UNIL | Université de Lausanne
Faculté des géosciences
et de l'environnement

Master of Science in Geography

Convolutional neural networks for semantic segmentation of multi-scale remote sensing imagery: comparison of model architectures

Patrick Talon

Supervised by Dr. Christian Kaiser

Satellite Image: © ESA / Eurimage / Swisstopo, NPOC
Data : OFS, 2004

Maîtrise universitaire ès sciences en géographie | Août - 2018

Table of contents

Abstract

This master thesis addresses the question of how convolutional neural networks (CNN) can help to achieve better accuracies for semantic segmentation of remote sensing data, in a view of automated and decision oriented applications. The attempt to answer this question is made by comparing the results achieved by two distinct models inspired by the popular U-Net and SegNet architectures. The experiments will be conducted on two remote sensing datasets that have been captured at different geographic scales. Results show that there is no unique solution for semantic segmentation of remotely sensed data and that different levels of scale would require different architectures of CNN. In order to achieve acceptable level of accuracies for public or private decision oriented applications, CNNs require more imposing labelled data for training and more balance between the land cover classes.

Résumé

Ce travail de mémoire s'intéresse à la question de l'usage des réseaux de neurones convolutifs (CNN) pour d'obtenir de meilleurs taux de précision lors de la segmentation sémantique de données issues de la télédétection, dans une perspective d'applications automatisées et orientées vers la décision. La tentative de réponse à cette question s'est faite en comparant les résultats obtenus par deux modèles distincts inspirés des architectures U-Net et SegNet. Des tests ont été effectués sur deux ensembles de données de télédétection capturés à différentes échelles géographiques. Les résultats montrent qu'il n'existe pas de solution unique pour la segmentation sémantique des données de télédétection et qu'une analyse à différentes échelles nécessiterait différentes architectures de CNN. Afin d'atteindre un seuil de précision acceptable pour des applications publiques ou privées axées sur la décision, les CNN requièrent des quantités de données d'entrainement plus imposantes et un meilleur équilibre entre les classes qui composent la couverture du sol.

Acknowledgment

I would like to thank my master thesis supervisor, Dr. Christian Kaiser, Senior lecturer at the Institute of geography and sustainability of the University of Lausanne. Dr. Kaiser was constantly available to provide help in my research and answer my questions. Thanks as well to my partner Chezca Comillas who helped in the correction of the project in a short period of time. Finally, I would like to thank my family and friends for their constant support and advises over the last year.

Patrick Talon

1 Introduction

Urban space is subject to many experiments and debates in the idea of making it more pleasant and attractive on a daily basis and in a fair way, for all its users and inhabitants. In a small country like Switzerland, densification and reduction of urban sprawl rate are part of the main objectives and projects of urban planning. It is not easy for cities and towns to grow and flourish guaranteeing a quality habitat for each individual while taking into account environmental issues. The territory is a complex space resulting from dynamic socio-spatial processes and the analysis of the distribution of the features composing it can allow a better understanding of these spatial phenomena.

Several researches have already been carried out thanks to various tools and methods allowing the analysis of the territory with the classification of remote sensing imagery at pixel level, a technique more commonly referred as semantic segmentation. Histogram analysis of aerial and satellite images is the first technique that made possible accurate semantic segmentation of land cover. More elaborate techniques were then invented, especially in the field of machine learning, like support vector machine. More recently with the development of deep learning techniques, artificial neural networks (ANN) are experimented in different branches of science. A specific kind of ANN called convolutional neural network (CNN) is already recognized as state-of-art methods for computer vision. They are interesting in this context because of their relatively low costs and the speed at which they can process heavy and abundant data such as high-definition satellite images.

This type of technique allows by extension a characterization of the land use, and thus a classification of the territory into different categories. This provides a better understanding of the spatial organization and a thoughtful and relevant typology of the territory. Such analysis can be useful to planners and politicians in defining efficient, environmentally friendly and equitable land use policies.

The objective of this paper is to identify the advantages and disadvantages of using CNN for semantic segmentation of remote sensing imagery. The attempt

to achieve this will be made by comparing two different models of convolutional neural networks that are inspired by the U-Net and the SegNet architectures. Two different sources of data will be used and will be pre-processed in two different ways, which makes a total of four distinct datasets to be processed by the CNNs. The first data source is the Vaihingen dataset that is made of high resolution aerial orthophotos and their corresponding images with ground truth classification at pixel level. The second one is a satellite capture that covers the entire area of Switzerland, with an associated ground truth image that has been prepared specifically for this project.

In the first part, the problem statement is presented. A review of the literature focused on semantic segmentation and artificial neural networks is then made. The next section is more theoretical, providing the fundamental principles of ANN and deep learning, and explaining more specifically the U-Net and SegNet architectures that will be used and compared during this experiment. As the performances of artificial neural networks are highly correlated with the quality of the data, it is then necessary to describe the datasets in a statistical and qualitative manner to highlight its particularities. The results of the different semantic segmentation are then presented, and a discussion is made about the questions raised in the problem statement. A conclusion summarizes and put into perspective what has been accomplished in this paper.

Target readers are typically students who have a background in geography, and more particularly spatial analysis. It can also be a good way for anyone curious about ANN to be introduced to the main principles that relies behind this promising machine learning field, without the need of having a strong mathematical knowledge.

2 Problem statement

Remote sensing is involved in a wide scope of research and applications in geosciences. For a simple satellite photo of a targeted region, or more advanced analysis and monitoring of ecosystems, meteorological phenomena and land use, remotely sensed data is collected from many different sources and processed on a daily basis. Understanding the landscape is one of the historical objectives and purposes of geographers, and with the appearance of the spatial analysis paradigm in the middle of the 20th century, this innovative source of data allowed researchers to study Earth through new dimensions and at a brand new scale.

A wide range of methods and tools for remote sensed data analysis have been developed since then and spread through the scientific communities. It became possible to acquire a better understanding of the processes and dynamics that alter our direct and distant environment. For the first time in history, people were able to monitor their habitat at a global scale and observe natural and anthropic changes that could potentially have impacts on the ecosystems.

Remote sensing imagery can contain a lot of complex information. After the recent and commonly called “digital revolution”, an overflowing amount of digital data production occurred. It became necessary for researchers to develop new methods for efficient extraction of patterns from data, and accurate modelling of the underlying multivariate systems. At the same time, computers drastically gained in power and the prices for high performing systems significantly decreased. This allowed a fast and systematic data processing and analysis, essential for a good monitoring of environmental and socioeconomic phenomena.

Amongst the analysis methods applied to remotely sensed data, the segmentation and classification of digital image data at pixel level, commonly called semantic segmentation, might be one of the most frequently used nowadays. When applied to remote sensed data, each pixel of the original image is assigned to a specific semantic class that is supposed to represent it best.

Such classification helps to simplify and summarize the information in the perspective of detecting trends over space and time, and eventually decision making.

Most of recent semantic segmentation algorithms use pixel's RGB as well as non-visible electromagnetic radiations values as data input. The nature of the data can have a considerable impact on the results, as collection conditions are rarely similar. Thus, choosing the right dataset and adapted preprocessing methods are serious matters when it comes to work on a particular topic. When it comes to working on large amounts of data and phenomena that we do not have accurate model for, machine learning methods seem to be taking center stage since the middle of 20th century. The ability for a model to learn given a set of training samples and then automatically find patterns and features on new data has shown promising prediction results with various kinds of classification and regression tasks. It is thus quite naturally that geosciences researchers started to explore and develop machine learning techniques that were able to deal with remote sensing data. Artificial neural networks are one of them. They are designed to deal efficiently with multivariate non-parametric systems and find patterns in a large amount of data thanks to supervised learning.

Artificial neural networks seek to make a computer achieve operations that are inspired by biological brain specific functions: detect simple features in an environment that can be combined to generate complex and high level representations, in order to recognize scenes or objects in new environments. This is a classification task that can typically be applied to visual and sound perceptions. In order to achieve that, the biologic brain needs to learn and build an experience from all the "data" that has already been processed, using hierarchical and connected layers of neurons that operates simple computations on an input signal to produce a specific output that will be transmitted to other neurons. This whole process allows the brain to build representations in order to make sense of all the information that comes from the environment. Given a reasonable amount of digital training data, an artificial neural networks can automatically perform similar tasks, like image and speech recognition. Each layer of an artificial neural network is made of simple computational units, the

artificial neurons that have a specific task, and are in general all connected to every neurons in a following layer. The learning operates as the data goes through the layers and the network adjusts its weights and bias to minimize an error function in order to obtain the desired output. Such algorithm can be applied to remote sensing data in order to produce a land cover classification in the form of a semantic segmentation.

Training a neural network with an aerial images dataset represents a heavy task in terms of memory for a computer and can be hardly achievable. The solution to this problem is the convolutional neural network (CNN), specially designed for image recognition. This kind of technique belongs to a specific machine learning branch called “deep learning”, that has recently gained in interest for the researchers. It consists of artificial neural networks that are designed with a lot of layers of artificial neurons that are hierarchically organized. When using convolution layers, the number of entries that each neuron has to process is lessened, keeping at the same time the local correlations of pixel values. The first layers of the network can detect low level features of the data, and the representation level increases when the layers are deeper. The learning is called “deep” because of that long sequence of layers that processes the data.

CNN have already been tested on remotely sensed data to produce land use classification by semantic segmentation, and is giving good results as of today. It seems however not being in practical use yet for this kind of data, as institutions and companies may not be ready to use this new technology. Investments in terms of time and money are necessary for a significant transition to these new methods and labelled data is not always easily available.

The key challenges to generalize the use of CNN for semantic segmentation of remote sensing imagery are numerous and raise many questions. Considering the variations in data collection methods, is it possible to train and use a single CNN and keep a good accuracy over time? Scale is a persistent challenge in geography. How do CNN react to variation of scale in the data? And is the state-of-the-arte accuracy achieved by CNN good enough for blind trust in the outputs of land cover classification and for decision making?

The research question for this project can be formulated as follow: To what extent can convolutional neural networks perform efficient and accurate semantic segmentation on multi-scale remote sensing imagery with a view of making an automated and decision oriented application?

Some guiding hypothesis about semantic segmentation with CNN on remote sensing data are identified as a leading research axis for this general question. Considering what is being said in most of the recent literature (Castelluccio, Poggi, Sansone, & Verdoliva, 2015; Fu, Liu, Zhou, Sun, & Zhang, 2017; Längkvist, Kiselev, Alirezaie, & Loutfi, 2016; Marmanis et al., 2016; Volpi & Tuia, 2016), it is generally admitted that CNN are accurate models for pixel-wise classification of remote sensing imagery. They became efficient alternative in terms of time and costs for decision oriented applications as related technologies have substantially gained in accessibility and user-friendliness. But no method is perfect for accomplishing every kind of tasks and CNN are still facing a few challenges. Since they became a global phenomenon, many different model architectures have been proposed (Badrinarayanan, Kendall, & Cipolla, 2015; Long, Shelhamer, & Darrell, 2015; Ronneberger, Fischer, & Brox, 2015) and often only for particular datasets. In remote sensing analysis, scale is an important aspect and the ability of CNN to perform efficiently at geographic multi-scale level is not really discussed in the literature. It is noticeable in the different experiments that CNN have difficulties to reproduce “sharp” geometries during semantic segmentation, and this can represent a challenge especially for land cover when trying to classify human built structures. The question of consistency in data acquisition process is also central and it is not guaranteed that today’s CNN architectures will still be effective for next generation remote sensing data.

3 Literature review

This review of the literature gives a particular attention to the history of semantic segmentation techniques on remote sensing data, including more advanced and recent works on neural networks. The first part will focus on remote sensing and their global implication that their generalization has for geosciences. The second part reviews image segmentation techniques. For each mentioned approach, a link to remote sensing will be made in order to understand how the various paradigms led to the usage of artificial neural networks for dense pixel-wise prediction from remotely sensed data. The last part will focus on convolutional neural networks and will describe the state of the art regarding the theoretical framework used in the context of this paper.

3.1 Remote sensing

Remote sensing can be defined as the science of collecting data about an object of study without being in direct contact with it, i.e. from a distance (Richards & Jia, 2006). It is most of the time associated to aircraft and satellite platforms that use passive or active sensors to collect georeferenced image data of the Earth surface and the atmosphere. An active sensor works like a radar and sends its own signals that interact with a surface and bounces back, returning measures. A passive sensor responds to emissions that come from the reflection of the solar radiation and captures an image of the targeted area for a wide range of the electromagnetic spectrum. In the case of satellite platforms, the raw data is transmitted electronically to a receiving station located on the surface of earth. For aircrafts, the data is collected after landing.

Remotely sensed data has a few properties that are important to keep in mind when working with it. Digital image is always captured with predetermined spatial and spectral resolutions. Also the characteristics of the raw data is very different depending on which platform and which sensor is used (Lu & Weng, 2007). The scientist must therefore choose the right source of remote sensing data when exploring a specific problematic and must be aware of the scale aspect. It is possible to transform an image to a lower resolution, but there are several ways to do so, and they all have an influence on the outputs of an analysis. Brightness,

shadows, nebulosity, atmospheric and topographic conditions in general can have a great impact on the quality of the data, and generate bias that requires a lot of pre-processing work in most situations (Young et al., 2017).

Even if the term “Remote sensing” appeared in the 60’s, it all started in the middle of the 19th century, with a simple photography taken from a balloon (Kitchin & Thrift, 2009). The usage of this technique became a strategic tool during the First World War with photographs being taken from an airplane. Many other applications were developed later. In the 60’s, the first satellites were launched in space and allowed a global monitoring of earth’s surface. With that new abundant source of data and the digital revolution, researchers worked on more and more efficient digital image processing techniques. Nowadays, analysis of heavy imagery data has become even more accessible mainly because of two factors. The first one is the large increase in the number of satellites that are orbiting around the planet, providing countless of data to work on. The second one is the global increase in the number of highly performing computing devices that became available for private individuals and at reasonable prices in the retail as well as on the cloud.

Geosciences were greatly impacted by the advent of remote sensing (Kitchin & Thrift, 2009). Various applications like natural resources monitoring and weather forecast are used on a daily basis and people are making use of remotely sensed data sometimes without realizing. Spatial analysis might be the most affected field in the sense that Earth surface data could be captured at brand new scales, more frequently and effortlessly. Methods for land use classification were quickly developed to reach until today an almost automated process and researchers are still working on it with the shared objective of reaching higher classification accuracy.

3.2 Computer vision

Computer vision, or machine vision, is the field of Artificial intelligence that seeks to make computer automatically able to “see” like humans and recognize objects and scenes in their context and environment, for 2-D as well as 3-D content

(Szeliski, 2011). Given an image or a video, the key challenge for the algorithm is to create representations at multiple levels of abstraction and recognize the various features that composes the image to produce a segmentation in order to isolate the desired features. Where the task seems easy to achieve from a human point of view, it is complex to translate it to machine language and plenty of ways are described in the literature. From automated waste sorting to self-driving cars, the number of possible applications has been growing faster recently, mainly thanks to the diffusion of artificial neural networks techniques that allow close to real-time analysis of digital images.

3.3 Semantic segmentation

Image segmentation is the art of dividing an image into non-intersecting subsets, considering predefined criteria (Cheng, Jiang, Sun, & Wang, 2001.; Dey, Zhang, & Zhong, 2010; Dhanachandra, Manglem, & Chanu, 2015; Haralick & Shapiro, 1985). With semantic segmentation, or dense pixel-wise labeling, each pixel of the image is assigned to a specific semantic class. It is qualified as “semantic” as the attempt is made to give a meaning to the pixels. Semantic segmentation could therefore be defined as the art of understanding the role of each pixel within an image in order to label it as belonging to a specific object class (Thoma, 2016).

Haralick and Shapiro (1985) introduced their paper with a definition of what a good result for image segmentation should be to their sense. Their ideas could be summarized with the following points:

- Homogeneity in the segmented regions.
- No “rings” or “holes” in the segmented regions.
- Adjacent segmented regions should have significantly different properties
- Boundaries of the segmented regions should be simple and smooth.

These few statements are a good mean to realize what are the main difficulties of pixel-level image segmentation and should be kept in mind when evaluating the results obtained by the various methods that will be introduced next.

3.4 Conventional techniques

The very first techniques for semantic segmentation were mainly using the histogram of greyscale images, where thresholds could easily be set to segregate the pixels into two or more subsets depending on the intensity levels of the image. This task can easily be performed if the distribution of the pixel values is not normally distributed, and “valleys” appear in the histogram, which can be used as thresholds. In reality, this kind of method is rarely accurate as the different features composing an image are almost always mixed into a normal distribution (Pal & Pal, 1993; Thoma, 2016; Yuheng & Hao, 2017). Thus, some features from a targeted object in the picture can be mixed up with features from the background or from another object. The same issue occurs when the image contains noise, resulting in a poor global coherence in the segmentation.

Another traditional way of segmenting an image is edges detection. An edge can be considered as a locally abrupt discontinuity in neighboring pixels values, which is often an actual boundary between real features composing an image (Cheng, Jiang, Sun, & Wang, 2001; Pal & Pal, 1993; Yuheng & Hao, 2017). The edges can be highlighted thanks to the use of filters, then isolated with some post processing techniques. One of the common problems encountered with this method is that the results proposed by edge detection algorithms can seem unnatural when compared to what an actual human eye would detect. It is also complicated to classify features of an image using their boundaries only, thus ignoring other properties of the image that can be useful in such discrimination task.

Conventional techniques can operate well enough on greyscale images that contain simple objects. However, the needs of contemporary society require segmentation of color images as they are a source of a lot of additional high level information. This kind of data requires more advanced algorithm in order to be processed.

3.5 Machine learning

More recently, the field of artificial intelligence is being widely explored, and some researchers started to use machine learning for image classification. Machine learning is the art of making a computer learn from a reference dataset and create a model of the underlying phenomena that produced this particular data. When the model is correctly trained, it is supposed to be able to apply well on new data. Machine learning is a wide field that gathers many and various kind of methods, but all of them can be distributed within two groups: unsupervised and supervised learning models.

Unsupervised models are designed to learn from the statistical characteristics of the training data directly. The reference dataset does not need to be labelled, and the learning algorithm finds patterns by itself. These models are useful to find trends in data for phenomena that are not well known and identified. Low level features of an image are often directly related to RGB values and related image metrics such as brightness, contrasts or textures and can be relatively well segregated with clustering algorithms like K-means and Random Forest Classifier, as well as variations of them (Dhanachandra, Manglem, & Chanu, 2015).

Supervised learning models are based on algorithms that can learn from labelled reference dataset. Data is labelled when the user has assigned a target class to each sample before feeding the learning algorithm with it. The objective is to produce a specific output for a specific input by minimizing the error rate during the training stage. To do so, the algorithm has the ability to adjust weights and parameters, until global accuracy of the output approaches a maximum. This represents the main challenge of computer vision as of today, and it is generally admitted that good image segmentation results are obtained with supervised learning models like support vector machines (Chapelle, Haffner, & Vapnik, 1999; Wang, Wang, & Bu, 2011), or K-nearest neighbors (Bieniecki & Grabowski, s. d). Application of all these methods to remote sensing data for semantic segmentation have been successfully carried out by the research community (Huang, Davis, & Townshend, 2002; Mountrakis, Im, & Ogole, 2011), but

appeared to be less efficient for high resolution imagery where low to high dimensional information needs to be extracted (Fu, Liu, Zhou, Sun, & Zhang, 2017). The detection of such features requires more advanced techniques that are less sensible to noise and able to understand various organizations of simple features into more complex ones.

3.6 Artificial neural networks

Amongst supervised machine learning techniques, ANN have recently recorded an increase of occurrences in the literature for image segmentation and for various branches of science. The simplest kind of ANN is the perceptron (Rosenblatt, 1958), which represent the very first and fundamental ANN. It is composed of a unique artificial neuron that produces a binary output decision. All the ANN architectures that followed since then are based on the same principles. In their recent form, ANN are made of computational nodes, the artificial neurons, that are connected and arranged in layers that are able to learn from large training dataset by adjusting weights and biases in order to minimize an error function (Haykin, 1999). The main difference with other supervised machine learning techniques is that this is done without any task-oriented coding prerequisites; the network calibrates itself to produce the desired output for a specific data and human's involvement is minimal. In its simplest form, the first layer represents the normalized input data, the classification decision happens in the last layer of the network, and one or more layers in-between called hidden layers build low to high level representations in order to recognize simple patterns in the data and combines them into more complex ones. All the power of a neural network comes from its ability to generalize and predict well on data that it has never seen. ANN are not recent and their potential was already identified years ago, but stayed in the background because of lack of training data and computing power.

Early experiments with remote sensing imagery resulted in a globally poor accuracy compared to other conventional methods (Mas & Flores, 2008), mainly because land cover classification usually needs to be done at large scale with heterogeneous high resolution data. This causes a regular ANN to struggle with the amount of computation required, especially when the classification is done at

pixel-level. Recent progresses in the field allowed a more efficient semantic segmentation with ANN. Since the 2010's researchers are working on a specific type of ANN based methods that became an important subfield of machine learning called deep learning. The learning delivered by an ANN's architecture is considered deep when the number of layers is significantly increased, allowing multiple levels of abstraction and non-linear operations in order to understand that data. Nowadays, many different architectures of deep neural networks are being experimented and suggested for many kinds of application. For computer vision and image classification tasks in general, CNN are generally admitted as state-of-art method.

CNN are specifically designed for image classification. Convolution applied to digital images allows to extract the important features, while reducing its dimensionality and increasing its depth. This is achieved with the usage of successive layers of convolution filters and pooling for subsampling. CNN first showed impressive results for image categorization tasks like recognition of handwritten digits (LeCun, Bottou, Bengio, & Haffner, 1998) and later other image classification tasks (Krizhevsky, Sutskever, & Hinton, 2017), and worked pretty well on remote sensing data for scene classification (Castelluccio, Poggi, Sansone, & Verdoliva, 2015; Yu, Wu, Luo, & Ren, 2017).

CNN were then adapted to produce semantic segmentation by adding up-sampling, or transposed convolution layers between the convolution and the final classification in various architectures of CNN (Chen, Papandreou, Kokkinos, Murphy, & Yuille, 2016; Long, Shelhamer, & Darrell, 2015; Pinheiro & Collobert, 2015). This process allows to map the input image to an input with the same size, performing a semantic classification at pixel level. It didn't take long until researchers explored the use of this new technique on remote sensing data and many different architectures of CNN have been suggested (Fu, Liu, Zhou, Sun, & Zhang, 2017; Marmanis et al., 2016; Volpi & Tuia, 2016).

U-Net is a famous example that was originally made for greyscale microscopic scale biomedical image segmentation (Ronneberger, Fischer, & Brox, 2015). U-Net architecture was designed in 2015 to give a binary output to each pixel of the

input image, in order to highlight particular structures. It was then improved and adapted for more than two labels classification tasks.

One of the most cited application examples nowadays are the autonomous driving systems that have the capacity to analyze road scene in almost real-time, and identify the various objects and elements in its environment that gives the possibility to the self-driving system to take decisions by itself. SegNet is a popular architecture inspired by the VGG16 (Simonyan & Zisserman, 2014) that performed accurate semantic segmentation on the road scene images and videos of the CamVid dataset (Badrinarayanan, Kendall, & Cipolla, 2015).

U-Net and SegNet are the two architectures that have been chosen for the present project as they often appear in the literature and seem adapted to semantic segmentation of remote sensing data. They use different architectures and different methods for up-sampling and pixel-wise classification.

4 Artificial neural networks theory

ANN are inspired by the functioning of the biological brain in the processes of recognition. Even if these processes are not yet fully understood by scientists yet, a few major principles have been identified and are commonly admitted in the field of neurosciences. ANN do not pretend to be a simulation of the animal brain, but the main concepts are inspired by these key principles.

