentation. *Pattern recognition*, 13(1):3–16, 1981.

[26] J. Garson. Connectionism. In E. N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2018 edition, 2018.

[27] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness, 2018.

[28] Geoffrey Hinton. Lecture 6e RMSProp: Divide the gradient by a running average of its recent magnitude. http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf. Retrieved 29.07.2020.

[29] J. M. Gonfaus, X. Boix, J. van de Weijer, A. D. Bagdanov, J. Serrat, and J. Gonzàlez. Harmony potentials for joint classification and segmentation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3280–3287, 2010.

[30] A. Gonzalez-Garcia, D. Modolo, and V. Ferrari. Do semantic parts emerge in convolutional neural networks? *International Journal of Computer Vision*, 126(5):476–494, 2018.

[31] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[32] V. Grau, A. Mewes, M. Alcaniz, R. Kikinis, and S. K. Warfield. Improved watershed transform for medical image segmentation using prior information. *IEEE transactions on medical imaging*, 23(4):447–458, 2004.

[33] R. Harrabi and E. B. Braiek. Color image segmentation using multi-level thresholding approach and data fusion techniques: application in the breast cancer cells images. *EURASIP Journal on Image and Video Processing*, 2012(1):11, 2012.

[34] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[35] D. Hoiem, A. N. Stein, A. A. Efros, and M. Hebert. Recovering Occlusion Boundaries from a Single Image. *IEEE 11th International Conference on Computer Vision*, 1 2007.

[36] IntelLabs. River tail documentation. http://intellabs.github.io/RiverTrail/tutorial/. Retrieved 17.03.2019.

[37] ISBI. ISBI cell tracking challenge. http://celltrackingchallenge.net//. Retrieved 31.7.2020.

[38] N. R. Jachowski, M. S. Quak, D. A. Friess, D. Duangnamon, E. L. Webb, and A. D. Ziegler. Mangrove biomass estimation in Southwest Thailand using machine learning. *Applied Geography*, 45:311 – 321, 2013.

[39] Josef Steppan. Example MNIST images. https://commons.wikimedia.org/w/index.php?curid=64810040. Retrieved 29.07.2020.

[40] P. Kaiser, J. D. Wegner, A. Lucchi, M. Jaggi, T. Hofmann, and K. Schindler. Learning aerial image segmentation from online maps. *IEEE Transactions on Geoscience and Remote Sensing*, 55(11):6054–6068, 2017.

[41] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[42] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[43] Laughsinthestocks. Activation funtions. https://en.wikipedia.org/wiki/Activation_function. Retrieved 14.03.2019.

[44] A. Lehmann, J.-M. Jaquet, and J.-B. Lachavanne. Contribution of GIS to submerged macrophyte biomass estimation and community structure modeling, lake Geneva, Switzerland. *Aquatic Botany*, 47(2):99–117, 1994.

[45] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017.

[46] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[47] S. Ltd. Affinity photo. https://github.com/aleju/imgaug. Version 1.8.

[48] R. N. Mack, D. Simberloff, W. M. Lonsdale, H. Evans, M. Clout, and F. A. Bazzaz. Biotic invasions: Causes, epidemiology, global consequences, and control. *Ecological Applications*, 10(3):689–710, 2000.

[49] M. A. Nielsen. *Neural Networks and Deep Learning*. Determination Press, 2015. http://www.neuralnetworksanddeeplearning.com.

[50] NumPy team. NumPy. https://numpy.org. Version 1.16.3.

[51] Nvidia Corporation. Accelerating hyperscale data center applications with Tesla GPUs. https://developer.nvidia.com/blog/accelerating-hyperscale-datacenter-applications-tesla-gpus/. Retrieved 31.7.2020.

[52] Nvidia Corporation. GeForce GTX Titan (6GB). https://www.geforce.com/hardware/desktop-gpus/geforce-gtx-titan/specifications. Retrieved 31.7.2020.

[53] D. Opitz and S. Blundell. *Object recognition and image segmentation: the Feature Analyst® approach*, pages 153–167. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[54] N. R. Pal and S. K. Pal. A review on image segmentation techniques. *Pattern recognition*, 26(9):1277–1294, 1993.

[55] Parliament of Finland. Act on managing the risk caused by alien species. https://www.finlex.fi/en/laki/kaannokset/2015/en20151709, 2015. Retrieved: 2019-07-25.

[56] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. Version 0.19.0.

[57] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[58] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[59] Rewilding Europe. Bison rewilding plan. https://rewildingeurope.com/rewilding-in-action/wildlife-comeback/bison/. Retrieved: 2020-02-13.

[60] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, page 234–241, 2015.

[61] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[62] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *International Journal of Computer Vision*, 81(1):2–23, 2009.

[63] T. Silva, M. Costa, and J. Melack. Assessment of two biomass estimation methods for aquatic vegetation growing on the amazon floodplain. *Aquatic Botany*, 92:161–167, 04 2010.

[64] M. Subramanian. Anthropocene now: influential panel votes to recognize earth's new epoch. https://www.nature.com/articles/d41586-019-01641-5, 2019. Retrieved: 2019-07-31.

[65] Textron Systems. The feature analyst. `https://www.textronsystems.com/products/feature-analyst`.

[66] The European Parliament and Council. The european parliament and the council regulation no 1143/2014 on the prevention and management of the introduction and spread of invasive alien species. https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32014R1143&from=EN. Retrieved: 2019-07-31.

[67] The Finnish Government. Kansallinen vieraslajistrategia. http://vieraslajit.fi/sites/default/files/Vieraslajistrategia_web.pdf#overlay-context=fi/node/27. Retrieved: 2019-07-31.

[68] The Gimp Team. Gnu image manipulation program. https://www.gimp.org/. Version: 20.10.14.

[69] The Keras team. Keras / metrics.py. https://github.com/keras-team/keras/blob/master/keras/metrics.py. Retrieved: 2020-05-07.

[70] The OpenCV Team. Opencv. https://opencv.org/. Version: 3.4.9.

[71] M. Thoma. A survey of semantic segmentation. *arXiv preprint arXiv:1602.06541*, 2016.

[72] M. Treml, J. Arjona-Medina, T. Unterthiner, R. Durgesh, F. Friedmann, P. Schuberth, A. Mayr, M. Heusel, M. Hofmarcher, M. Widrich, et al. Speeding up semantic segmentation for autonomous driving. In *MLITS, NIPS Workshop*, volume 2, page 7, 2016.

[73] Y. Yang, S. Hallman, D. Ramanan, and C. C. Fowlkes. Layered object models for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1731–1743, 2012.

[74] Yann LeCun & Corinna Cortes & Christopher J.C. Burges. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/. Retrieved: 2020-05-12.

[75] X. Zhang. On the estimation of biomass of submerged vegetation using landsat thematic mapper (tm) imagery: a case study of the honghu lake, pr china. *International Journal of Remote Sensing*, 19(1):11–20, 1998.

[76] Y. Zhang, M. Brady, and S. Smith. Segmentation of brain mr images through a hidden markov random field model and the expectation-maximization algorithm. *IEEE Transactions on Medical Imaging*, 20(1):45–57, 2001.

[77] C. L. Zweig, M. A. Burgess, H. F. Percival, and W. M. Kitchens. Use of unmanned aircraft systems to delineate fine-scale wetland vegetation communities. *Wetlands*, 35(2):303–309, 2015.