4.1 Artificial neuron

The neuron is the main unit of the central nervous system. It is made of four fundamental elements: the dendrites, the nucleus, the axon and the synapses. Information in the form of electrical signals are transmitted from the dendrites to the nucleus of the neuron in order to be processed. A new signal can then be fired and transmitted via the axon to adjacent neurons that is connected by the synapses and so on. The connections between neurons can have different intensities and the information going through is impacted by it. This organized and connected network of neurons allows its host to capture and identify stimuli from his environment, which makes interaction with it possible.

An artificial neuron

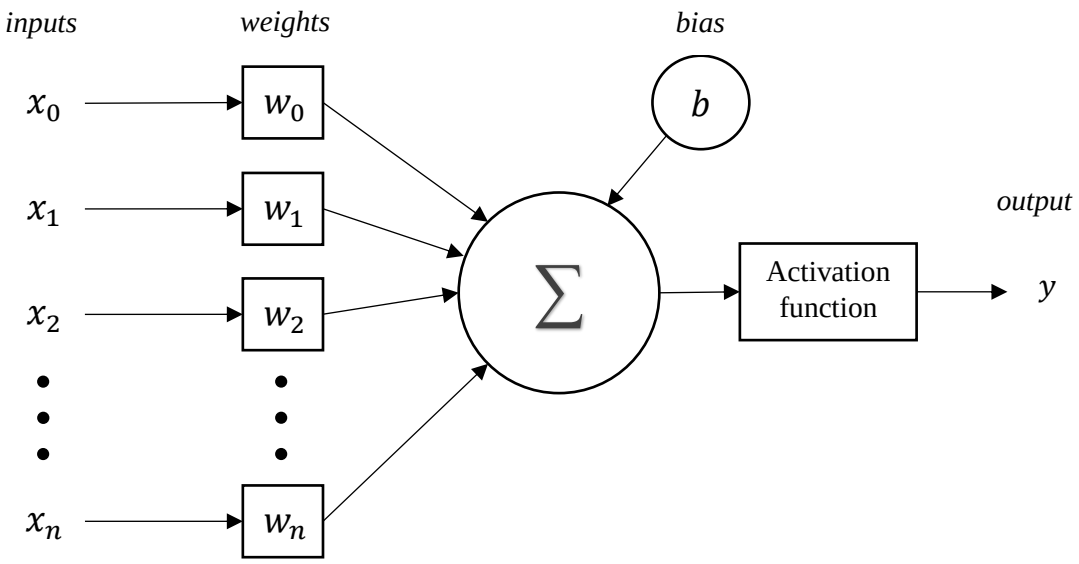


Figure 1

By analogy, an artificial neuron is the fundamental unit of the ANN. It is represented in *Figure 1*. The whole operation can also be transcribed in a relatively simple equation (*Equation 1*).

The neuron receives one or more weighted input data in the form of $(w_i * x_i)$, for example pixel values, that are summed up. These weights w_i are one of the parameters that are adjustable and participate in the learning. Usually a bias b is then added to this sum. The result goes through an activation function $\varphi()$ that decides if the neuron can fire and transmit a signal y to other artificial neurons.

$$y = \varphi(\sum_{i=0} (w_i * x_i) + b)$$

Equation 1

The activation function can be of many forms but some are more common. The usual given example is the basic threshold function, or step function *(Figure 2)*, that gives a value of 0 or 1 to the artificial neuron's output. 0 means that the neuron won't fire, 1 means that it will.

Threshold function

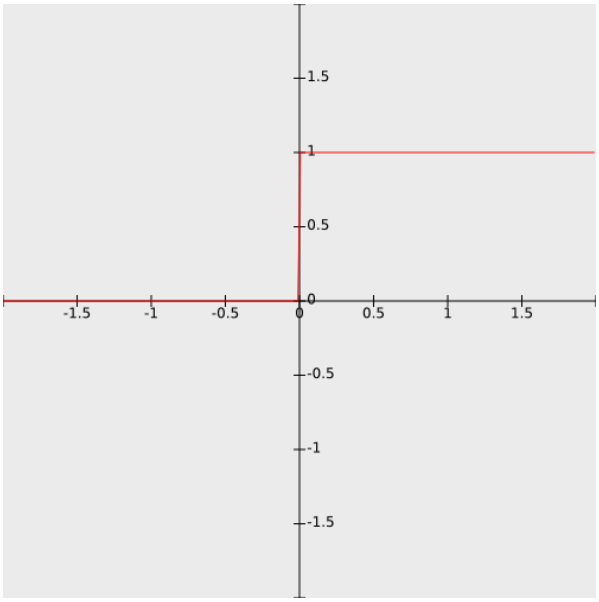


Figure 2

This function was initially used in the perceptron, the very first model of the most basic artificial neuron. This step function becomes limited when the desired output is not of binary form. It is more common nowadays to work with ANNs that need to perform a classification decision for multiple outputs and it is therefore

more convenient to obtain from the network a percentage of activation for each neuron, and a set of probabilities for all the possible outputs. This can be done for instance with the Softmax function (*Figure 3*).

Softmax function

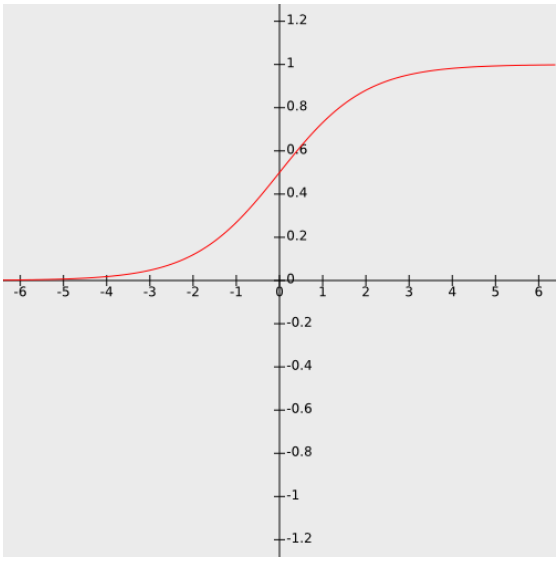


Figure 3

This function also introduces non-linearity to the network which allows to increase the levels of representation that can be built by stacking multiple layers of artificial neurons. It is also a tool of normalization for the ANN as it squeezes its output between 0 and 1.

ReLU function

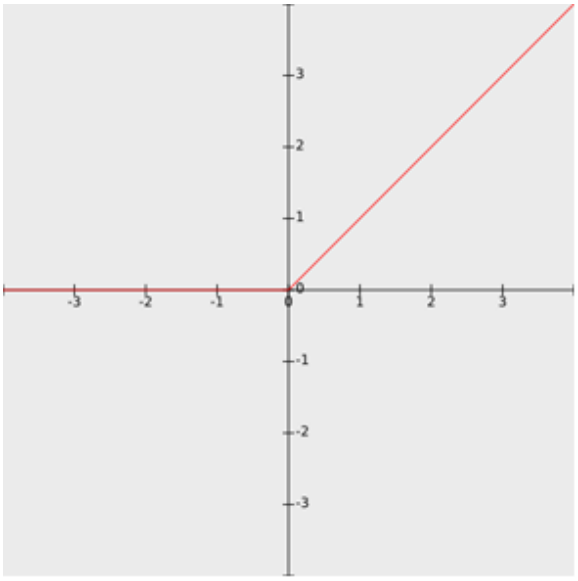


Figure 4

More recently and specifically for semantic segmentation with CNN, another type of activation function is used: the rectified linear unit (ReLU). This non-linear function, plotted in *Figure 4*, gives a value of 0 to any negative input. In the context of CNN, this function is interesting as it is computed faster, and about 50% of the neurons will not be activated, thus reducing significantly the global computational heaviness of the network and hence the time required for the training.

4.2 Artificial neural networks

An artificial neuron is a relatively simple computational unit and the task it can perform does not require a lot of energy. When these neurons are abundant and interconnected, they can interact and accomplish much more complex tasks. *Figure 5* shows an example of a simple artificial neural network. The neurons are organized in layers that are connected in a parallel structure. In general, a neuron from a layer is connected to every neurons of the next layer. A connection means that the output of a neuron becomes the input of another one and the data flow goes in one direction that is generally represented as going from left to right in the figures that can be found in the literature. There are three different types of layers. The first one is the input layer that feeds the raw data to the network. The second type is the hidden layer. There can be many of these in an ANN, and these are basically where most of the learning happens. In general, the deeper a layer is (i.e. the closest to the output layer), the higher the abstraction level. The last kind of layer is the output layer, which contains as many neurons as the number of targeted labels for classification. This layer is where the classification decision probabilities are made for each input. To each connection in the network is associated a unique weight that gives a greater or lesser importance to the signals emitted from a neuron to other ones. Biases are the other learnable parameters used in the hidden and output layers that gives more flexibility for fitting the data. Both weights and biases are initially randomly generated numbers that are adjusted by the learning algorithm during the training phase. These parameters store the experience acquired thanks to a labelled training dataset and are responsible for the knowledge that the network acquires.

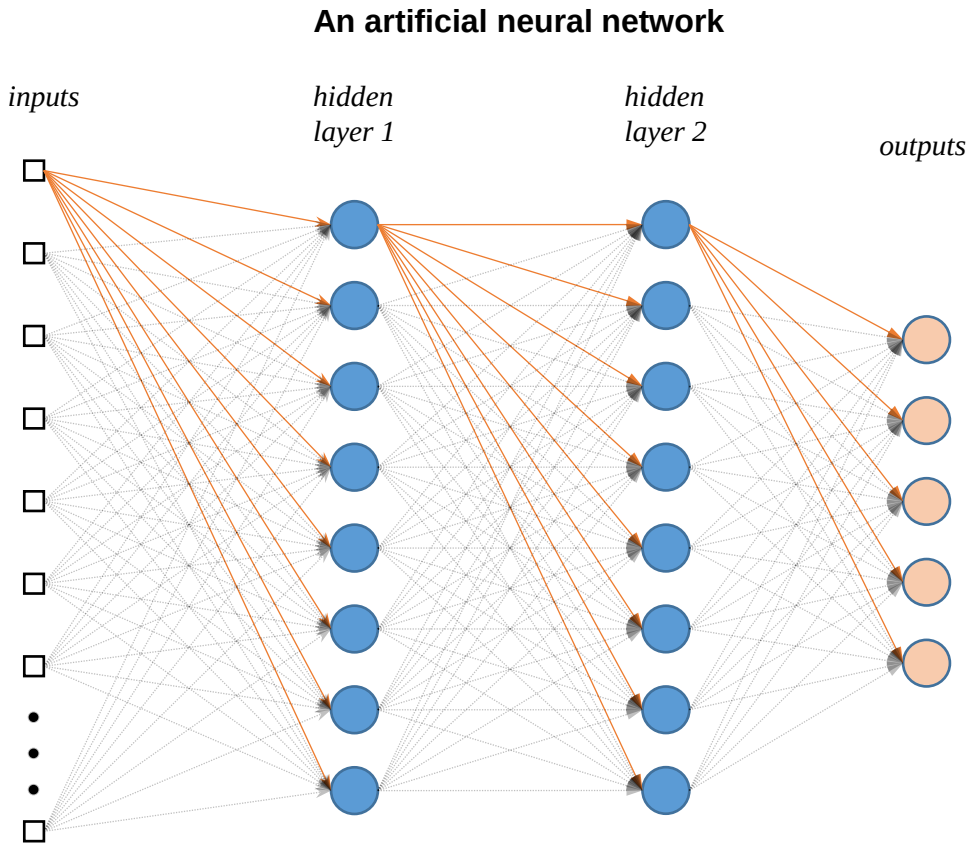


Figure 5

4.2.1 Learning algorithm

For the learning to happen, an adjustment of the weights and biases is done multiple times for each given sample of the training dataset, in order to obtain the desired classification output. This process is often referred as an input-output mapping. The adjustments are not randomly made but managed by the learning algorithm. In the first stage, the algorithm computes how badly the network is performing by comparing during the training each predicted output to a ground truth classification associated to the training input sample. This comparison is performed by computing the total error made by the output nodes.

$$E_{o1} = \frac{1}{2}(T_{o1} - A_{o1})^2$$

Equation 2

In *Equation 2*, the squared error E of the output neuron $o1$ is calculated by subtracting the actual its output A_{o1} to the targeted output T_{o1}. This result is then squared and multiplied by $\frac{1}{2}$. This is repeated for all the output neurons O of the

network and all these errors are then summed up to give the total error for one training sample, as per *Equation 3*.

$$E_{tot} = \sum_{i=1}^{O} E_{o_i}$$

Equation 4

In the context of ANN, this is usually referred as the cost function, which can be seen as a tool for monitoring the learning performance of the network. The objective of the ANN is to adjust itself in order to have all the A_o getting closer to the T_o, which comes to find the minimum of the cost function.

The only possible way for the network to adjust itself is to update its weights and biases. This can be achieved thanks to gradient descent. Gradient descent algorithm computes the slope at any point of the function thanks to the derivative of E_{tot} with respect to a weight w_i.

$$G = \frac{\partial E_{tot}}{\partial w_i}$$

Equation 5

This slope is then used as an indicator showing the direction where the lowest local point of the curve is. A step of a size determined by the learning rate η is made towards that direction and this represents an update w_i^+ of the weight w_i.

$$w_i^+ = w_i - \eta * G$$

Equation 6

This process is iterated for each input sample multiple times until a minimum is approached for both weights and biases cost functions. As the number of parameters in ANNs is usually very large, the learning algorithm is able to estimate a local minimum, and rarely the function's global one. This entire process is called backpropagation, because the cost is calculated in the output layer, and then operates layer by layer in the backward direction, until reaching and adjusting the weights between the input and the first hidden layer.

Backpropagation was a big step towards the recent spreading of ANN in machine learning methods.

The particularity of neural networks compared to other machine learning methods is that this optimization has to be done on many variables instead of just a few. Let a low resolution greyscale digital image of size $28 * 28$ be an input for the basic neural network as shown in *Figure 5*. Let $H1$ be the number of neurons in the first hidden layer, $H2$ the number of neurons in the second, and O the number of neurons for the output layer. Each pixel's value x_i is an input, which gives a total of $I = 784$. The number of weights N that the network needs to optimize is $N = (I * H1) + (H1 * H2) + (H2 * O)$, giving a total of $6'376$. Added to this, the number of bias M is equivalent to $M = H1 + H2 + O$, making a total of 21 for this example. When adding more layers and working with higher resolution data, the task of parameters optimization can quickly become computationally heavy and that is the reason why dedicated frameworks have recently been developed for deep learning.

4.3 Convolutional neural networks

Particularly efficient for computer vision, CNN is a type of deep ANN that seek to imitate animal vision in the processes of pattern recognition and scene classification made possible with the visual cortex. Instead of reacting directly to light stimuli, CNN gives the possibility for a computer to interpret the numbers that constitute the pixel values of an image and map them to classification scores. They are a particular kind of architecture of ANN that use mainly convolution and max-pooling layers as hidden layers, and usually fully connected layers for the classification probabilities output. CNN perform well with images as they consider the spatial information and relative position of the pixels. When carrying out a convolution on a raw image input, the neurons from the hidden layer are not linked to every input pixel as for traditional ANN, but are connected locally. CNN can take inputs with 3 dimensions: a height and a width representing the size of an image, and a depth that represents the channels, typically R, G and B values.

4.3.1 Down-sampling

The first hidden layer of a CNN is usually a convolutional layer (CONV). A square window of predetermined dimension that contains weights is positioned on the input image and computes a feature map by moving to every possible position. This window is usually referred as a filter, or kernel. *Figure 6* is an illustration of the process. In this example, a feature map of size $7 * 6$ and depth 1 is produced by multiplication of the weights from the filter of size $3 * 3$ with the corresponding superposed input pixel values of an input image of size $8 * 7$ and depth of 3. At one location, this is done for all the channels. All the multiplication results are then summed up to give the first element of the feature map, which represent an activation of neuron. As mentioned earlier, this neuron is not connected to all the input neurons: only to $3 * 9$ in the example of *Figure 6*. The filter then moves one step to the right, overlapping with 18 out of 9 pixels from the first computation, and repeats the process until reaching the right boundary of the image. It moves then one step down and scans the next line and so on until the feature map is fully computed. All this process can in fact be explained as a succession of matrices multiplications. A padding can also be used to avoid the shrinking of the input size that is inherent to convolutions, by adding a virtual pixels margin of size 1 (for $3 * 3$ filters) with null values on each side of the input image. It also possible to vary the size of the strides made by the filter.

A single convolutional layer can create multiple feature maps by using multiple filters that are able to extract different features from the input data, with the condition that the filters are all of same size. Feature maps are key elements of CNN as they are able to detect primary elements like curves, edges, or colors that compose an image. By adjusting the weights inside the filters thanks to backpropagation, the feature maps can learn to be receptive to specific kind of elements and the network learns how to combine them into more complex structures in the next convolutional layers, in order to produce the desired classification outputs.

After computing the feature maps in a convolutional layer, a ReLU activation function is usually applied to integrate non-linearity to the model. As seen before, ReLU replaces all negative values of the feature maps by zeros, making the

training faster without affecting the global accuracy of the CNN. Keeping the content of *Figure 6* as an example, the output of a convolutional layer with ReLU activation function would be a map of size $7 * 6$ that contains neuron activations, with a depth equal to the number of filters used. This entire output can now become the input to another layer.

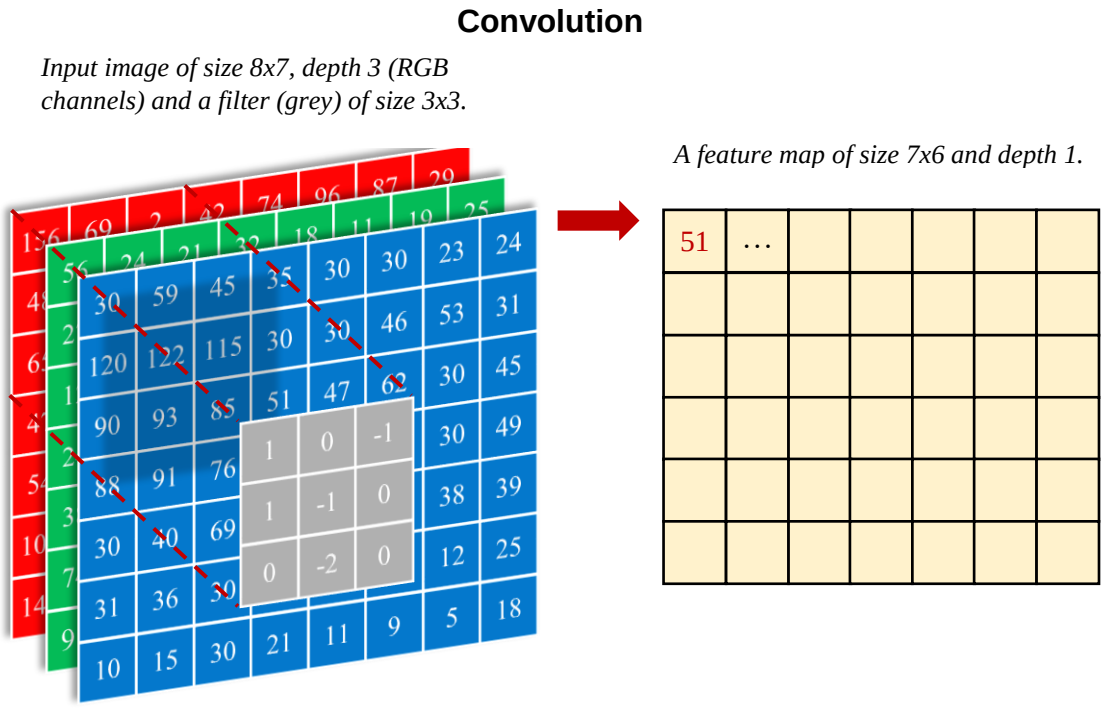


Figure 6

The pooling layer is the other kind of layer that is used in CNN. Most of the time, it is used after each convolutional layer of the network. It takes as input each feature map and produces a subsampling, as shown in *Figure 7* for the most common type of pooling, the max-pooling.

Max-pooling (POOL) is a subsampling operation that retains the maximum value within a window of predetermined size. This window moves on the entire feature map with no overlaps and creates a new output of smaller size. Using a $2 * 2$ window as in *Figure 7* reduces the number of values of the feature maps by a factor 4 which is an obvious way of lessening the computational heaviness of a network. This layer is also useful for a few other reasons. First, it helps to prevent an over-fitted model as it summarizes the information by decreasing the number of learnable parameters. Secondly, it makes the model insensible to spatial

variance of the features and objects, which can still be detected wherever they are located on the input image as high values will always be returned. Unlike convolutional and fully connected layers, pooling layers does not have any weights to adjust and simply perform a fixed subsampling operation.

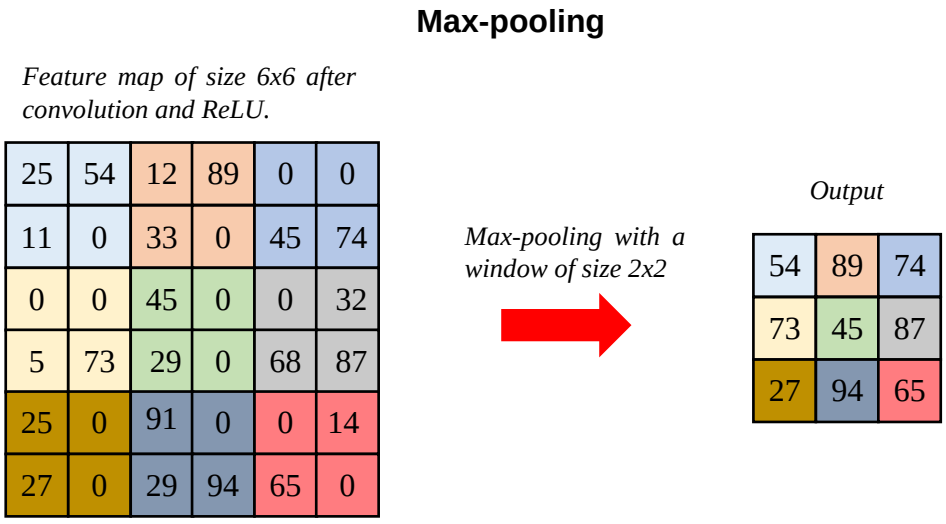


Figure 7

Fully connected layers (FC) are used at the end of the CNN to produce the output in the desired shape. They are characterized as fully connected as opposed to the convolutional and max-pooling layers that are not. Each neuron of this layer receives as input each activation from the preceding layer. In fact, they are hidden layers as described in *Figure 5*, and can also be stacked. In general CNN architectures end with fully connected layers a Softmax activation function that provides the final prediction probabilities for each class.

When a CNN is designed with many layers and becomes very deep, it becomes more prone to over-fitting as the number of parameters increases considerably. Adding dropout layers to the network is a way to reduce this phenomenon by randomly deactivating a predetermined percentage of neurons. It is also a good way to decrease significantly the duration of the training. Most of the time dropout is applied to the entire network but it is also possible to target specific layers.

All these layers and operations can be stacked and combined in many different ways, and each architecture can have its own advantages and disadvantages. In the end, the challenge is to find an optimum between depth of the neural network

and performance in terms of training time. *Figure 8* shows a typical architecture of CNN.

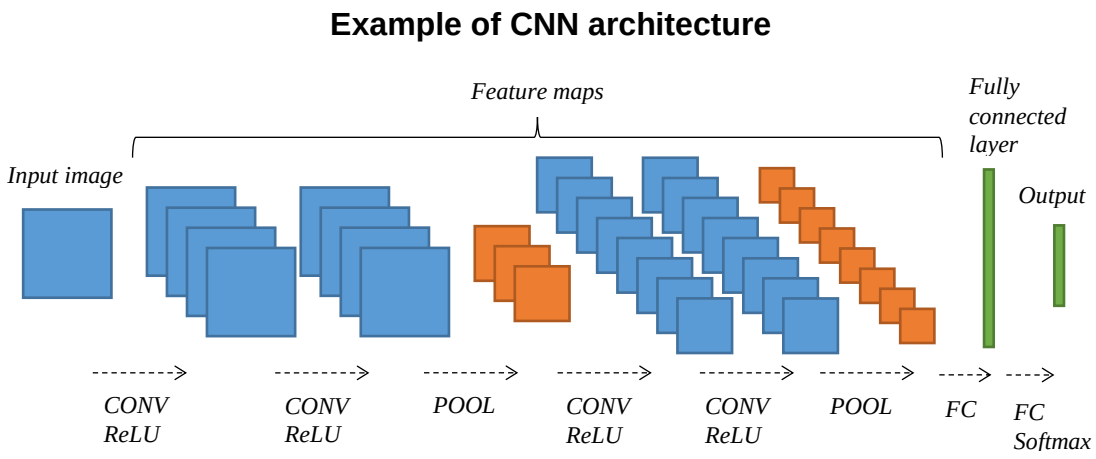


Figure 8

4.3.2 Up-sampling

Semantic segmentation requires more than just convolutional and sub-sampling layers, as the prediction is not done for an entire scene, but at pixel level. In fact, it needs layers that will do the exact opposite operations on the data after the convolution stage, until the model delivers an output image of the same size as the input, with each pixel being labelled with the class that has the highest probability. Up-sampling can be achieved with the usage of transposed convolution layers (Dumoulin & Visin, 2016). Transposed convolution works like a normal convolution, but zeros are added between all the values of the input as shown in *Figure 9*. The result is an output of larger size than the input. As for usual convolutional layers, the strides, padding and filter sizes can be adapted to produce various output shapes, and the weights of the filters are learnable. Mapping an input pixel to an output labelled pixel through a neural network might seem complicated to achieve when considering the amount of connections. As convolution, pooling and transposed convolution layers are made of neurons that are not fully, but locally connected to the pixels of the input image, an inherent connectivity pattern exists and makes this pixel-wise classification possible.

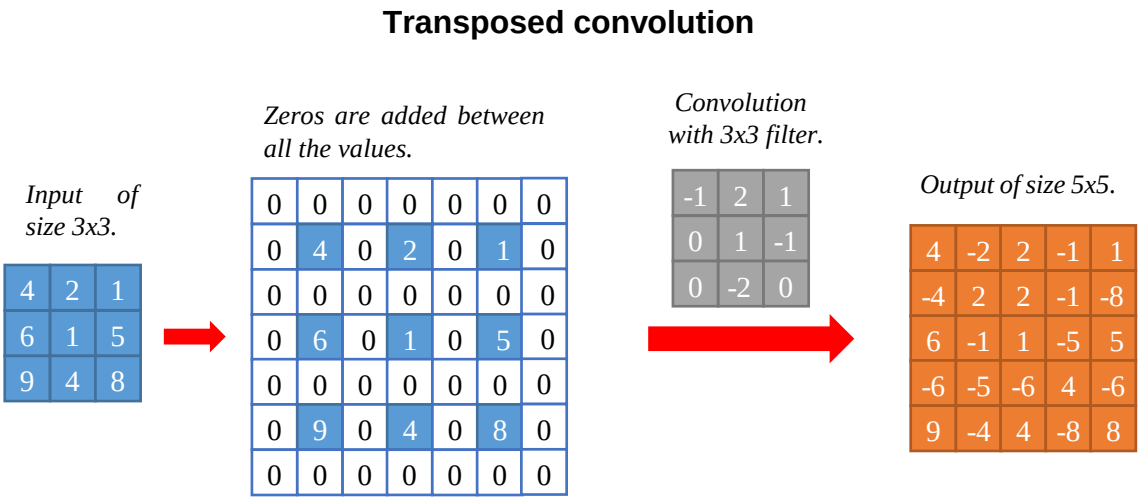


Figure 9

Transposed convolution is not the only up-sampling method for semantic segmentation with CNN. When applying max-pooling to feature maps issued from the convolution layers, it is possible to store the information that describes the position of the maximum value extracted from each max-pooling window (see *Figure 7*). This information is stored in the form of arrays of indices that can be used to up-sample the feature maps into sparse feature maps, that replaces the missing values with zeros. This up-sampling method performs particularly fast as there are no added weights to train, unlike the transposed convolution.

4.4 SegNet and U-Net Architectures

The two CNN models that have been selected for the experiments conducted in this paper are slightly modified versions of the SegNet and the U-Net. The following figures (*Figure 10, Figure 11*) are representations of the original architectures that have been suggested in the respective scientific articles.

4.4.1 U-Net

U-Net is made of 18 convolutional layers, 4 transposed convolution layers, 5 max-pooling that appear after each 2 convolutions in the encoder stage. In the decoder part, transposed convolutions up-sample the feature map and two convolutions are done after each of them. A Softmax classification is performed at the end. This model is heavy as there are many convolution layers, and each of them have weights and biases to learn. The actual model used for the

experiments in this project is a lighter version named Mobile U-Net. The main difference is that conventional convolutional layers are replaced by depth-wise convolutional layers. With this kind of layer, the convolution is done separately for each channel with filters of size $1 * 1$. These changes trade a part classification accuracy for more efficiency, but remains interesting for some tasks that needs to be done on low power devices.

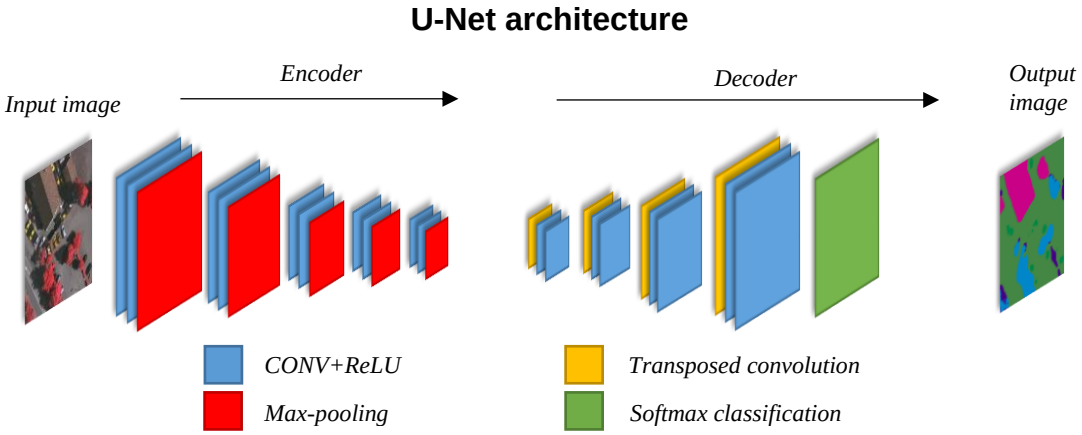


Figure 10

4.4.2 SegNet

SegNet is made of 20 convolutional layers, 4 max-pooling layers, and 4 up-sampling layers. The convolution stage begins with 2 convolutional layers with a max-pooling. This is repeated once, and the following is similar but with 3 convolutional layers. There is usually no transposed convolution in a SegNet architecture.

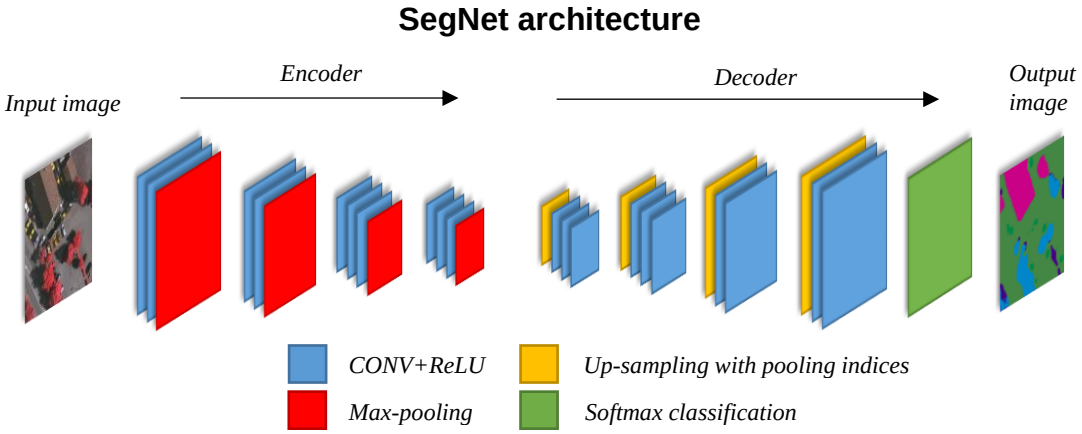


Figure 11

This model uses the pooling indices for up-sampling and does not have any fully connected layer after the convolutions. Softmax classification is made at the end of the network. This CNN has a symmetrical shape and has less parameters to train than the U-Net, as the up-sampling is done with the pooling indices method. Although its architecture remains similar, the SegNet inspired encoder-decoder used in the present experiments is using transposed convolutions for the up-sampling stage, as the framework used does not have an implemented function for extraction of pooling indices yet. Mobile U-Net and SegNet are similar in their architecture, but Mobile U-Net is supposed to output less accurate results as it was designed to be lighter for mobile applications.

4.5 Accuracy

The accuracy of a model is the percentage of correctly predicted samples reported to the total number of samples. In the context of CNN, the accuracy measures the percentage of pixels that have been labelled with the target semantic class as described in the associated ground truth image. Accuracy is usually computed at the end of the training with multiple test samples picked from the dataset and that have never been seen before. It can also be calculated regularly during the training on a validation dataset in order to have an indication on whether the model is performing well or not.

5 Deep learning frameworks

The generalization of deep learning techniques has largely been supported by the emergence and the development of dedicated libraries and frameworks that are now open source and easily accessible on internet. A large community is interested in these technologies, and experiments and progresses are made on a daily basis by both artificial intelligence professionals and amateurs. Most of the libraries are developed for usage with Python language and can be installed on most operating systems. The next few paragraphs will introduce the most popular frameworks[1] for deep learning.

The first major framework is the very popular TensorFlow[2]. TensorFlow is an open source deep learning framework that provides adapted environment and tools for designing and developing deep neural networks. It was developed by Google's AI research team "Google Brain" and the code was released in 2015 to the public. Python is considered as a slow language as it runs the code line by line. The particularity of ANN is that an optimization has to be done on millions of variables in a parallel way instead of just a few for traditional Machine Learning techniques. This can be computationally very heavy when executed exclusively with Python. TensorFlow was specifically designed to operate efficiently with such computation task on large arrays that are called "tensors" in this context. Thanks to the library's ability to process multiple variables in parallel, the computation's speed is increased significantly.

First the user defines the neural network's computational graph using TensorFlow's tools and functions. Then a session needs to be run in order to feed the input data to the graph, and train the neural network. The session runs in a C++ environment that operates much faster than Python, and then returns the results to the user. In other words, Tensorflow uses the readability and relative simplicity of Python language to define various parameters and the

[1] Popularity ranking of deep learning frameworks according to https://blog.thedataincubator.com/2017/10/ranking-popular-deep-learning-libraries-for-data-science/
[2] Tensorflow is available at https://www.tensorflow.org/

computational graph and runs it in a much faster C++ environment before returning the outputs back in the Python interface.

Besides the fact that it was developed by a company occupying center stage in the computer market, there are few reasons why Tensorflow became so popular. First it was one of the first to provide a more readable syntax for Deep Learning, making the development and testing of neural network architectures much easier for researchers. It is also very flexible as the user can easily switch from Central Processing Units (CPU) to Graphics Processing Unit(s) (GPU) or Tensor Processing Units (TPU) platforms to run a session, without any modification of the code. The software also provides visualization tools for the computational graph, and for various statistics related to the training, and assessment of the classification accuracy all along the process.

Theano[3] is considered as the first attempt to build a deep learning library using computational graphs in a python environment and running on either CPU or GPU. It was developed in an academic setting by a group of researchers at the Université de Montréal. While its development came to an end in September 2017, it remains a popular library as of today. It might however be gradually superseded by more recent frameworks.

Caffe[4] is another deep learning framework that was created by Yangqing Jia during his PhD from 2014 onwards. This library is quite similar to TensorFlow but is more oriented for computer vision problems. It also provides pre-trained weights which can allows faster training in some particular cases. Facebook developed and released in 2017 their own version of the library, Caffe2, which is more modular and lighter for usage on mobile devices essentially.

PyTorch[5] is a fully Python integrated library for deep learning. The key difference with other frameworks is that it not required to entirely define the computational graph before testing the code, and debugging can be done progressively. It is

[3] GitHub page at https://github.com/Theano/Theano/
[4] GitHub page at https://github.com/BVLC/caffe
[5] GitHub page at https://github.com/pytorch/pytorch

similar to NumPy library in the sense that it facilitates the use of arrays but is built to run efficiently on GPU.

On top of these leading deep learning libraries, projects of high-level libraries recently emerged. Keras[6] for instance was developed with the objective of allowing fast experiments with only a few lines of code. This library can run on TensorFlow, Theano, or other fundamental frameworks backend. TFLearn[7] is another example that only runs with Tensorflow. This kind of easy-to-use libraries probably participated to the growing of deep learning techniques in research communities or for use by individuals.

In order to work with these libraries and train deep neural networks in a reasonable time window, it necessary to have access to at least one high performance and recent GPU. Most of the time, these frameworks can run on either CPU(s) or GPU(s), though running on GPU will always be much faster.

There are two possibilities for training an ANN. The first one is to own or build a local installation with adapted computer components. Investment costs can be important for an acceptable computing power and the installation of Deep Learning libraries can take a bit of time, and also expertise as there can be many dependencies and version issues that need to be solved during the process, especially on Windows. The alternative is to use cloud platforms that provides powerful CPU's and GPU's. These online platforms are designed to be user-friendly and time efficient but can become costly depending on the nature of the task. Typically, the user buys computing time, writes the algorithm as if he would run it on a local machine, and uploads the code and the dataset on the platform. The training can be launched and everything happens in the cloud. The output is in the form of files that can be downloaded afterwards. This method might remain cheaper than building a local installation in most situation. In the end, it is essential to identify the needs of a project in order to optimize costs and time.

[6] GitHub page at https://github.com/keras-team/keras
[7] GitHub page at https://github.com/tflearn/tflearn

The *Semantic Segmentation Suite*[8] has been used in the context of the present project. It consists of an open source project that allows fast switch between the main CNN models that appear in the literature and that are ready-to-use. As of today, a total of 12 different models are available for training and testing, and more are about to be implemented. The "*Encoder-decoder based on SegNet*" and the "*Mobile UNet for Semantic Segmentation*" are the two models that has been used for the present experiments. The *Semantic Segmentation Suite* also provides tools that simplify the record of accuracy during the training, as well as a loss and accuracy charts afterwards. The user needs to preprocess his datasets in a particular way which is described in the methodology chapter. The models run on a local machine with TensorFlow on a NVIDIA GTX 1080 and with 16GB RAM.

[8] GitHub page at https://github.com/GeorgeSeif/Semantic-Segmentation-Suite

6 Methodology

This chapter presents the successive steps that led to the semantic segmentation of two remote sensing datasets, and to the comparison of the results obtained with the U-Net and SegNet inspired models. An explanation on why these datasets have been chosen and how they have been built, as well as a description and a statistical exploration will be provided. In order to preprocess the data before feeding them as in input to a CNN, it is important to understand the various requirements that such method can have. The needs and limits of the algorithm will therefore be identified, and the preprocessing steps will be described.

There is a recurrent process that can be highlighted in the literature about ANN in practice. After preprocessing the dataset, this one is divided into three subsets. The first subset and the most important one contains the training data, which the CNN will learn from. The second subset is for validation and is used for assessing the accuracy and the error at defined intervals during the training. The last subset is useful after the training, for testing the model with data that has never been processed by the CNN. It allows to perform the final evaluation of accuracy and cost of the model.

6.1 Data

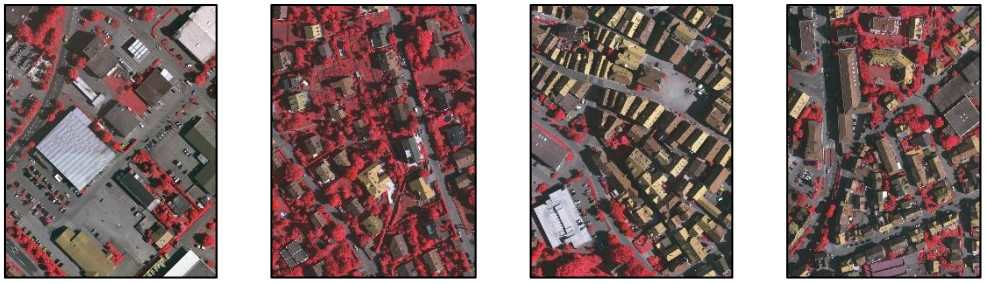
Data acquisition is the very first stage of the analysis and very important as the entire process depends on the reliability of the datasets. The Vaihingen dataset and the homemade OFS – Mos25 data have been selected for this project. It is essential to train the neural network with reliable, abundant and representative labelled data. Unfortunately, this kind of dataset can be costly, especially for high resolution satellite imagery. The selection of the first data for this experiment, the Vaihingen dataset, is based on its open source availability and its generalized use in the research community for semantic segmentation. The second dataset was built with the help of files that are not free of costs but were still used for this project as they are available for students in the GIS servers of the University of Lausanne for research purpose.

6.1.1 Vaihingen datasets

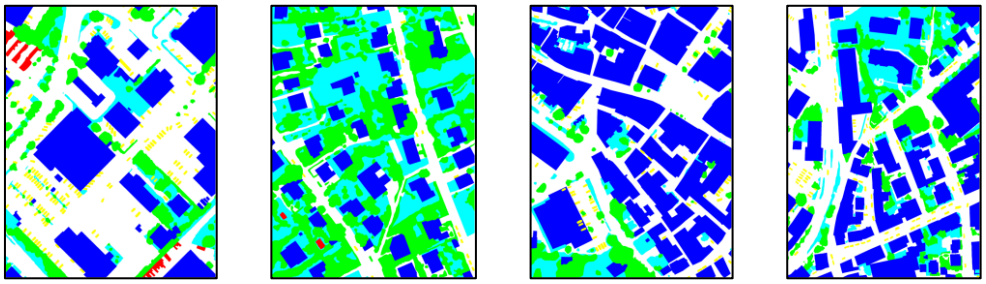
The first dataset is the Vaihingen dataset. It was initially created and made available to a few research groups from the International Society for Photogrammetry and Remote Sensing (ISPRS) for a semantic segmentation challenge organized in 2014: the *"ISPRS Semantic Labeling Contest"* of the ISPRS Working Group II/4[9]. It was created for researchers to experiment semantic segmentation on a unique dataset at an international scale, thus allowing a meaningful comparison between the classification accuracies obtained by various architectures of convolutional neural networks designed by different groups. Since summer 2017, the dataset is available open source on internet[10], which is quite convenient in the context of the present experiment.

Vaihingen dataset samples

Aerial images



Ground truth images



For all 16 ground truth images:			
Class	**Color**	**Number of pixels**	**%**
Clutter / Background		526 083	0,67
Cars		945 687	1,21
Low Vegetation		16 272 917	20,84
Tree		18 110 438	23,19
Buildings		20 417 332	26,15
Impervious surface		21 815 349	27,94
Total		**78 087 806**	**100,00**

Figure 12

[9] ISPRS Working Group II/4 web page : http://www2.isprs.org/commissions/comm2/wg4.html

[10] *The Vaihingen dataset set was provided by the German Society for Photogrammetry, Remote Sensing and Geoinformation (DGPF) [Cramer, 2010]: http://www.ifp.uni-stuttgart.de/dgpf/DKEP-Allg.html.*

Vaihingen is a small city in the south-west of Germany, located northwest of Stuttgart. The population is estimated at $29'041$ in 2016 with a density of $395.7/km^2$, according to the Baden-Württemberg Statistics[11]. The territory is mainly urban, with usual variations in the density when going from central to sub-urban areas.

The dataset consists of 16 colour-infrared (CIR) aerial images of various sizes going from $2'336 * 1'281$ to $2'818 * 2'558$ pixels. The resolution is of 9 cm pixels per pixel. Corresponding ground truth images are provided and divides the territory into 6 distinctive semantic classes: *"Buildings", "Impervious Surfaces", "cars", "low vegetation", "Tree",* and *"Clutter/Background"*. Four examples showing various characteristics of the territory are given in *Figure 12*. As this data has been captured at a high resolution, shapes are generally homogeneous and simple, with clear boundaries.

The "*Clutter / Background*" class gathers all the elements that are usually not interesting in the urban context for semantic segmentation. This class is not well represented in the dataset and may obtain poor classification accuracies. *Figure 12* also shows the proportions of pixels for each class in the whole set of ground truth images. It is clear that some classes like "*Buildings*" and "*Impervious surfaces*" (mostly roads), "*Trees*" and "*Low vegetation*" are largely more represented than others as they totalize 98.12% of the pixels. This will have an impact on the training of the network in terms of accuracy.

6.1.1.1 Pre-processing

The data needs to be pre-processed in order to meet the input prerequisites of the CNN algorithm as suggested in the *Semantic Segmentation Suite*. In the first instance, the raw aerial images and their associated ground truth from the Vaihingen dataset had to be cropped to smaller patches of equal size. In order to test the responsiveness of the neural network to the size and scale of the input data, two different cropping sizes have been retained and both will be used to

[11] Sources: Statistisches Landesamt Baden-Württemberg, Vaihingen, 2016, https://www.statistik-bw.de/BevoelkGebiet/Bevoelkerung/99025010.tab?R=GS118073

produce distinct datasets for separate training sessions. As the patches need to be square shaped and all of the same size, some pixels in the right and bottom boundaries of the raw images have been omitted. *Figure 13* shows the number of patches as well as the aspects that they have when cropped with the selected sizes.

Vaihingen patch samples

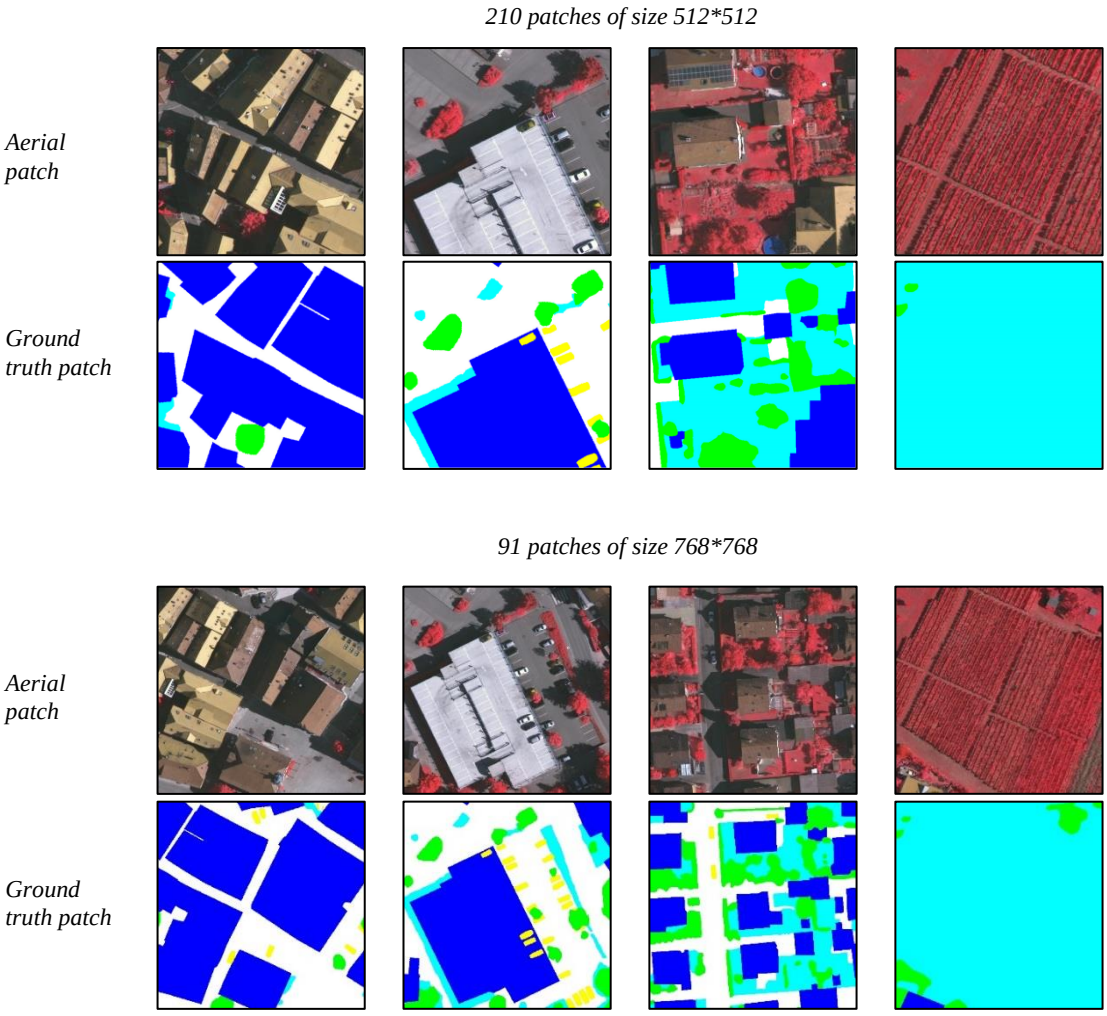


Figure 13

Compared to other major and popular sources of data like MNIST[12] or CamVid[13], the Vaihingen dataset can be considered rather small and it can be expected that the overall accuracy will not be able to reach as good levels. For training, validation and testing, these patches have to be subdivided into three sets. This

[12] Available at http://yann.lecun.com/exdb/mnist/
[13] Available at http://mi.eng.cam.ac.uk/research/projects/VideoRec/CamVid/

is usually done in a random manner, but making sure that all ground truth classes are represented in each subset. The proportions that each subset should have can be discussed, but it is generally considered that small datasets should allow at least 80% for training, then sharing the 20% left for validation and for testing. These are the retained ratios in the context of this experiment.

6.1.2 OFS - Mos25 datasets

The second dataset has been built with data from two different sources, specifically for this project. With a view to identify the sensibility of the CNN to different scales, this dataset covers their entire territory of Switzerland at a resolution of 25 meters per pixel. The preparation of this dataset involves two different files. The first one is a satellite RGB "*Landsat Mosaik*" image *(Figure 14)* of Switzerland of size $17'500 * 12'000$, with a resolution 25 meters per pixel, captured by the American satellite Landsat 5 between 1990 and 1994[14].

Mos25 file

Figure 14

The second one is a raw CSV table (*Figure 15,* left) that specifies for a given pair of coordinates different precision levels of semantic class, representing the most frequent land cover of the surrounding hectare. This data has been captured by the Federal office of statistics of Switzerland (OFS) for the year 2004[15], which is not the same year as the first file. It is however assumed for this project that land cover of the Swiss territory has not significantly evolved during that period. Three levels of precision are proposed in the table. The most abstracted level contains

[14] Satellite Image: © ESA / Eurimage / Swisstopo, NPOC
[15] Statistiques de la superficie : © 2004 Office fédéral de la statistique, Neuchâtel

three classes. The second level subdivides these three into 10, and the last one subdivides the 10 into 27. The second level with 10 classes is the one that has been selected for this project, and contains the following labels: *"Buildings", "Transports", "Special infrastructures", "Green spaces"* (in urban environment), *"Arboriculture"*, *"Fields", "Pastures", "Forest", "Lakes & rivers", and "Unproductive surface".*

OFS file

X	Y	LU18_10
485500	109700	300
485500	109800	300
485500	109900	300
485500	110000	300
485500	110100	300
485600	109800	220
485600	109900	300
485600	110000	300
485600	110100	300
485600	110200	400
485700	109800	220
485700	109900	220
.	.	.
.	.	.
.	.	.
549900	157900	220

For the entire ground truth image:			
Class	**Color**	**Number of pixels**	**%**
Special infrastructures		272,480	0.41
Green spaces		314,304	0.48
Arboriculture		817,184	1.24
Transports		1,523,832	2.31
Buildings		2,815,752	4.26
Lakes & rivers		2,987,392	4.52
Pastures		9,008,182	13.64
Unproductive surface		14,256,834	21.58
Fields		15,211,200	23.03
Forests		18,852,792	28.54
Total		**66,059,952**	**100.00**

Figure 15

In order to have labelled pixels that suits as a ground truth image, a raster map (*Figure 15,* right) has been created from the CSV file thanks to an open source GIS software[16], then calibrated to the resolution of the satellite image. This means that one ground truth pixel is subdivided into 16 labelled pixels that are associated to 16 pixels of the satellite image. As the ground truth image was initially at a lower resolution than the satellite one, it is expected that the precision of the learning might be affected, as wrong labels can be assigned to some pixels of the satellite image. This dataset is however a way to assess the performances of CNN on remote sensing imagery at different scale

[16] QGIS, open source GIS software available at https://www.qgis.org/en/site/

Switzerland's territory can be segregated into 3 main parts. The larger one covers the Alps in the south and southeast of the country. The second one is a band going from southwest to northeast, called the Swiss "Plateau". This part is where most of the urban settlements are located, as it is the flat part of Switzerland. The remaining part is the Jura, slightly mountainous too, in the northwest. These three regions can easily be identified when looking at the map in *Figure 15*. The pixel ratio for each class is also presented in *Figure 15*. Amongst the 10 classes, 4 are more represented and totalize 86.78% of the pixels. This lack of balance needs to be considered later when assessing the results of the semantic classification.

6.1.2.1 Pre-processing

The pre-processing steps applied to this data are the same as the one applied to Vaihingen dataset: cropping the satellite and ground truth images into patches of two different sizes in order to obtain two distinct datasets (*Figure 16*).

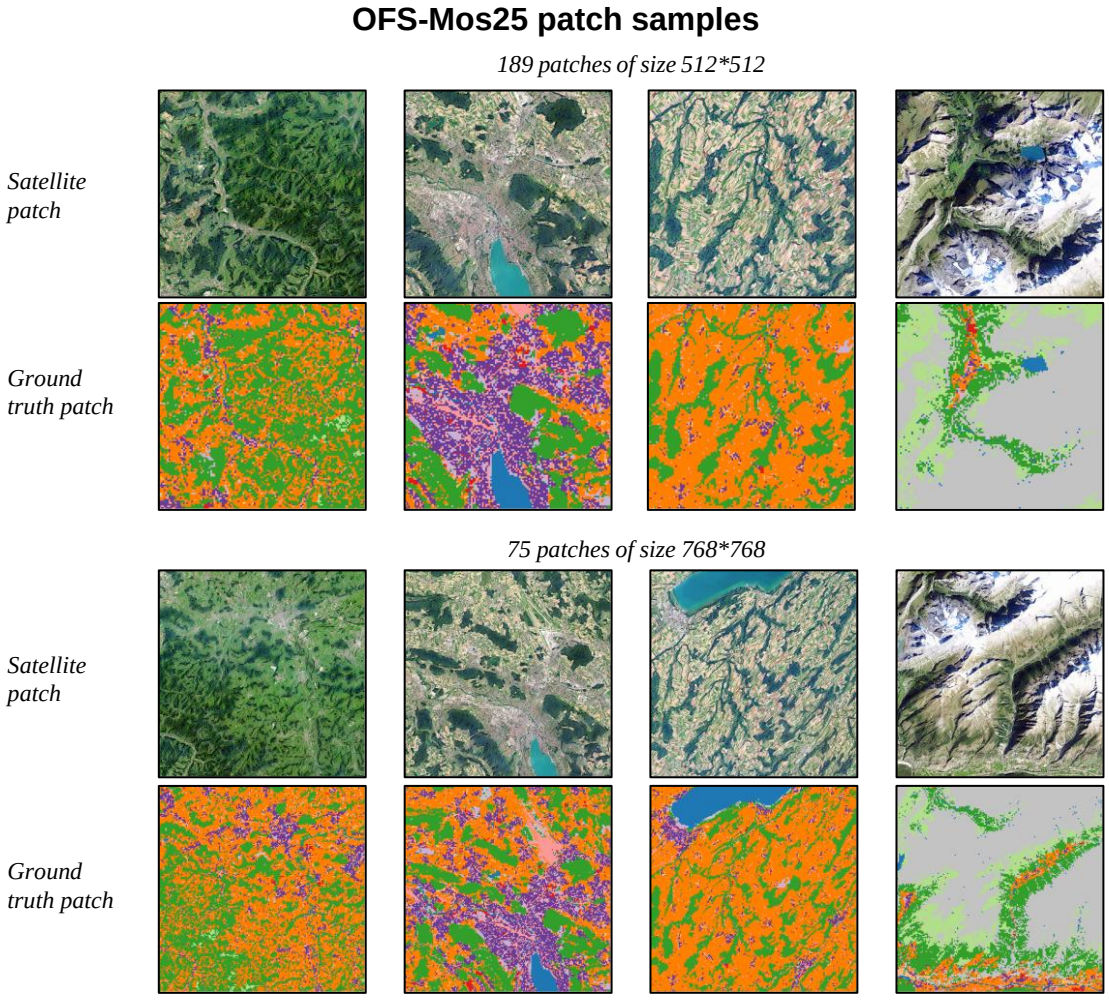


Figure 16

As the resolution is here very low, the ground truth patches have pixelated patterns, with very isolated labels here and there and a generally scattered distribution for most classes. Because the ground truth is here available only for the territory of Switzerland, square patches had to be cropped inside in a manner that no pixels are located outside the boundaries of the country. This means that most pixels that are close to the limits are not part of the dataset in its final form. As well as for the first dataset, 80% of the patches are randomly selected for the training and the remaining for validation and testing.

6.2 Training the models

The Semantic Segmentation Suite allows the training of different CNN models with the exact same shaped datasets. Two distinct CNN architectures have been selected for comparison. *Table 1* lists all the trainings that have been made with the Vaihingen and OFS-Mos25 datasets. In the same table, an ID is associated to each training, and will be used when a particular training needs to be referred in the accuracy results table. As already mentioned the experiment has been conducted with two different patch sizes. As both datasets can be considered rather small compared to others that are not related to remote sensing, all the models have been train once with image augmentation and once without. Image augmentation is a technique that allows to create more data from the original data. For example, one can take all the digital images from a dataset and rotate them in various angles, zoom in a region, operate a horizontal or vertical symmetry, change the lighting or contrasts, and also use combination of all these to produce more data for the network to learn from. The new information generated is not redundant as these transformations can be associated to real life conditions when observing an object, and this works for remote sensing similarly. All the possible combinations give a total of 16 distinct trainings that will be compared.

Some hyperparameters have to be set before launching any training. First, the batch size defines the number of training example that are fed at the same time to the network before it performs an adjustment of the weights. Usually, the bigger the batch, the faster the network, but the more memory is required. A

batch of size 1 has been selected here as the number of training example is rather small, and the size of each example is rather big. The other hyperparameter is the number of epochs to be done. An epoch means that the entire set of training examples have gone through the network once. Here, 300 epochs will be performed for a training. For the SegNet architecture, a dropout of 0.5 is set during the encoder stage. There is no dropout in the decoder part, nor in the U-Net model. SegNet model has $34'968'134$ parameters to adjust while Mobile U-Net has $8'872'262$.

List of trainings

Vaihingen dataset			**Training ID**
512*512 patches	Mobile U-Net	Non-augmented	VAI_512_UNet_NA
		Augmented	VAI_512_UNet_A
	SegNet	Non-augmented	VAI_512_Seg_NA
		Augmented	VAI_512_Seg_A
768*768 patches	Mobile U-Net	Non-augmented	VAI_768_UNet_NA
		Augmented	VAI_768_UNet_A
	SegNet	Non-augmented	VAI_768_Seg_NA
		Augmented	VAI_768_Seg_A
OFS - Mos25 dataset			
512*512 patches	Mobile U-Net	Non-augmented	SUI_512_UNet_NA
		Augmented	SUI_512_UNet_A
	SegNet	Non-augmented	SUI_512_Seg_NA
		Augmented	SUI_512_Seg_A
768*768 patches	Mobile U-Net	Non-augmented	SUI_768_UNet_NA
		Augmented	SUI_768_UNet_A
	SegNet	Non-augmented	SUI_768_Seg_NA
		Augmented	SUI_768_Seg_A

Table 1

During the training, a validation is performed after each epoch to evaluate the overall accuracy and error function. When completed, a prediction is done to evaluate the accuracy of the semantic classification on the 10% data saved for the test. This allows to see if the model is not over fitted and able to generalize

well with content that has never been seen. A graph showing the evolution of the overall accuracy and another one showing the total cost is then generated for each epoch. In the results chapter, a comparison table helps to visualize the differences in the training accuracies and the semantic segmentation outputs.

7 Results

Results of the 16 trainings (*Table 1*) are presented in the form of 4 figures, each corresponding to one of the datasets. Each figure shows the predictions for 3 samples with different territory characteristics. For one sample, original image input, ground truth image, and prediction results for both models are shown, once with data augmentation, once without. For better readability and visualization, accuracies for each trainings are presented in a separate table (*Table 2, Table 3*). Training took between 8 and 8.30 hours for U-Net architecture and between 5.30 and 6.30 for SegNet. All the accuracy and loss graphics related to validation during the trainings can be consulted in *Annex 1*. Although U-Net takes longer for the training process, it is much faster when making predictions on the test data, with about 1 second of calculation per image. SegNet requires a little more, with about 1.2 second per prediction.

7.1 Visual assessment

As the evaluated accuracy of a model does not consider the visual aspects, it is first necessary to make a visual assessment of the predicted outputs in order to highlight the pros and cons of each models for a given dataset.

7.1.1 Vaihingen – $512 * 512$ patches

Figure 17 shows the results for the Vaihingen $512 * 512$ patches. The first test sample is a low residential area with low density. For both Mobile U-Net and SegNet models, outputs seem better when image augmentation as less “rings” appear in the other segmented regions. “*Buildings*” and “*Trees*” classes seem however to be well identified by the 4 models. The SegNet with augmented dataset output gives the most interesting result visually, with homogeneous regions and smooth segmentations between these. The few mistakes that can be identified are actually understandable in the sense that a human eye could do a similar classification. For instance, one of the alleys that goes from the road to one of the houses has the ground truth label “*Low vegetation*” which seems to be an over-simplification that the CNN does not make. The Mobile U-Net without image augmentation gives the worse result, with many rings of road class appearing within buildings or vegetation classes.

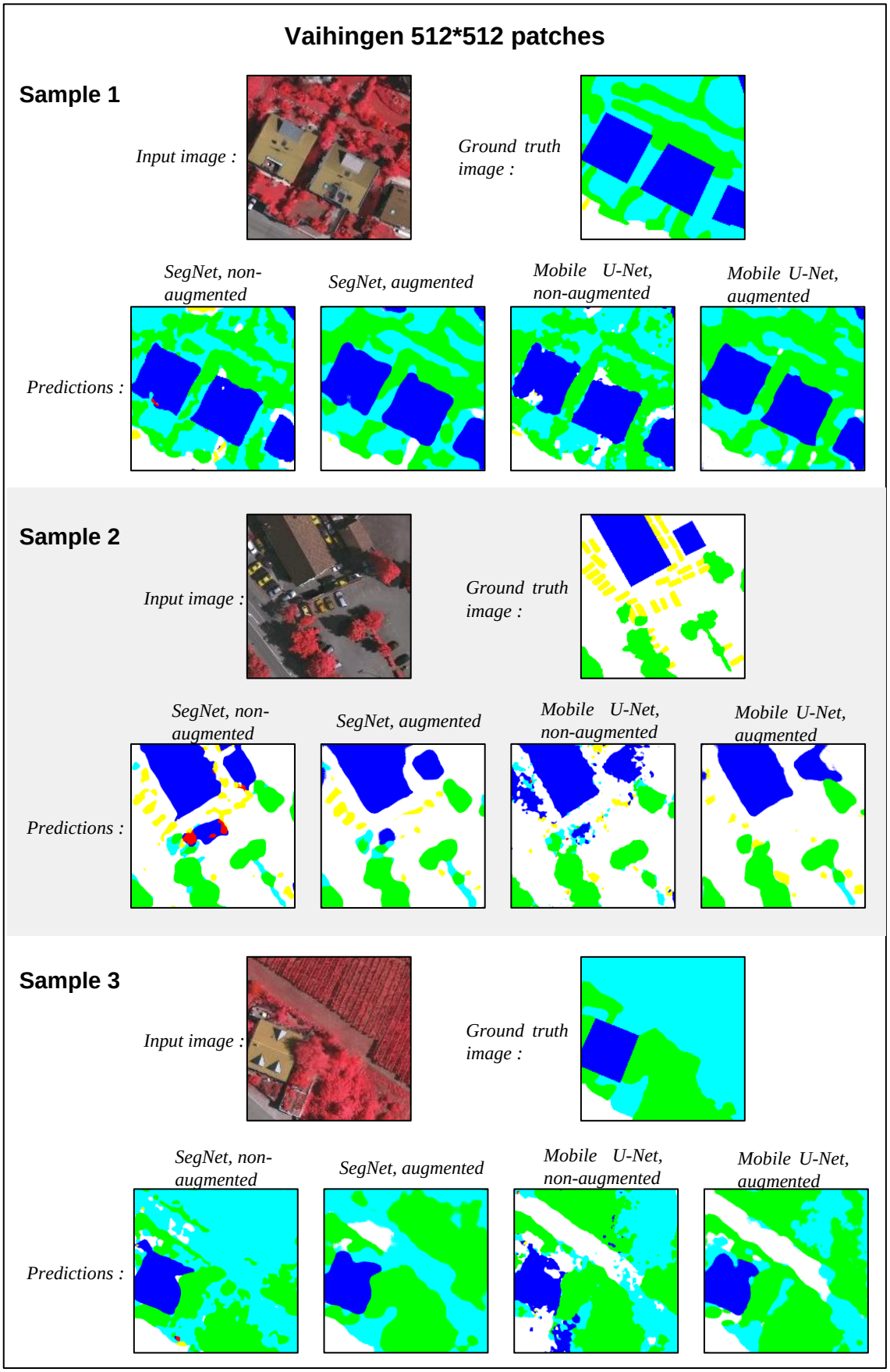


*Figure 17 – Results for Vaihingen 512*512 patches.*

The second sample image shows more road surface and cars. Again, the results seem smoother when image augmentation is used. SegNet seem to identify cars more easily, even if no model is able to detect all of them. It can be noted that all the models easily deal with the trees' shadow that are particularly present in this sample.

The third sample is more rural, with a high presence of the "*Low vegetation*" class. The models have a hard time to differentiate the "*Low vegetation*" and "*Trees*" classes, especially for the vineyards looking cultures that are located on the top right of the input image. Mobile U-Net largely mixes-up the "*Impervious surface*" class with the "*Low vegetation*". This time, it is the SegNet with non-augmented data that seems to give the best output.

7.1.2 Vaihingen – $768 * 768$ patches

Prediction samples for the Vaihingen patches of size $768 * 768$ are presented in *Figure 18*. The first sample shows a dense urban environment, with multiple buildings that are surrounded by vegetation, and orthogonally arranged roads. A few cars are also present. All the models provide a visually good output. Non-usage of image augmentation causes more scattered segmentations. It seems that the models are not able to reproduce sharp and thin shapes that appear in the ground truth images. The roof in the middle of the image has a similar texture to the road that is adjacent and is therefore never well separated from it. The SegNet with augmented data is the most precise in terms of car recognition.

The second sample has been selected to show how badly all the models perform with the "*Clutter / background*" class. None of the trainings were able to determine correctly its presence. This class is combination of semantically different objects (rivers, sport turfs, containers…) and it looks like the CNN do not know what to do with these pixels. This can obviously be attributed the very low number of pixels that represent this class in the dataset, and image augmentation techniques does not seem to be of any help in this situation. Only low vegetation and tree classes respectively seems to perform well.

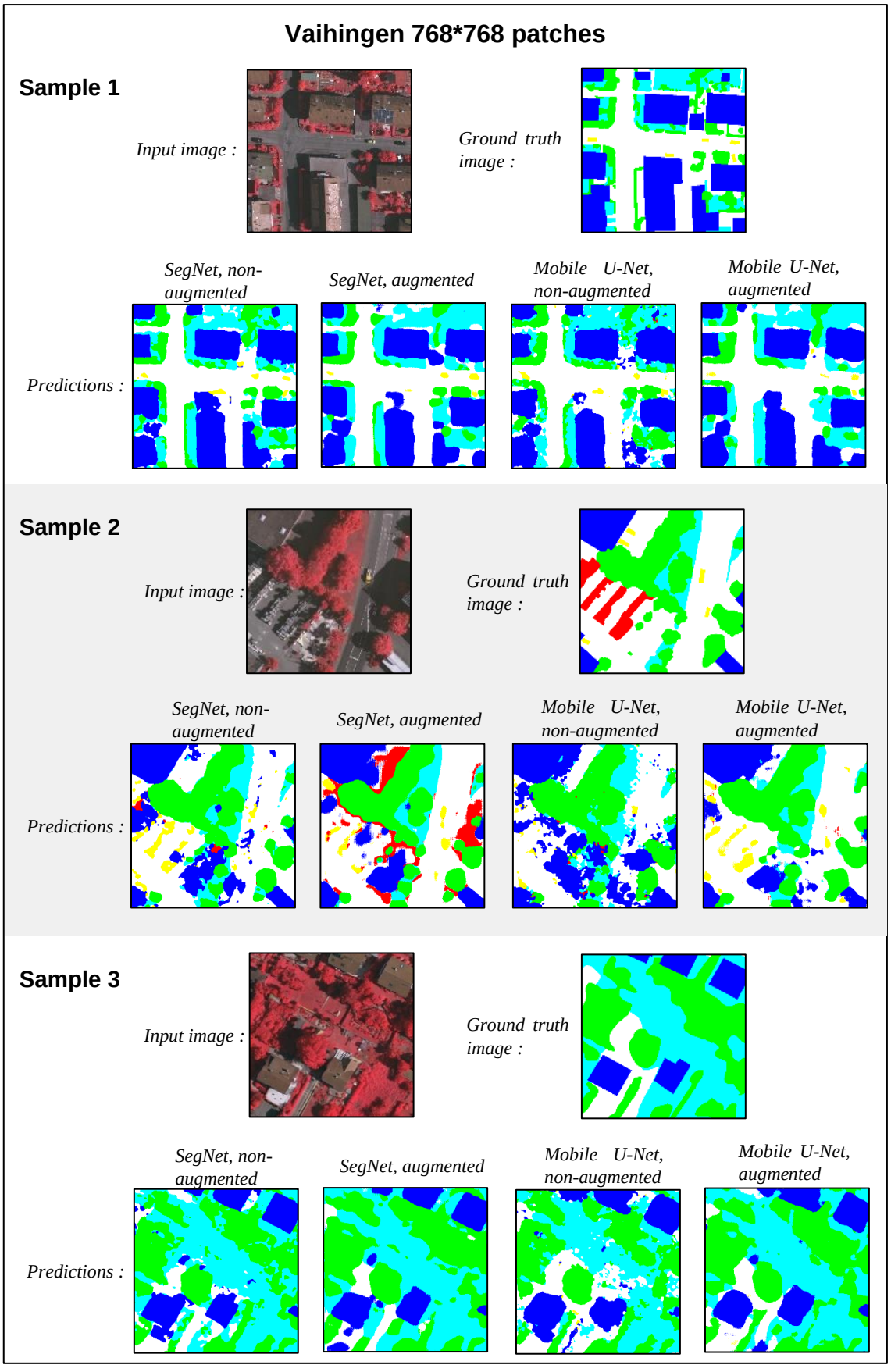


*Figure 18 – Results for Vaihingen 768*768 patches.*

The third sample shows a low density housing area. Besides the usual smooth and homogeneous segmentation results that provides image augmentation for both models, all the models achieve rather good predictions. Some spots are always misclassified with the "*Buildings*" label, but these pixels actually turn out to have similar texture and color. For this example, Mobile U-Net with image augmentation gives the best looking output.

7.1.3 OFS-Mos25 – $512 * 512$ patches

Test outputs for the patches of size $512 * 512$ made from the OFS-Mos25 dataset are presented in *Figure 19.* The first noticeable phenomenon on every samples is that image augmentation causes a smoothing of the segmentation, and many classes like *"Transports"*, *"Special infrastructures"* and *"Green spaces"* that should appear are simply wiped out by other more frequent classes like the *"fields"* and *"forests"*.

The first sample is the urban and periurban region of Arbon, along the Constance Lake, located in the northeast of the country. The predictions of the non-augmented SegNet and Mobile U-Net architectures look close to the ground truth image, as more isolated labels like the *"Arboriculture"* are predicted amongst more common ones. All the models make the same mistake in a particular sport of the lake, where lighter green shades in the water take over the dark blue and are labelled as *"Forests".* Visually, the SegNet architecture without augmentation of the training data seems to give the best output.

The second sample is located in the surrounding of the small city of Moutier, in the Jura region. Mobile U-Net without image augmentation tends to misclassify *"pastures"* and *"forests"* with the *"fields"* label. Urban settlement is also well identified, as for the SegNet one. Non-augmented SegNet seems again to provide the best looking output.

The last sample patch shows a mountainous region that is located near the village of Andermatt, in the Alps. Besides the over simplified results caused by the image augmentation, all the models successfully identify the gross regions

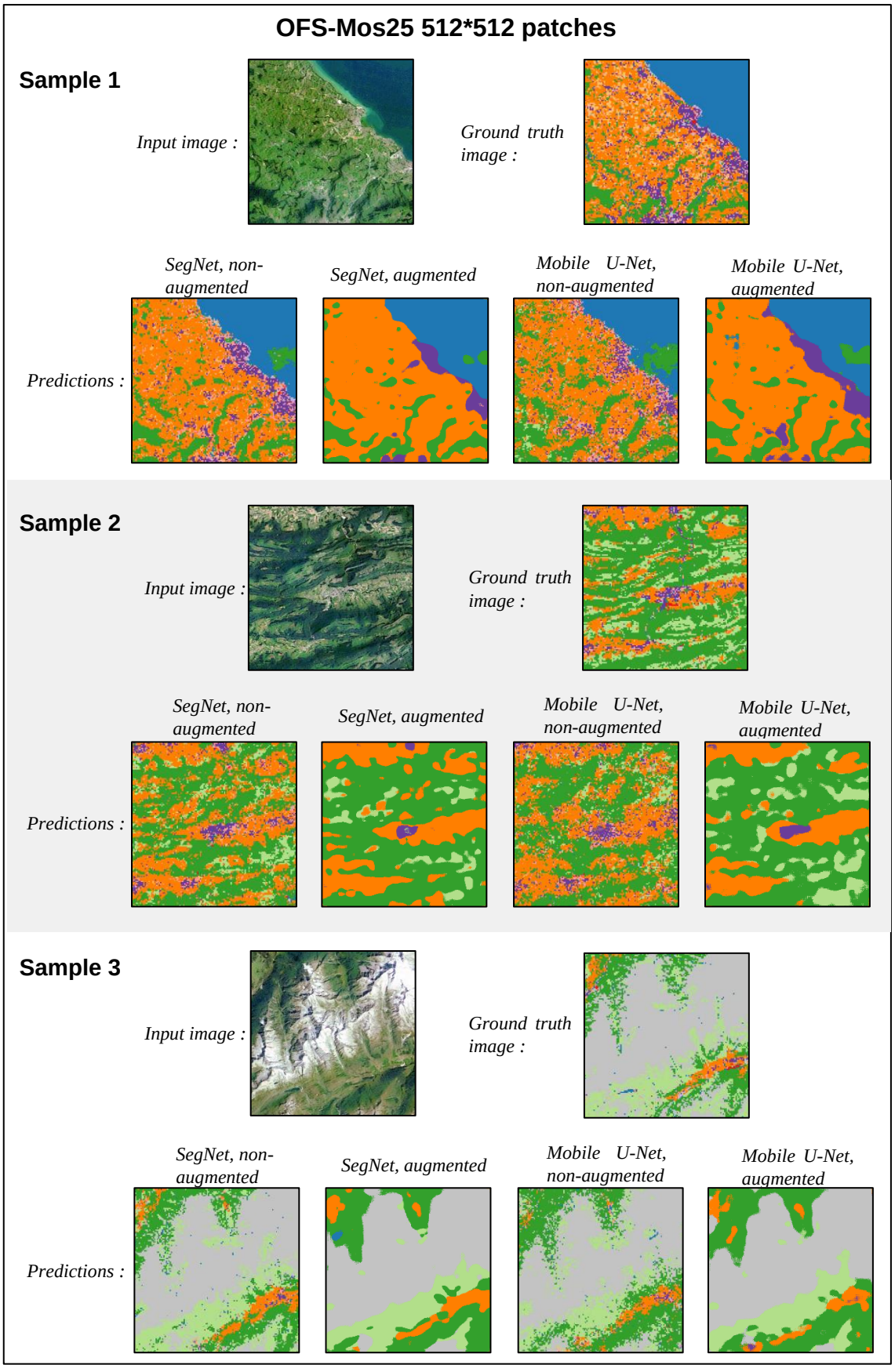

OFS-Mos25 512*512 patches
Sample 1
Input image :
Ground truth image :
Predictions :
SegNet, non-augmented
SegNet, augmented
Mobile U-Net, non-augmented
Mobile U-Net, augmented
Sample 2
Input image :
Ground truth image :
Predictions :
SegNet, non-augmented
SegNet, augmented
Mobile U-Net, non-augmented
Mobile U-Net, augmented
Sample 3
Input image :
Ground truth image :
Predictions :
SegNet, non-augmented
SegNet, augmented
Mobile U-Net, non-augmented
Mobile U-Net, augmented

Figure 19

that have to be segmented. With slightly more precision, SegNet without image augmentation provides the best predictions.

7.1.4 OFS-Mos25 – $768 * 768$ patches

This last result figure (*Figure 20*) is dedicated to the prediction outputs obtained after training the CNNs with the $768 * 768$ OFS-Mos25 patches. As per the previous predictions for the $512 * 512$ patches, augmented dataset causes homogeneous segmentations and labels with low frequency are completely ignored by the CNNs.

The first sample shows an urban and rural environment in the south of Zurich. A bit of Zurich Lake appears as well as Zoug's Lake on the bottom part. The results are globally good, with a higher precision in the shapes achieved by the SegNet model. The lakes are well delimited too. Both architectures however seem to have a hard time recognizing small and thin features like the roads and rivers that get included in other classes that are surrounds them.

The second test sample gathers urban, rural, and mountainous characteristics, which are located near Einsiedeln. The four predictions give very distinctive results and both models make their own classification mistakes. Sihl Lake, on the top of the patch, is only detected by the SegNet model with non-augmented training and the Mobile U-Net with augmented data. Mobile U-Net with no augmentation performs very bad, labelling the majority of the patch with *"Unproductive surfaces"*. Non-Augmented SegNet output looks slightly better but misclassifies a rather big spot in a mountainous region as "*Buildings*". This patch gave a hard time to each model and none can be eligible as a better option than the others.

The last sample evaluated in this visual assessment is a more mountainous region in the Alps. Predictions are globally looking good, with the usual slightly better precision for the SegNet in the identification of buildings and other less frequent classes.

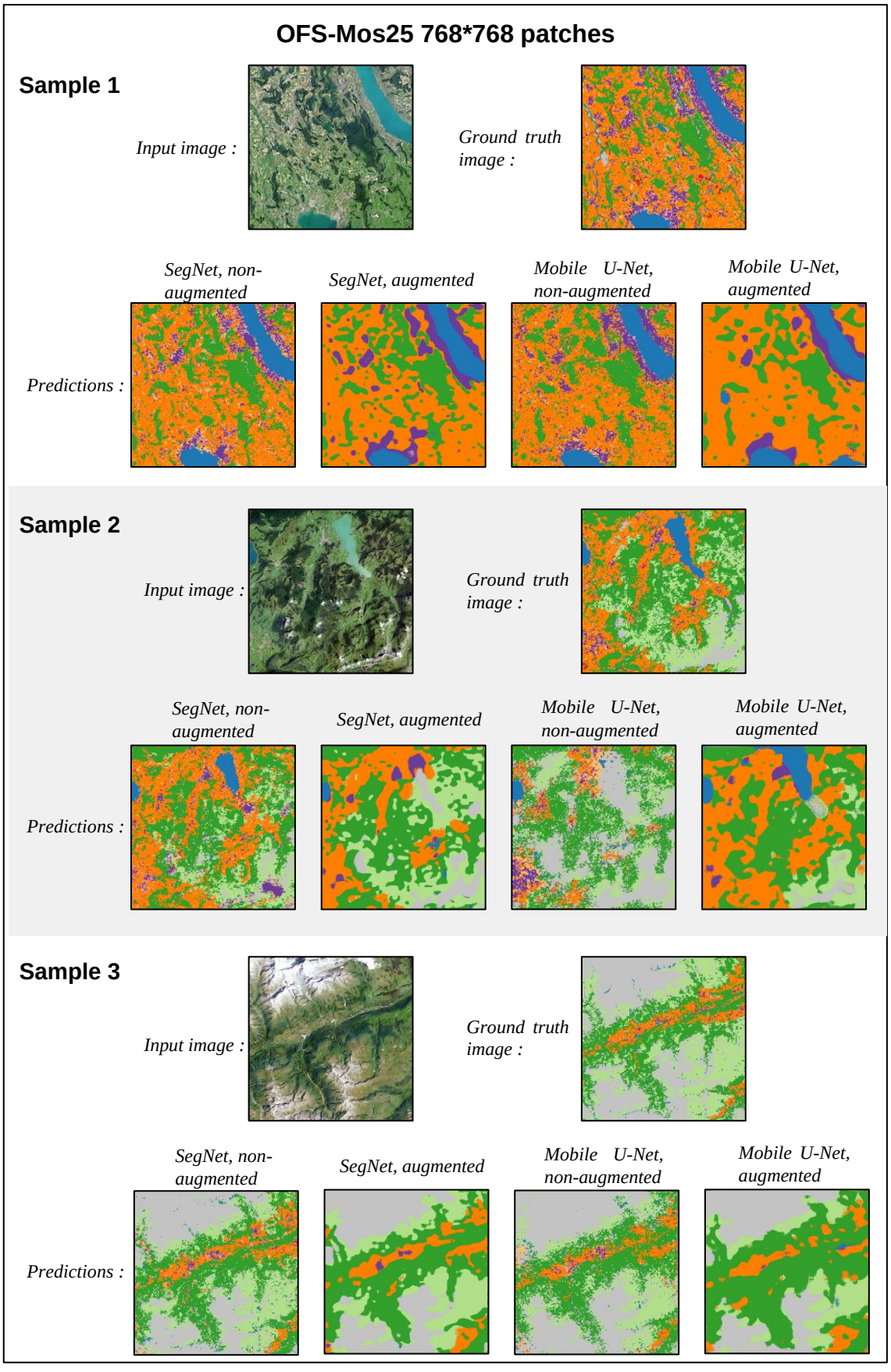

OFS-Mos25 768*768 patches
Sample 1
Input image :
Ground truth image :
SegNet, non-augmented
SegNet, augmented
Mobile U-Net, non-augmented
Mobile U-Net, augmented
Predictions :
Sample 2
Input image :
Ground truth image :
SegNet, non-augmented
SegNet, augmented
Mobile U-Net, non-augmented
Mobile U-Net, augmented
Predictions :
Sample 3
Input image :
Ground truth image :
SegNet, non-augmented
SegNet, augmented
Mobile U-Net, non-augmented
Mobile U-Net, augmented
Predictions :

Figure 20

7.2 Models accuracy

Accuracy values based on the predictions made on the test datasets have been computed. *Table 2* shows the average overall accuracy and the average class accuracy for each training with the Vaihingen dataset. In a similar way, *table 3* shows the accuracy results for the models trained with the OFS-Mos25 dataset. For each column, minimum values are highlighted in red and maximum values in blue.

7.2.1 Vaihingen datasets

The best overall accuracy on the Vaihingen dataset is obtained with 82.53% by the SegNet architecture without image augmentation and with $768 * 768$ patches. The worse overall accuracy is for the Mobile U-Net on the non-augmented Vaihingen small patches dataset, reaching only 72.37%. When looking at the accuracy values for each class (*Table 2*), SegNet is more often attributed with the highest accuracy values than Mobile U-Net, and U-Net obtains most of the lowest values. The usage of $768 * 768$ patches over the $512 * 512$ ones gave a better accuracy on the "*Impervious surface*" class objects which represent the majority of pixels in the dataset, as well as for the "*Low vegetation*" class. This has for effect to push the overall accuracy up. In general, buildings and trees accuracies are higher with the smaller patches. The usage of image augmentation does not appear to be of any help as it can perform as many high accuracies as bad ones depending on the class.

The class that generally obtains the best accuracy is the "*Building*" class, with an average of about 91% between all the models. The "*Clutter / background*" class is the least represented class within the dataset and is not well predicted by the models, but scores high values of accuracy. This happens because the algorithm that computes the accuracies give a value of 100% to the classes that are not present in the test input images. This is often the case for this class that has a very heterogeneous spatial distribution in the images. The impact of this imprecision is minimal as this class has a minimal weight in the overall accuracy. The lowest accuracy is for the "*Cars*" class, with an average of about 52%. This can be expected as it represents only 1.21% of the dataset's pixels.

Accuracies - Vaihingen dataset							
Training ID	Overall accuracy	Impervious surface	Buildings	Trees	Cars	Clutter / Background	Low vegetation
VAI_512_UNet_NA	72,37	75,26	90,46	77,56	43,90	90,63	56,03
VAI_512_UNet_A	74,54	78,72	92,18	84,86	50,09	90,83	49,05
VAI_512_Seg_NA	78,09	77,45	94,26	84,94	55,21	90,59	67,35
VAI_512_Seg_A	76,45	69,20	94,03	84,17	49,73	91,03	61,81
VAI_768_UNet_NA	76,55	78,84	87,49	80,59	42,21	88,91	62,39
VAI_768_UNet_A	80,83	82,91	87,14	81,49	50,85	88,90	66,83
VAI_768_Seg_NA	82,53	82,58	89,29	82,51	58,87	89,35	76,42
VAI_768_Seg_A	82,49	81,24	90,95	82,36	63,67	90,16	66,97
Average:	**77,98**	**78,28**	**90,73**	**82,31**	**51,82**	**90,05**	**63,36**

Table 2

Accuracies - OFS-Mos25 dataset											
Training ID	Overall accuracy	Buildings	Transport	Special infrastructures	Green spaces	Arboriculture	Fields	Pastures	Forests	Lakes & rivers	Unproductive surface
SUI_512_UNet_NA	57,42	14,02	8,65	0,44	0,58	17,20	51,51	56,67	68,04	14,90	33,80
SUI_512_UNet_A	68,26	18,63	0,00	0,00	0,00	15,79	66,78	66,42	77,94	14,55	34,33
SUI_512_Seg_NA	64,98	22,04	7,97	0,69	1,03	17,62	64,33	68,59	72,09	18,51	37,19
SUI_512_Seg_A	70,05	15,32	0,02	0,00	0,00	15,79	71,89	68,01	80,54	18,31	34,74
SUI_768_UNet_NA	51,27	20,34	3,93	0,91	1,21	3,33	53,23	45,92	62,49	29,89	33,04
SUI_768_UNet_A	62,51	18,80	0,00	0,00	0,00	0,00	75,13	60,88	71,88	40,06	23,10
SUI_768_Seg_NA	60,83	24,52	7,02	0,99	1,88	2,73	68,74	61,37	73,48	39,12	28,18
SUI_768_Seg_A	66,61	27,95	0,07	0,00	0,00	0,05	73,60	75,46	76,72	32,80	35,20
Average:	**62,74**	**20,20**	**3,46**	**0,38**	**0,59**	**9,06**	**65,65**	**62,92**	**72,90**	**26,02**	**32,45**

Table 3

7.2.2 OFS-Mos25 datasets

For the OFS-Mos25 datasets, best overall accuracy is delivered by the SegNet model with image augmentation and patches of $512 * 512$ pixels, with 70.05%. The lowest overall accuracy is of 51.27% and is obtained by the Mobile U-Net on the $768 * 768$ patches without image augmentation. The overall accuracy reaches better levels when training the CNNs with smaller patches. This is relevant as well when going through each class, except for the "*Lakes & rivers*" that seems to behave better with $768 * 768$ patches. In general, SegNet performs better than Mobile U-Net as it totalizes more of the highest accuracies and less of the lowest amongst the classes.

"*Forests*" and "*Fields*" classes score the best with respectively 72.9% and 65.65% average accuracy between all the models. "*Transport*", "*Special infrastructure*" and "*Green spaces*" obtains the lowest accuracies. They systematically obtain a value of 0% when image augmentation is used, because the models operate a smoothing that incorporates them in other predominant classes like "*Fields*", "*Pastures*" and "*Forests*". This happens in the advantage of all the other classes that see their respective accuracy levels increase by a few percent.

8 Discussion

This chapter comes back on the initial hypothesis of this project and tries to answer the underlying questions in view of the results obtained with the different training settings. For the Vaihingen dataset, $768 * 768$ patches and SegNet without image augmentation was probably be the best alternative in this context. On the other hand, the highest accuracy for the OFS-Mos25 dataset was obtained with the augmented $512 * 512$ patches and the SegNet architecture again. This model seems to be more interesting in terms of accuracy for semantic segmentation on both low and high resolution satellite data.

In comparison with the overall accuracies that have been achieved by the participants of the "*Vaihingen: 2D Labelling challenge*"[17], an accuracy of 82.53% on the Vaihingen dataset is an interesting result considering the SegNet architecture that was not specifically designed for this kind of task. However, results obtained with the OFS-Mos25 did not reach satisfactory accuracies, possibly because the dataset is "crafted" with different sources from different years. When comparing the accuracy graphics in *Annex 1*, it is noticeable that e validation accuracy values for each non-augmented data trainings has a tendency to decline after about 100 epochs. This could be associated to an over-fitting of the model to the training data. The usage of image augmentation seems to counter this problem but the learning becomes slower and there is more variance in the accuracy. It can however potentially be interesting to develop an architecture dedicated to low resolution satellite imagery, as long as adequate training data is available. This could make a systematic and automated application possible for semantic segmentation of low resolution remote sensing data as modifications of the land cover at such scale are not significant in long time lapses.

The main issues for both datasets remain in the fact that training data should be more abundant and more balanced in terms of classes. The other problem that

[17] 140 valid participations, with accuracies going from 78.4% to 91.6%, average of 88.23%. Full results table is available at http://www2.isprs.org/commissions/comm2/wg4/vaihingen-2d-semantic-labeling-contest.html

was observed in the case of the Vaihingen dataset is that some shapes appearing in the ground truth images were over-simplified and CNNs tended to be closer to reality for smaller shapes, like small alleys going from roads to houses.

It has been observed in the various experiments that CNNs encounter difficulties to reproduce the sharp shapes of human built structures. Until another solution is found, this kind of issue could possibly be corrected with post-processing methods as suggested in many papers (Badrinarayanan, Kendall, & Cipolla, 2017; Chen, Papandreou, Kokkinos, Murphy, & Yuille, 2016; Pinheiro & Collobert, 2015; Yao, Poleswki, & Krzystek, 2016). It is however possible for CNN to reach high accuracy levels without post-processing (Long, Shelhamer, & Darrell, 2015; Volpi & Tuia, 2016), which is always more interesting in terms of data processing efficiency.

When looking at the test image outputs of each model and evaluating them with a human perception, the shapes of the segmented regions looked globally more natural on the Vaihingen dataset when using image augmentation techniques. These were however not helpful in most situations in terms of accuracy. This can probably be related to the specificities of the data augmentation algorithm of the *Semantic Segmentation Suite*, which could be adjusted specifically for processing remote sensing data. It may also come from a problem of poor balancing of the classes in both datasets. In such situation, the usage of class balancing method as the median class balancing suggested in the SegNet paper (Badrinarayanan, Kendall, & Cipolla, 2015) should be helpful, but was not fully integrated in the *Semantic Segmentation Suite* yet.

For CNN that are intended to a particular application, duration of training does not matter much if the model can be used afterwards for predictions over a long period of time. However, the time required to make predictions once the model is trained can be a more critical aspect depending on the purpose of the potential applications. Mobile U-Net seems faster in that sense, but clearly achieves lower accuracies. In the context of remote sensing and more specifically land cover classifications, the phenomena that are studied do not necessarily require a real

time analysis. This means that the development of deeper and heavier CNN should be an interesting option with the advent of new generation GPU or TPU, as long as over-fitting does not become an issue.

Scale seems to have an important influence as well for semantic segmentation with CNN. In the case of the OFS-Mos25 dataset, the shape that the CNNs need to recognize are more heterogeneous, and the precision of the predictions that needs to be achieved is higher. For low resolution remotely sensed imagery, the pixels are always a mixture of several semantic land cover objects and despite all the training that can be performed, the choice of the good label is not an evidence for a CNN, nor for a human eye. This issue of spatial precision would probably not be as much of a handicap if the up-sampling method with indices was available in the *Semantic Segmentation Suite* and in general in deep learning frameworks for the SegNet architecture.

The number of examples for training of the CNNs matters a lot in the context of CNN. Compared to the Vaihingen dataset, the OFS-Mos25 dataset has 15% less pixels that the models can learn from. Also, it has more classes that have to share these pixels. This is without a doubt an element that could explain the large difference in the accuracy obtained between the two datasets. With CNN and ANN in general, the ability for the model to build a bank of features that helps creating the network's low and high level representations are always highly dependent on the quality of the datasets, in terms of quantity of data, balance between the classes and quality of the ground truth. This raises the question of the limits of ANN regarding remote sensing data. Each territory has its own characteristics and data capture conditions, which also change with time. CNN architectures that are dedicated to remote sensing imagery should be easily upgradable, and able to properly learn these variances in the data. This could be possible for instance with fine-tuning of the parameters over time.

In summary and in the attempt to give a proper answer to the research question of this project, CNNs are adapted tools that can efficiently provide accurate semantic segmentation of remote sensing data. The perspectives for an automated process are also very promising. Reservations can however be made

regarding the multi-scale aspect. The two models that were tested seem to have more difficulties with the low resolution dataset. They could possibly be adapted for better predictions, essentially if they can be provided with larger dataset of better quality. The challenge for more accuracy is in both data acquisition and model architecture. For decision oriented applications, a transition to CNN for semantic segmentation of remotely sensed data should slowly begin and have a bright future, in the same way that ANN have progressively replaced traditional machine learning algorithm in many computer vision web based applications.

9 Conclusion

The objective of this experiment was to evaluate how the usage of CNN could achieve highly accurate semantic segmentation of remote sensing data of different geographical scales. The results of the experiments that have been conducted within this project may show that there are no unique solution in terms of CNN architecture for this particular task. Each training that has been experimented produced models that can accomplish better results that depend on the datasets, the pre-processing method, or even particular semantic classes. Added to this, the visual aspect of the test outputs did not always reflect the actual accuracy achieved by the models.

The tool used for training the models, the *Semantic Segmentation Suite,* is still under development. The ideas behind its conception are user-friendly oriented, with a very readable and customizable code. Even if such open source projects are not always perfect for research purpose, the enthusiastic community slowly makes very promising contributions towards easier experimental usage of ANN. Progresses are also quickly made in the different deep learning platforms and frameworks, which creates a positive dynamic.

CNNs are still new in the field of remote sensing and spatial analysis and are currently an active subject of research. As recognized state-of-art method for image recognition and computer vision in general, CNNs probably still have to show better results for semantic segmentation of remote sensing data in order to be trusted as a tool for decision oriented applications. Acquiring adapted pixel-level labelled data for an efficient training is also a tough task that requires time, money and most of all strictly respected processes. This would however be worth the investments when considering all the potential applications.

The future contributions of CNNs in remote sensing high resolution data semantic segmentation and analysis could be of many forms and could be useful to various fields of today's geography. From the environmental change detection to automated statistics about land use in short time intervals, the scope of application is wide. In Switzerland, the analysis of such data is done over a period

of about 5 years. Considering today's abundance of digital imagery that are captured daily by the numerous satellites orbiting around Earth and with always higher resolution, CNNs could allow more frequent and automated analysis of the territory and therefore a more reasonable and responsible usage of the land.

It would be interesting for future works on semantic segmentation with CNN and remote sensing data to test different settings of image augmentation. It could also be interesting to investigate the possibility to create a larger and more balanced dataset. The development of a dedicated architecture for remote sensing data, with adapted convolution filters trained to detect the key features of the data and with convenient options for fine-tuning over time could be a major step towards generalization of CNN in public or private institutions.

10 References

Albert, A., Kaur, J., & Gonzalez, M. C. (2017). Using Convolutional Networks and Satellite Imagery to Identify Patterns in Urban Environments at a Large Scale (p. 1357-1366).

Badrinarayanan, V., Kendall, A., & Cipolla, R. (2017). SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *39*(12), 2481-2495.

Blaschke, T. (2010). Object based image analysis for remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing*, *65*(1), 2-16.

Castelluccio, M., Poggi, G., Sansone, C., & Verdoliva, L. (2015). Land use classification in remote sensing images by convolutional neural networks. *arXiv preprint arXiv:1508.00092*.

Chapelle, O., Haffner, P., & Vapnik, V. N. (1999). Support Vector Machines for Histogram-Based Image Classification. *IEEE Transactions on Neural Networks*, *10*(5), 10.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2016). DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *ArXiv:1606.00915 [Cs]*.

Chen, X., Xiang, S., Liu, C.-L., & Pan, C.-H. (2013). Vehicle Detection in Satellite Images by Parallel Deep Convolutional Neural Networks (p. 181-185). IEEE.

Congalton, R. G. (1991). A review of assessing the accuracy of classifications of remotely sensed data. *Remote sensing of environment*, *37*(1), 35–46.

Dey, V., Zhang, Y., & Zhong, M. (2010). A review on image segmentation techniques with remote sensing perspective. *ISPRS TC VII Symposium - 100 Years ISPRS*, *38*.

Dhanachandra, N., Manglem, K., & Chanu, Y. J. (2015). Image Segmentation Using K -means Clustering Algorithm and Subtractive Clustering Algorithm. *Procedia Computer Science*, *54*, 764-771.

Dumoulin, V., & Visin, F. (2016). A guide to convolution arithmetic for deep learning. *arXiv:1603.07285 [cs, stat]*.

Egmont-Petersen, M., de Ridder, D., & Handels, H. (2002). Image processing with neural networks—a review. *Pattern recognition*, *35*(10), 2279–2301.

Fu, G., Liu, C., Zhou, R., Sun, T., & Zhang, Q. (2017). Classification for High Resolution Remote Sensing Imagery Using a Fully Convolutional Network. *Remote Sensing*, *9*(6), 498.

Haralick, R. M., & Shapiro, L. G. (1985). Image segmentation techniques. *Computer Vision, Graphics, and Image Processing*, *29*(1), 100-132.

Haykin, S. (1999). *Neural Networks: A Comprehensive Foundation* (2nd éd.). Upper Saddle River, NJ, USA: Prentice Hall PTR.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Identity mappings in deep residual networks. In *European Conference on Computer Vision* (p. 630–645). Springer.

Hecht-Nielsen, R. (1988). Theory of the backpropagation neural network. *Neural Networks*, *1*(Supplement-1), 445–448.

Hinton, G., Osindero, S., Welling, M., & Teh, Y.-W. (2006). Unsupervised Discovery of Nonlinear Structure Using Contrastive Backpropagation. *Cognitive Science*, *30*(4), 725-731.

Huang, C., Davis, L. S., & Townshend, J. R. G. (2002). An assessment of support vector machines for land cover classification. *International Journal of Remote Sensing*, *23*(4), 725-749.

Jianbo Shi, & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *22*(8), 888-905.

Kanevski, M., Munoz-Marı, J., Tuia, D., & Volpi, M. (2011). A survey of active learning algorithms for supervised remote sensing image classification. *IEEE Journal of Selected Topics in Signal Processing*, *5*(3), 606-617.

Kitchin, R., & Thrift, N. J. (Éd.). (2009). *International encyclopedia of human geography* (First edition). Amsterdam: Elsevier.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84-90.

Längkvist, M., Kiselev, A., Alirezaie, M., & Loutfi, A. (2016). Classification and Segmentation of Satellite Orthoimagery Using Convolutional Neural Networks. *Remote Sensing*, *8*(4), 329.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436-444.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (p. 3431–3440).

Lu, D., & Weng, Q. (2007). A survey of image classification methods and techniques for improving classification performance. *International Journal of Remote Sensing*, *28*(5), 823-870.

Marmanis, D., Wegner, J. D., Galliani, S., Schindler, K., Datcu, M., & Stilla, U. (2016). Semantic segmentation of aerial images with an ensemble of cnns. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *3*, 473.

Mas, J. F., & Flores, J. J. (2008). The application of artificial neural networks to the analysis of remotely sensed data. *International Journal of Remote Sensing*, *29*(3), 617-663.

Mountrakis, G., Im, J., & Ogole, C. (2011). Support vector machines in remote sensing: A review. *ISPRS Journal of Photogrammetry and Remote Sensing*, *66*(3), 247-259.

Noh, H., Hong, S., & Han, B. (2015). Learning Deconvolution Network for Semantic Segmentation. *arXiv:1505.04366 [cs]*.

Pal, N. R., & Pal, S. K. (1993). A review on image segmentation techniques. *Pattern Recognition*, *26*(9), 1277-1294.

Pinheiro, P. O., & Collobert, R. (2015). From image-level to pixel-level labeling with convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (p. 1713–1721).

Ren, & Malik. (2003). Learning a classification model for segmentation (p. 10-17 vol.1). IEEE.

Richards, J. A., & Jia, X. (2006). *Remote sensing digital image analysis: an introduction* (4th ed). Berlin: Springer.

Ronneberger, O., Fischer, P., & Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (p. 234–241). Springer.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386-408.

Schmidhuber, J. (2015). Deep Learning in Neural Networks: An Overview. *Neural Networks*, *61*, 85-117.

Sharma, A., Liu, X., Yang, X., & Shi, D. (2017). A patch-based convolutional neural network for remote sensing image classification. *Neural Networks*, *95*, 19-28.

Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv:1409.1556 [Cs]*.

Tang, T., Zhou, S., Deng, Z., Zou, H., & Lei, L. (2017). Vehicle Detection in Aerial Images Based on Region Convolutional Neural Networks and Hard Negative Example Mining. *Sensors*, *17*(2), 336.

Szeliski, R. (2011). *Computer Vision: Algorithms and Applications*. London: Springer-Verlag.

Thoma, M. (2016). A Survey of Semantic Segmentation. *ArXiv:1602.06541 [Cs]*.

Unger Holtz, T. S. (2007). Introductory Digital Image Processing: A Remote Sensing Perspective, Third Edition. *Environmental and Engineering Geoscience*, *13*(1), 89-90.

Volpi, M., & Tuia, D. (2016). Dense Semantic Labeling of Subdecimeter Resolution Images With Convolutional Neural Networks. *IEEE*, *55*(2), 881-893.

Wang, X.-Y., Wang, T., & Bu, J. (2011). Color image segmentation using pixel wise support vector machine classification. *Pattern Recognition*, *44*(4), 777-787.

Yao, W., Poleswki, P., & Krzystek, P. (2016). Classification of urban aerial data based on pixel labelling with deep convolutional neural networks and logistic regression. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLI-B7*, 405-410.

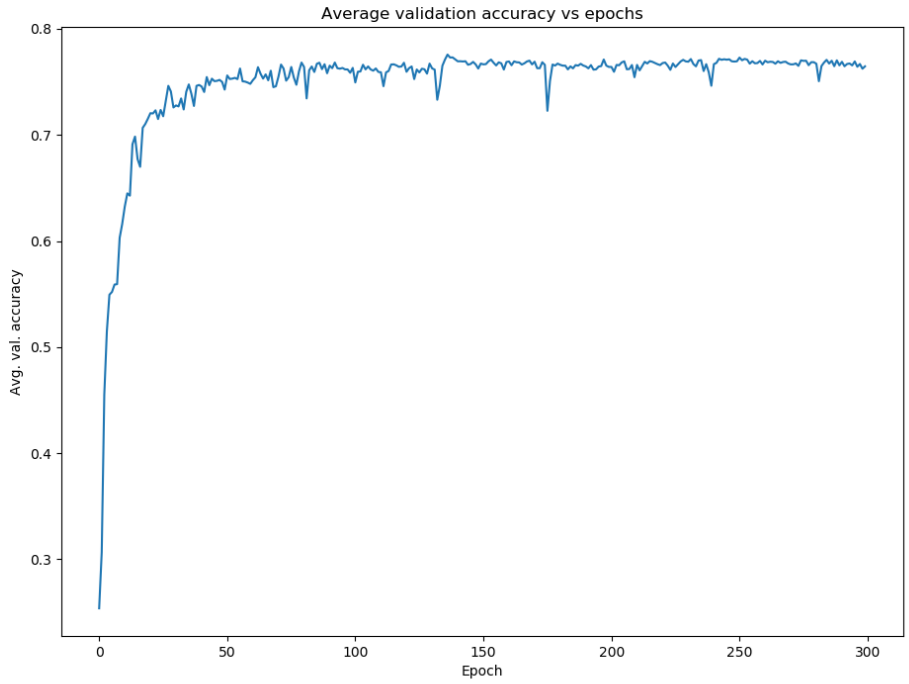
Young, N. E., Anderson, R. S., Chignell, S. M., Vorster, A. G., Lawrence, R., & Evangelista, P. H. (2017). A survival guide to Landsat preprocessing. *Ecology*, *98*(4), 920-932.

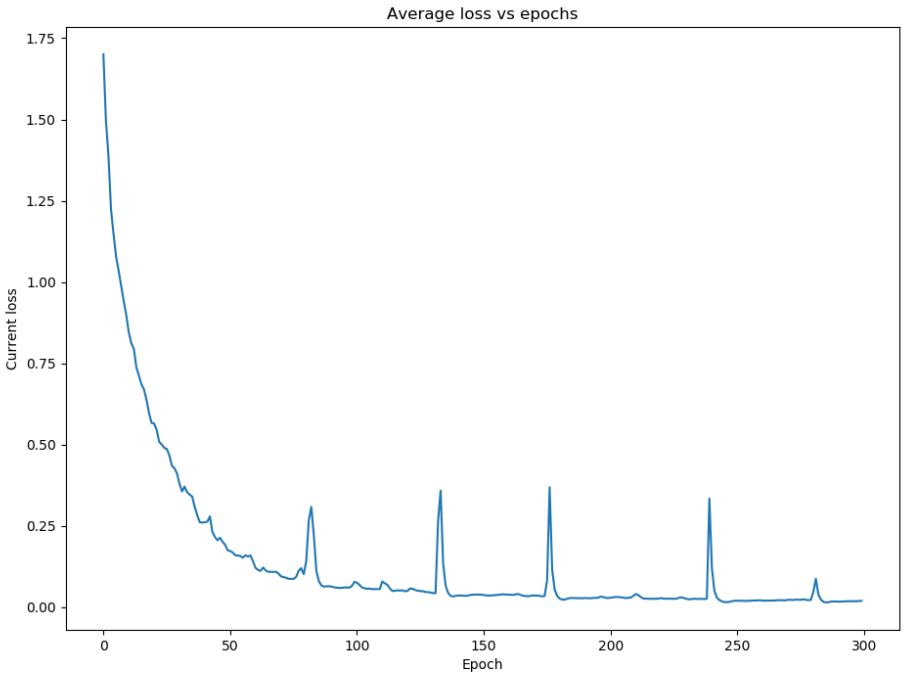
Yu, X., Wu, X., Luo, C., & Ren, P. (2017). Deep learning in remote sensing scene classification: a data augmentation enhanced convolutional neural network framework. *GIScience & Remote Sensing*, *54*(5), 741-758.

11 Annexes

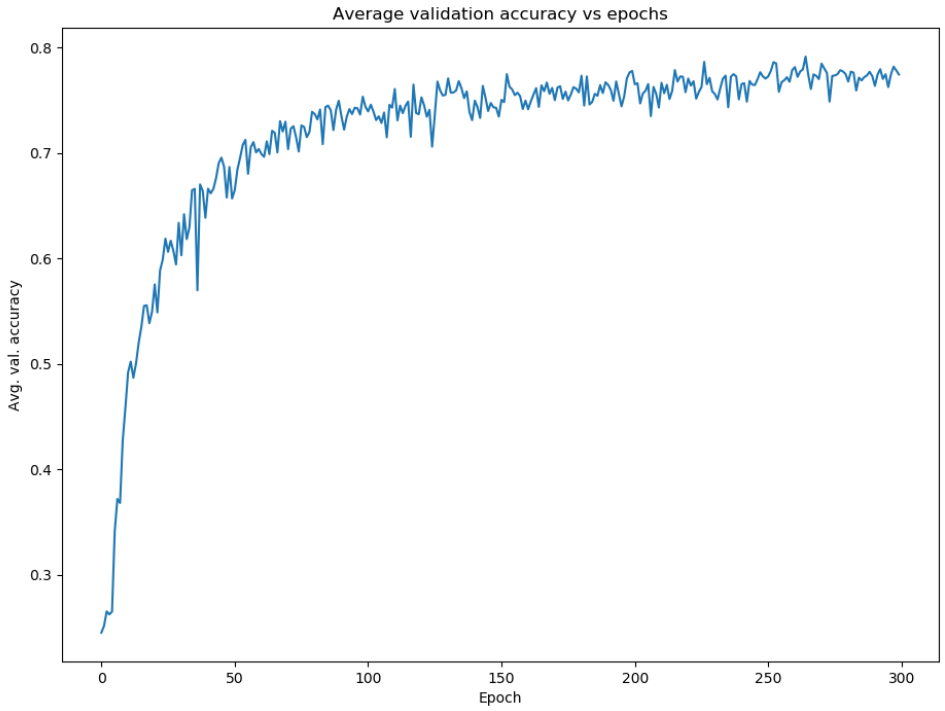
11.1 Annex 1 – Accuracy and loss graphics

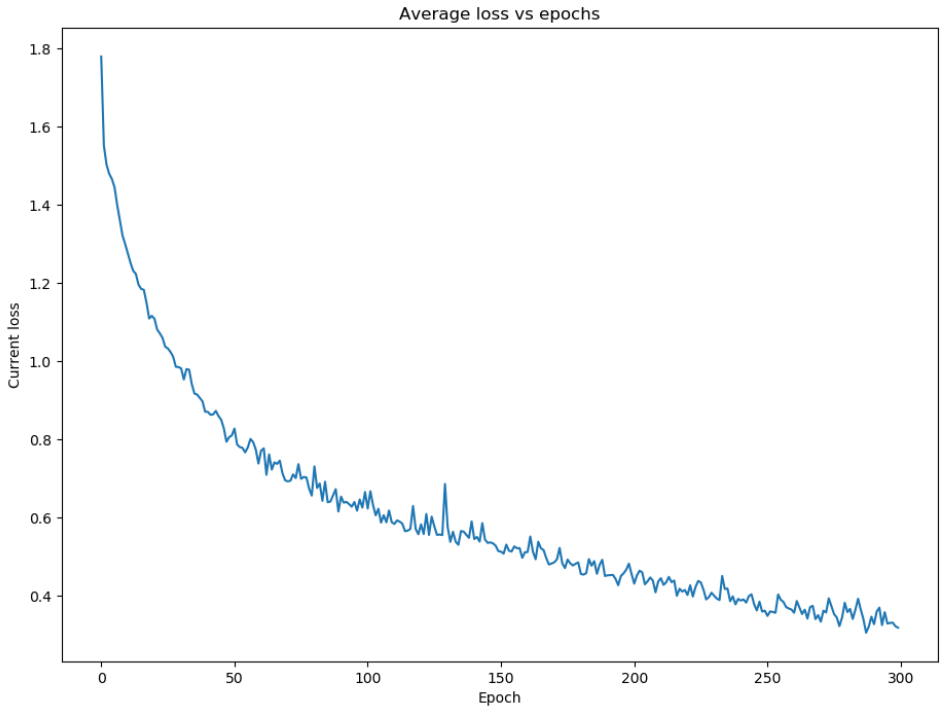
VAI_512_UNet_NA



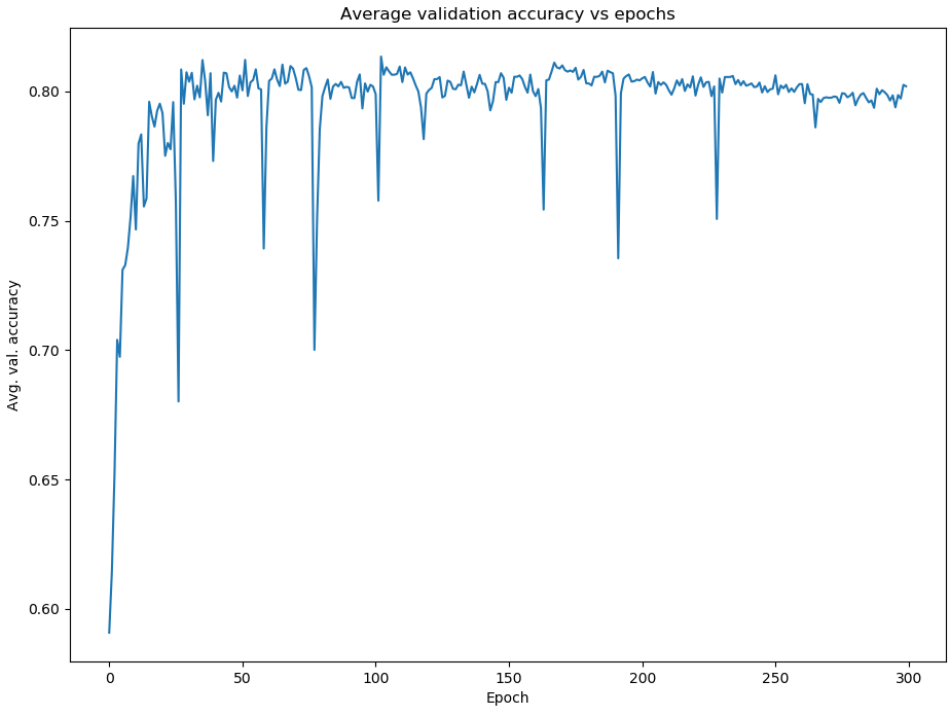


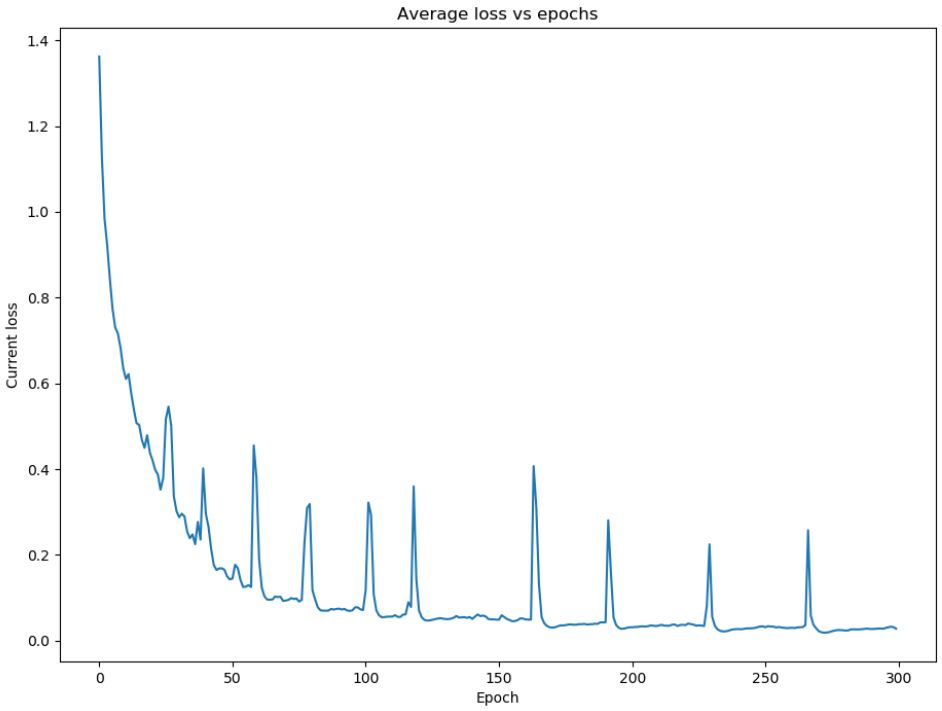
VAI_512_UNet_A



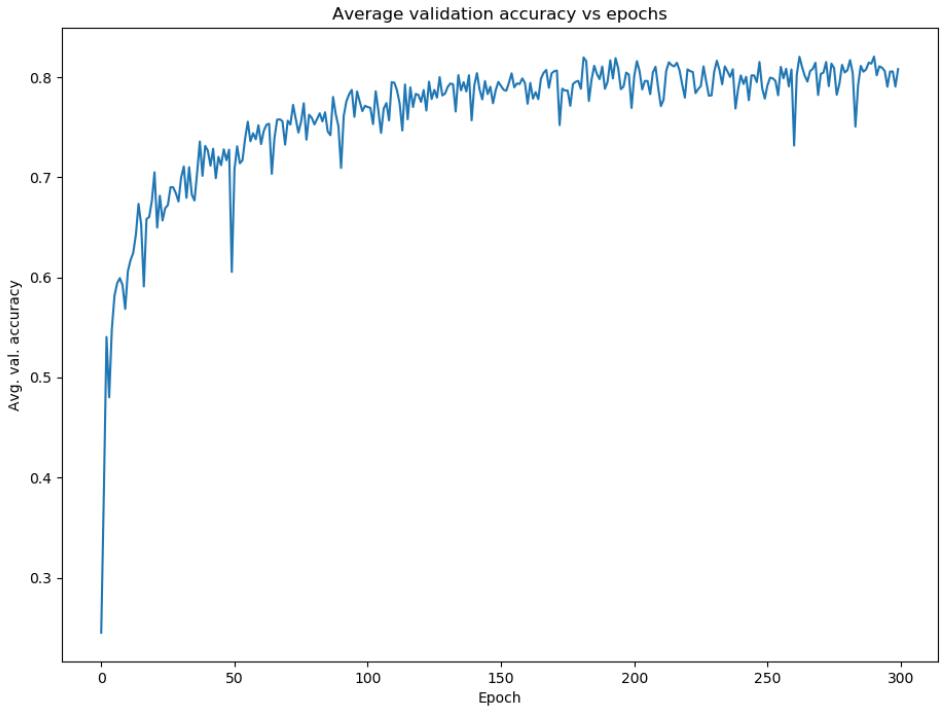


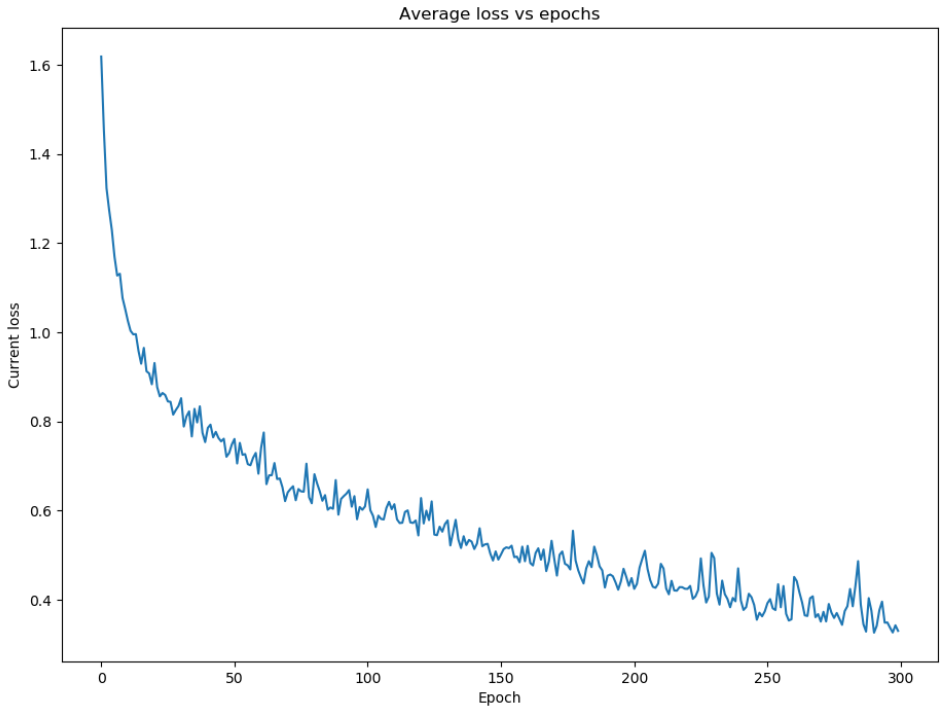
VAI_512_Seg_NA



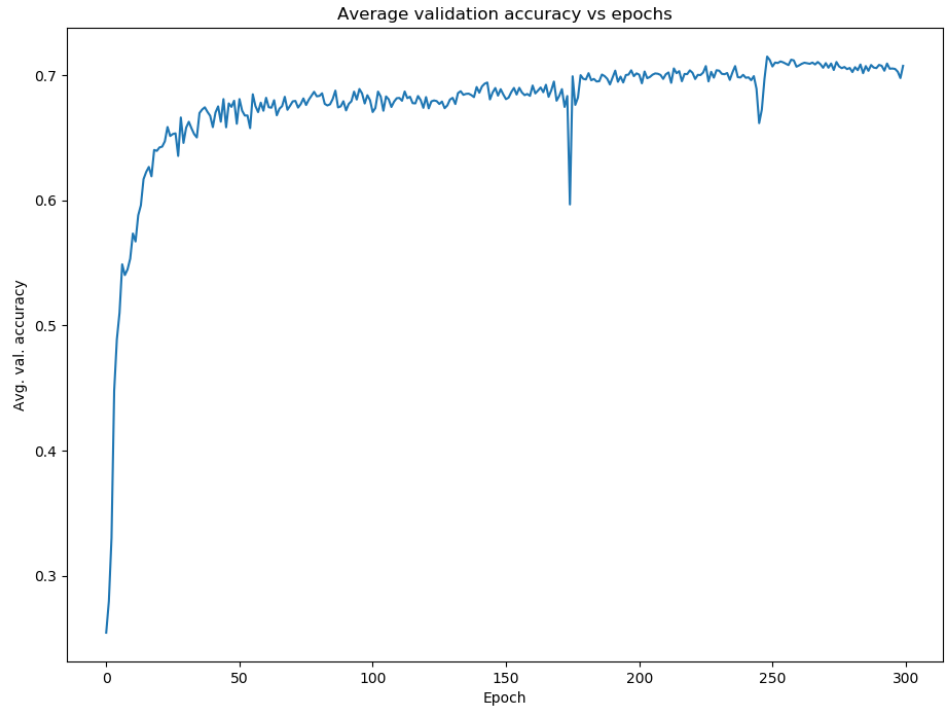


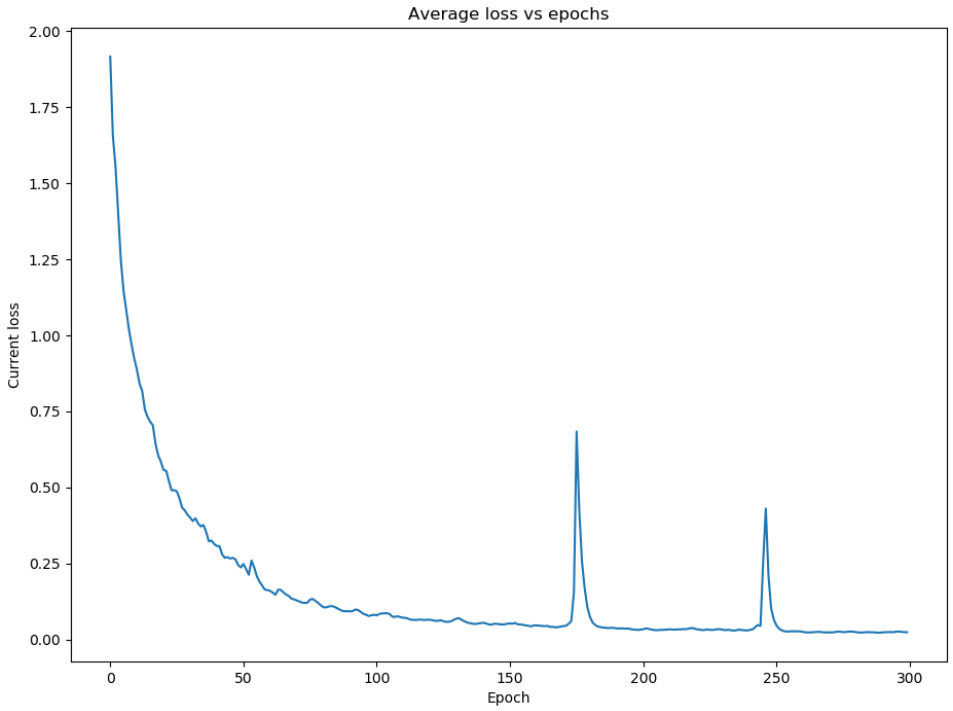
VAI_512_Seg_A


Average validation accuracy vs epochs
Avg. val. accuracy
Epoch
0.3
0.4
0.5
0.6
0.7
0.8
0
50
100
150
200
250
300

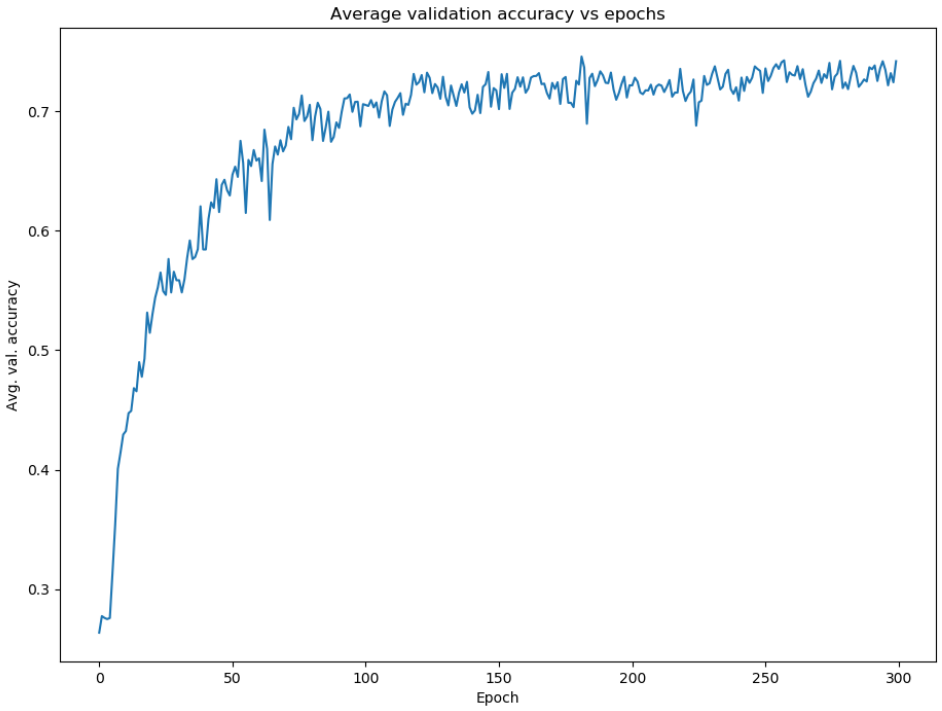

Average loss vs epochs
Current loss
Epoch
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
50
100
150
200
250
300

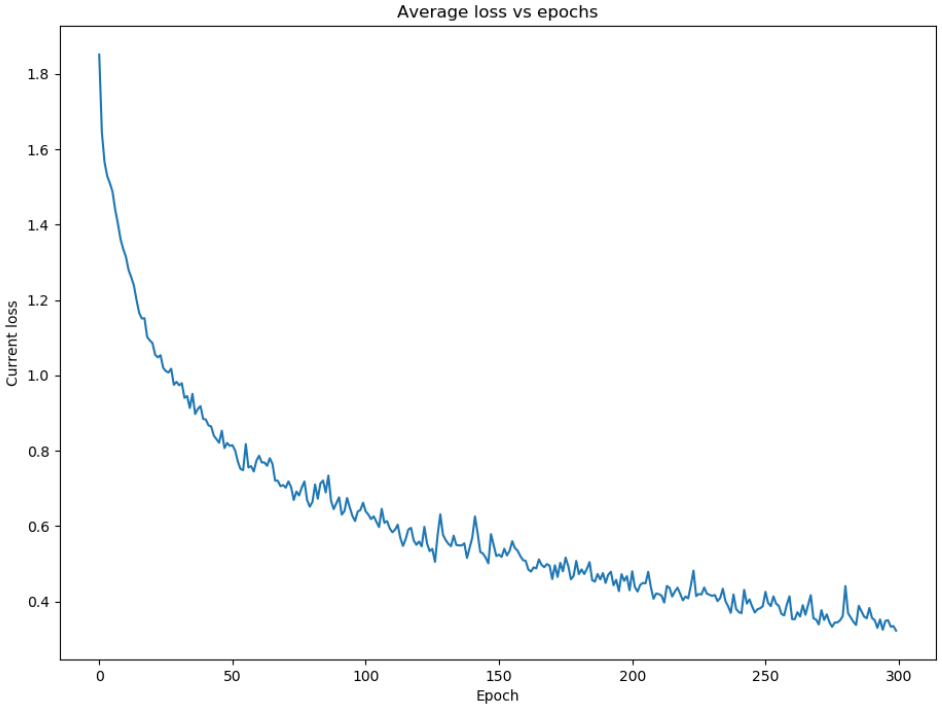
VAI_768_UNet_NA



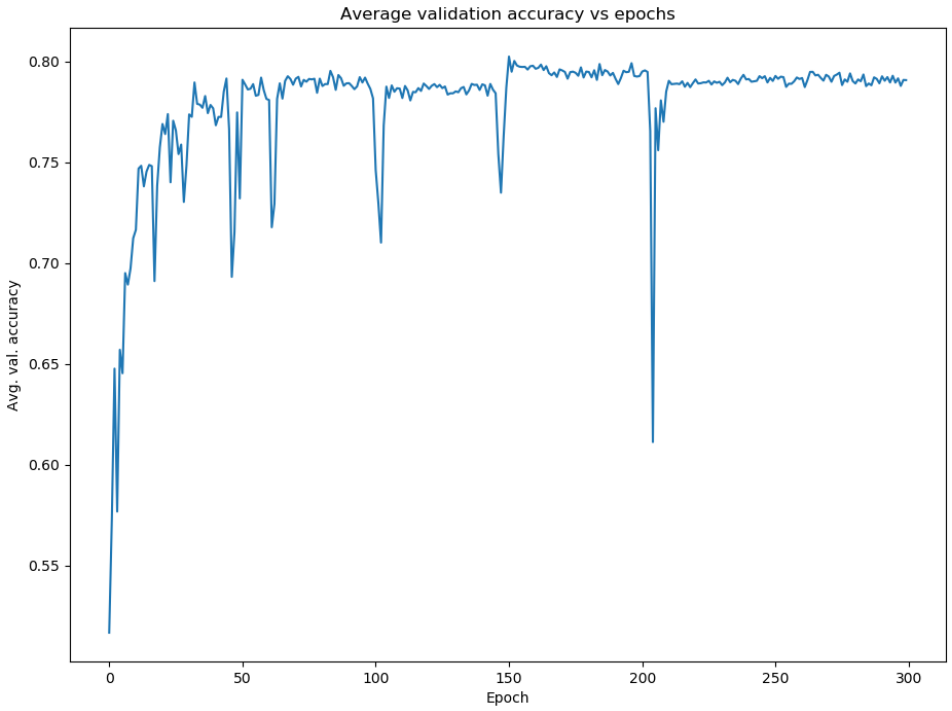


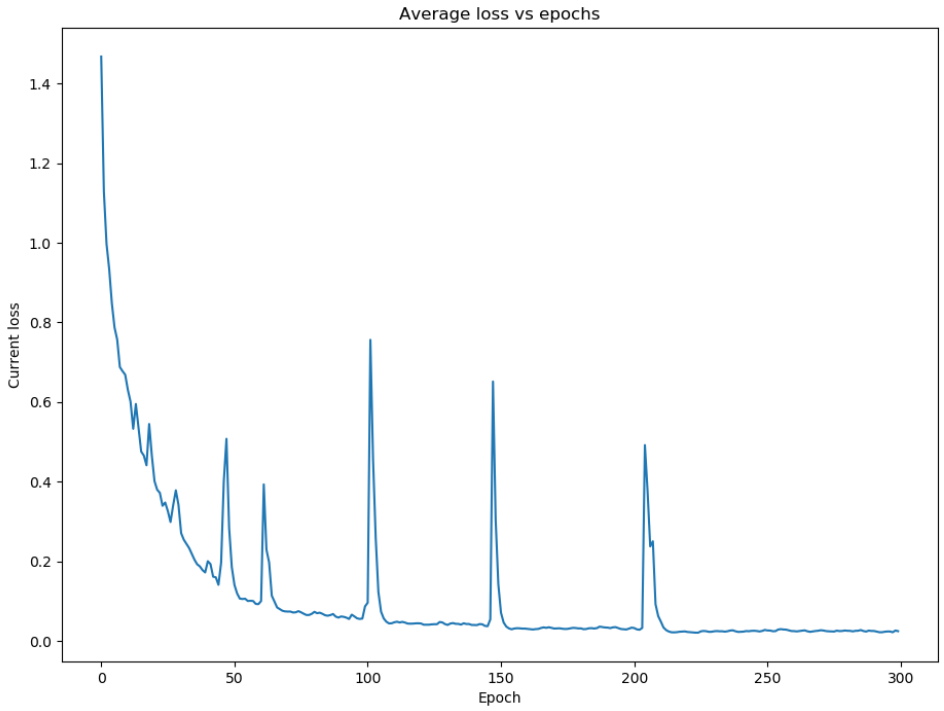
VAI_768_UNet_A



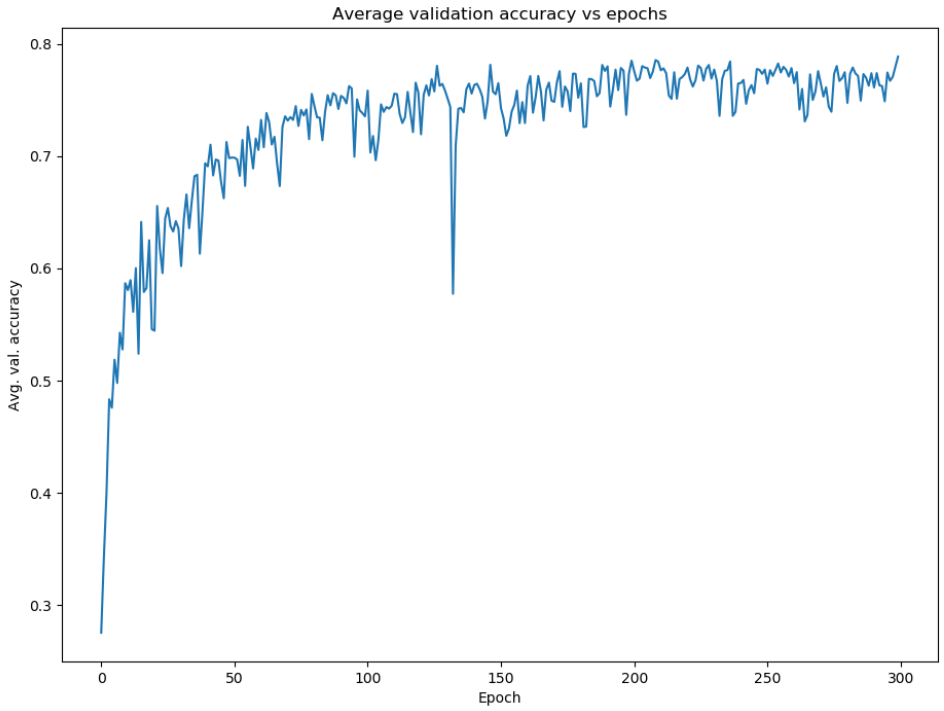


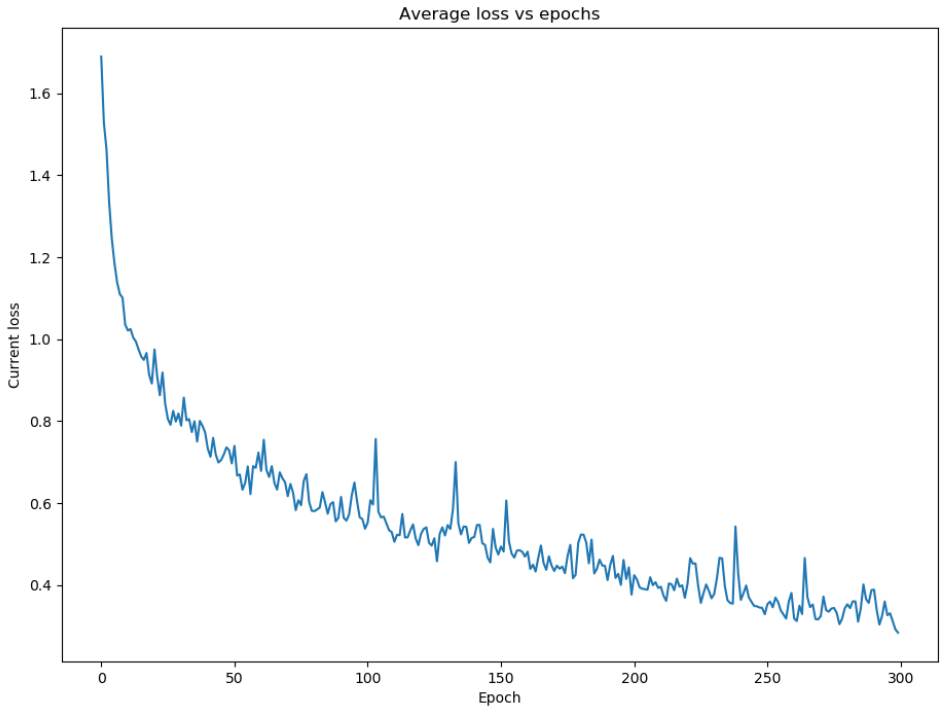
VAI_768_Seg_NA



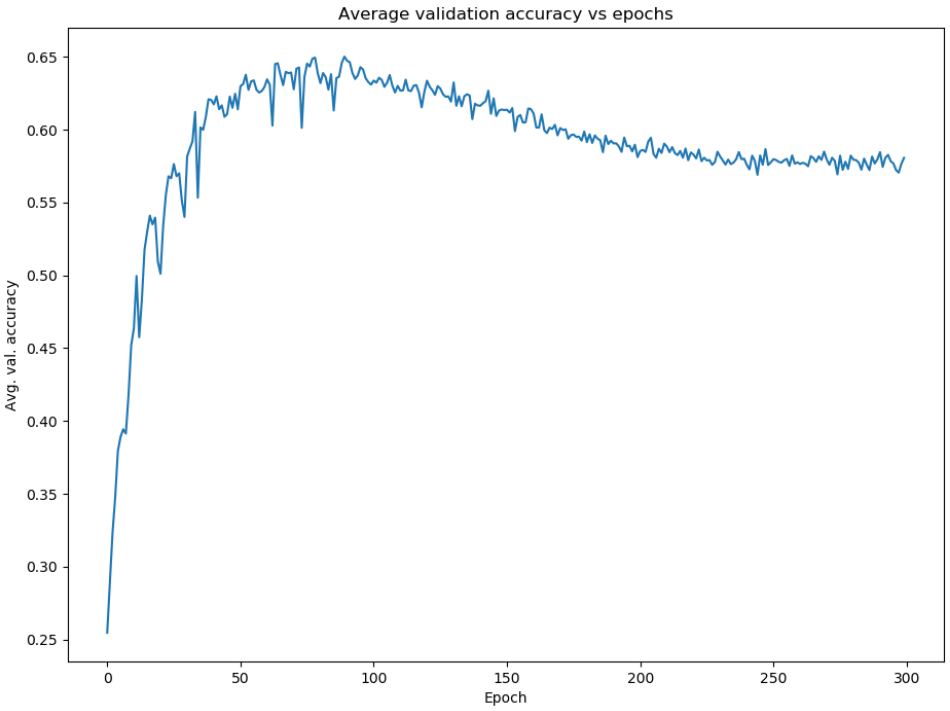


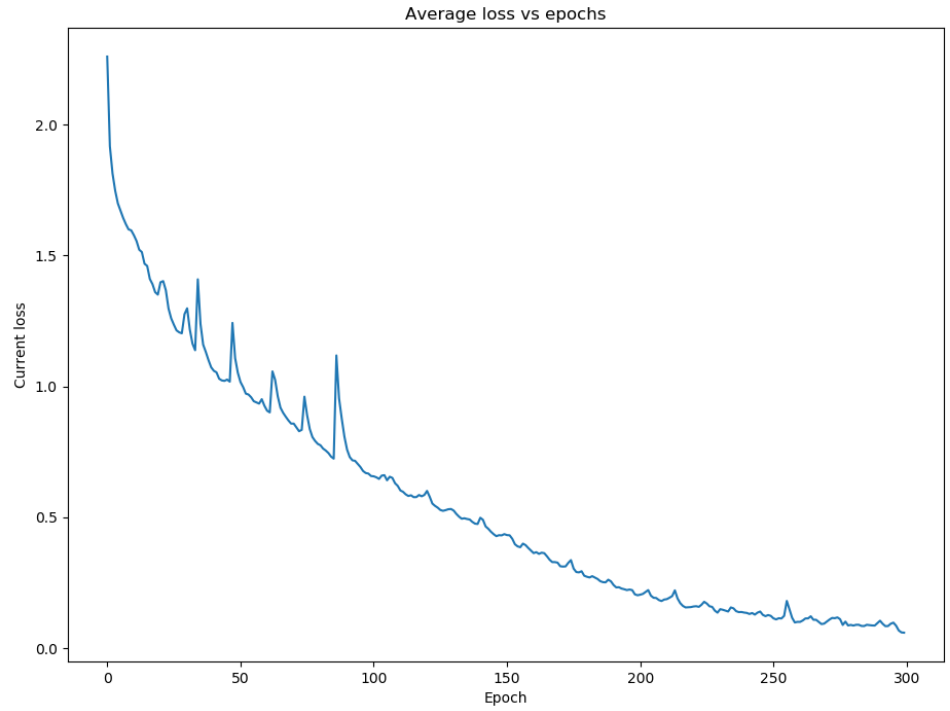
VAI_768_Seg_A


Average validation accuracy vs epochs
Avg. val. accuracy
Epoch
0.3
0.4
0.5
0.6
0.7
0.8
0
50
100
150
200
250
300

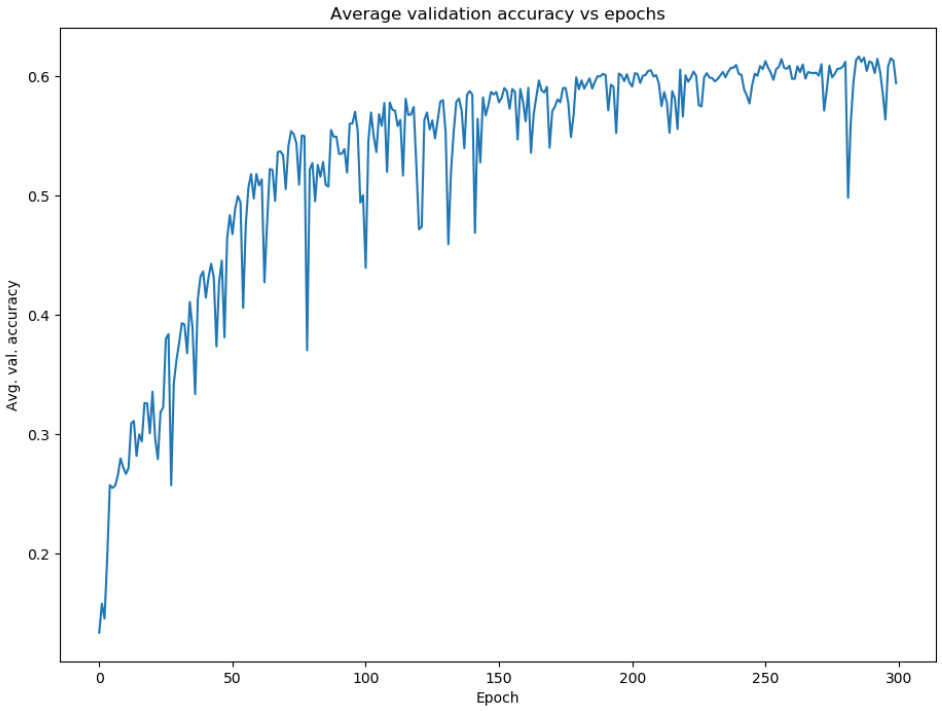

Average loss vs epochs
Current loss
Epoch
0.4
0.6
0.8
1.0
1.2
1.4
1.6
0
50
100
150
200
250
300

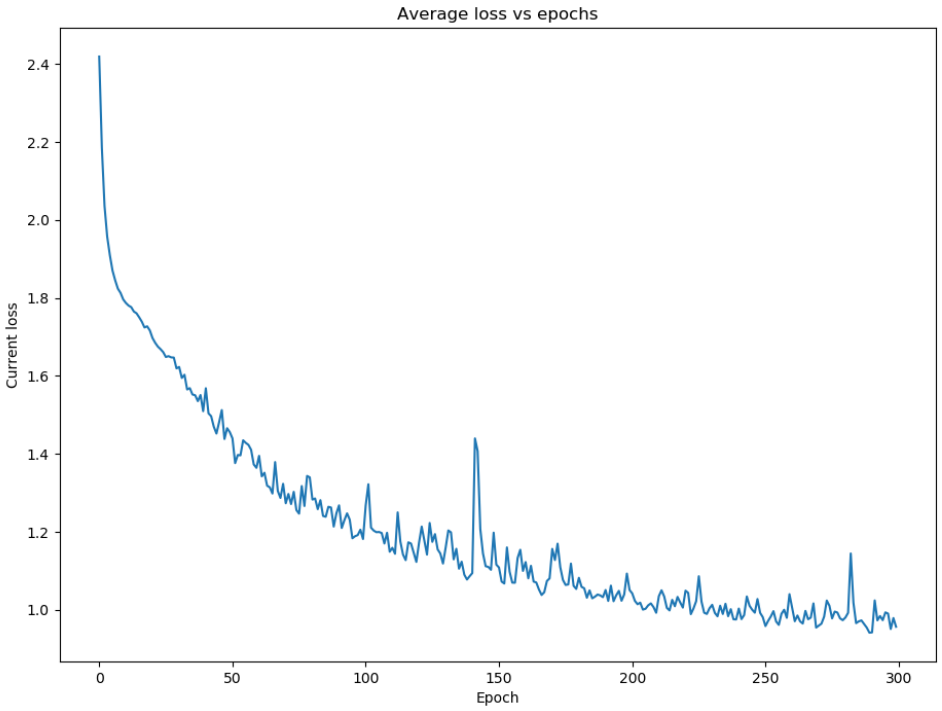
SUI_512_UNet_NA



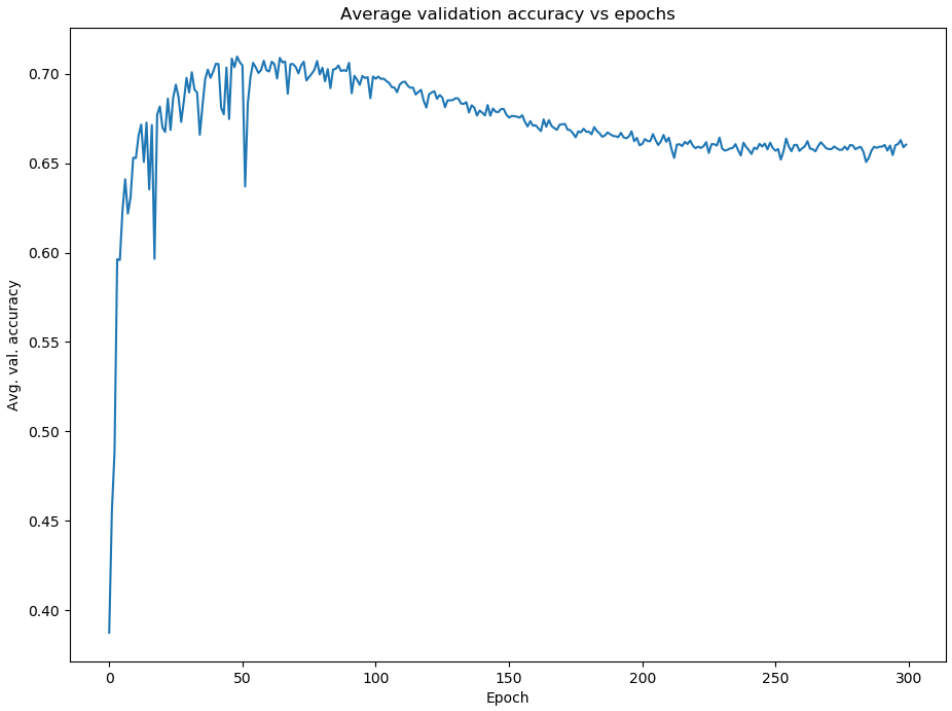


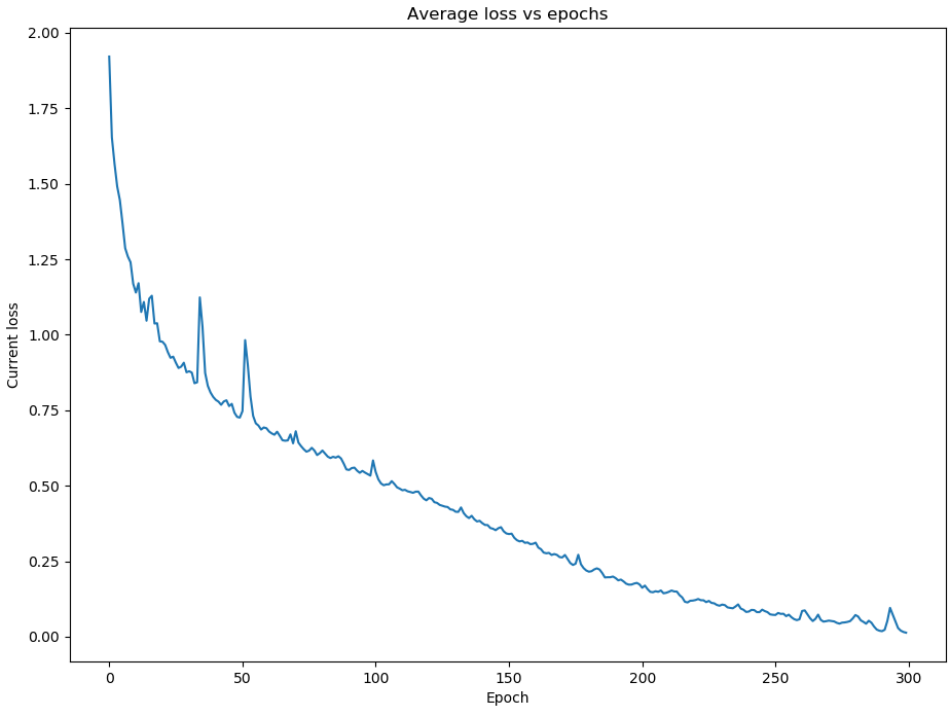
SUI_512_UNet_A



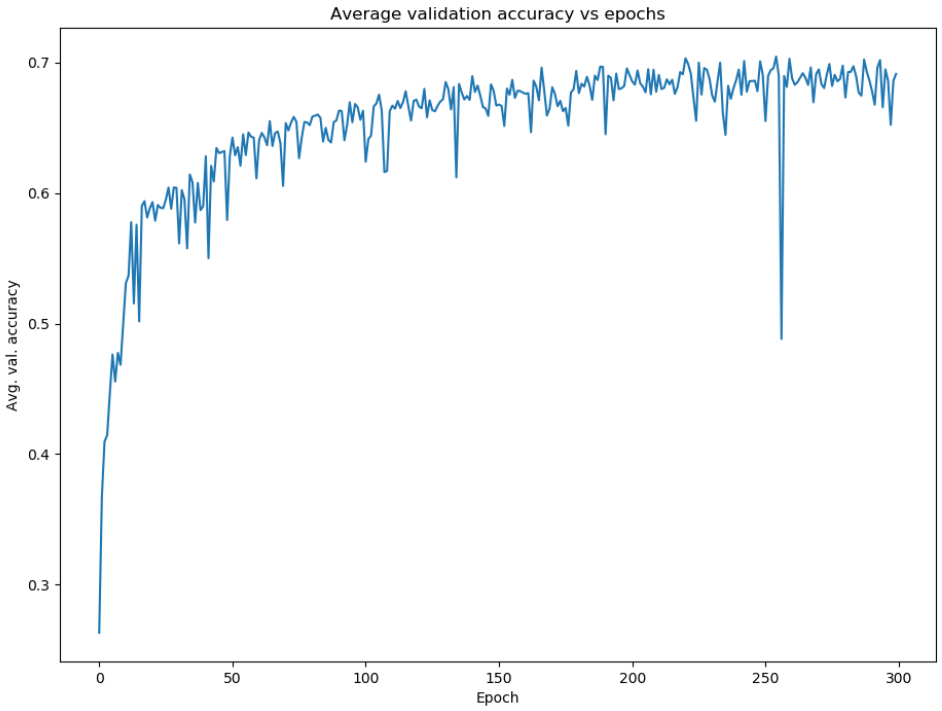


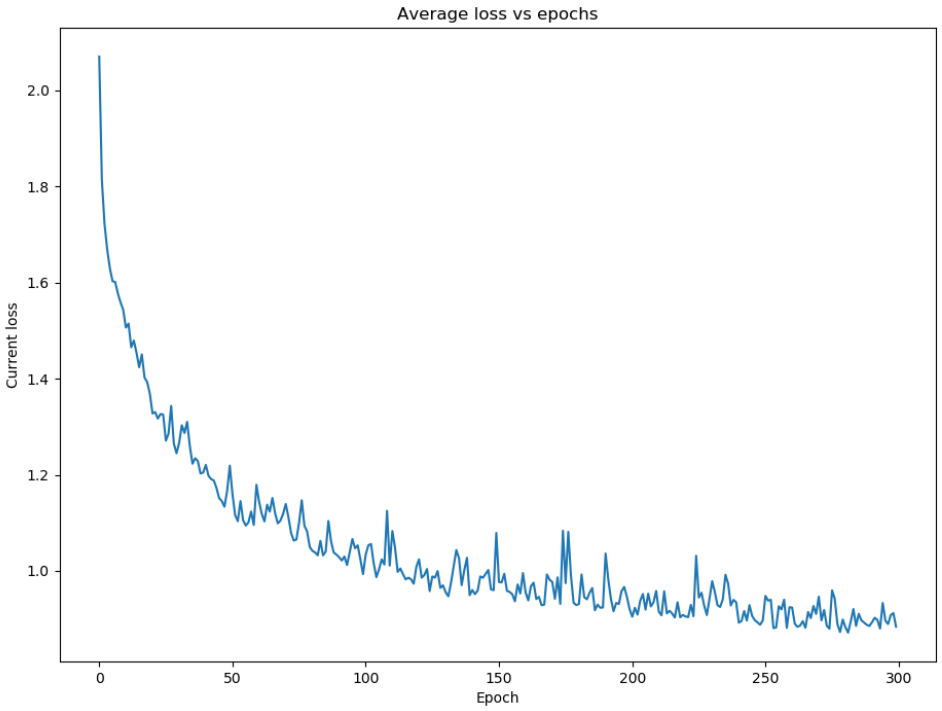
SUI_512_Seg_NA



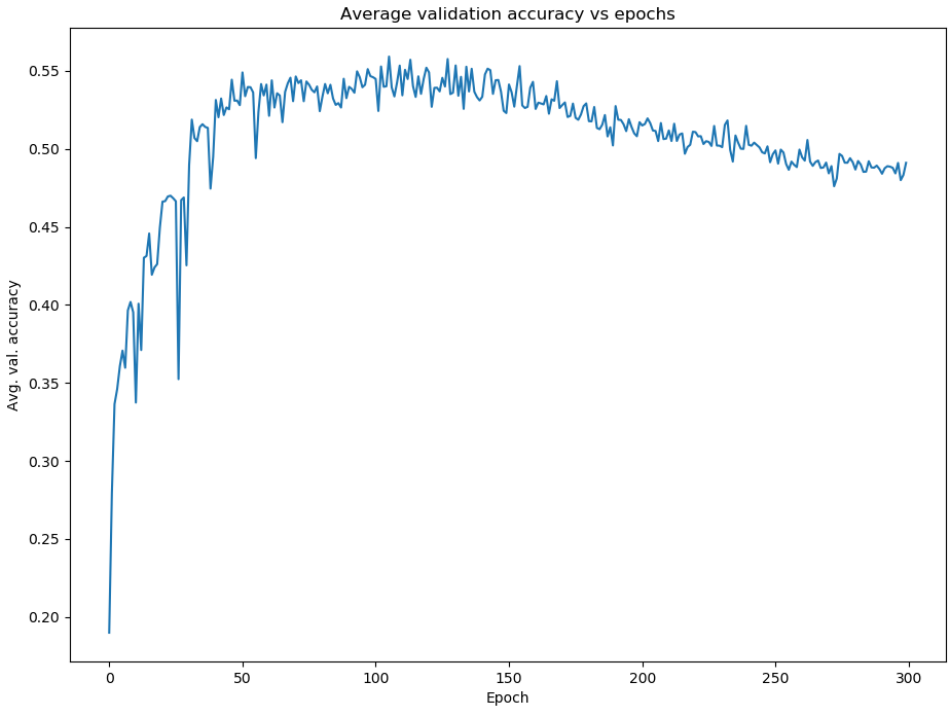


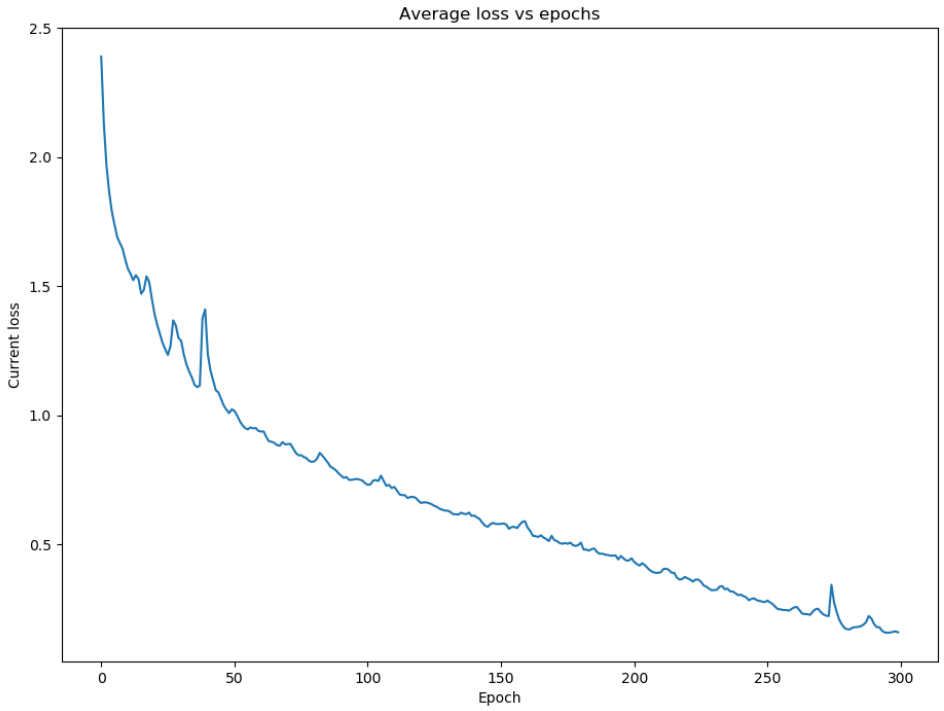
SUI_512_Seg_A


Average validation accuracy vs epochs
Avg. val. accuracy
0.7
0.6
0.5
0.4
0.3
0
50
100
150
200
250
300
Epoch

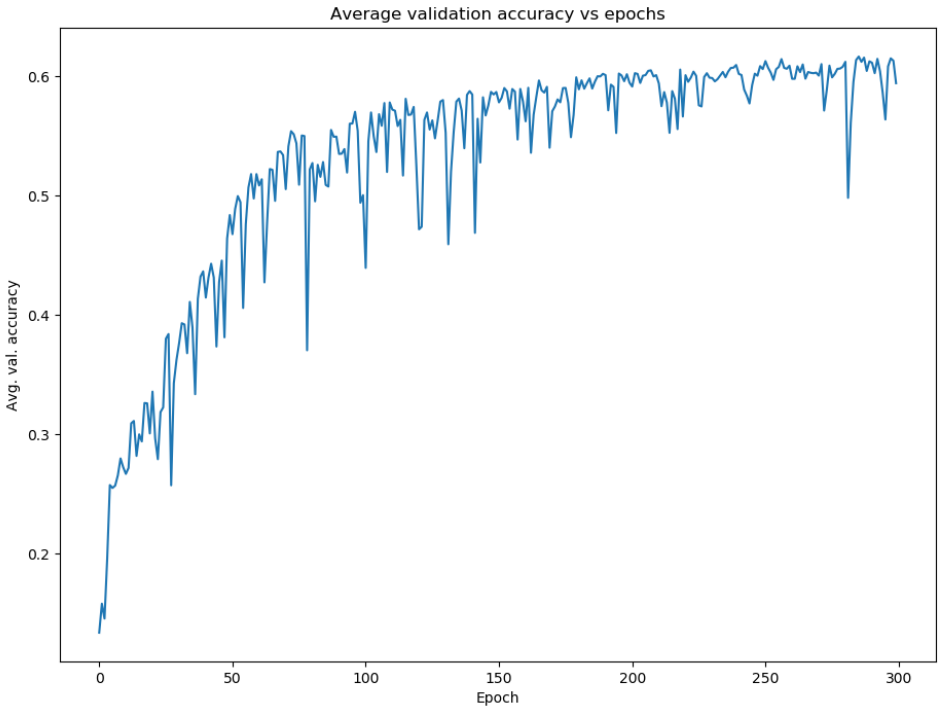

Average loss vs epochs
Current loss
2.0
1.8
1.6
1.4
1.2
1.0
0
50
100
150
200
250
300
Epoch

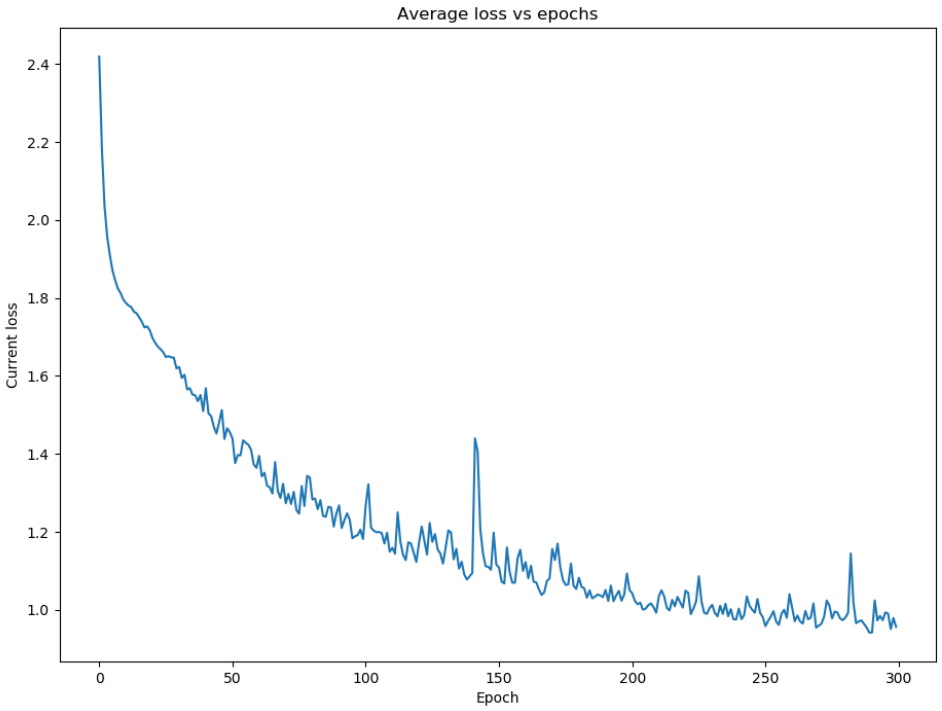
SUI_768_UNet_NA



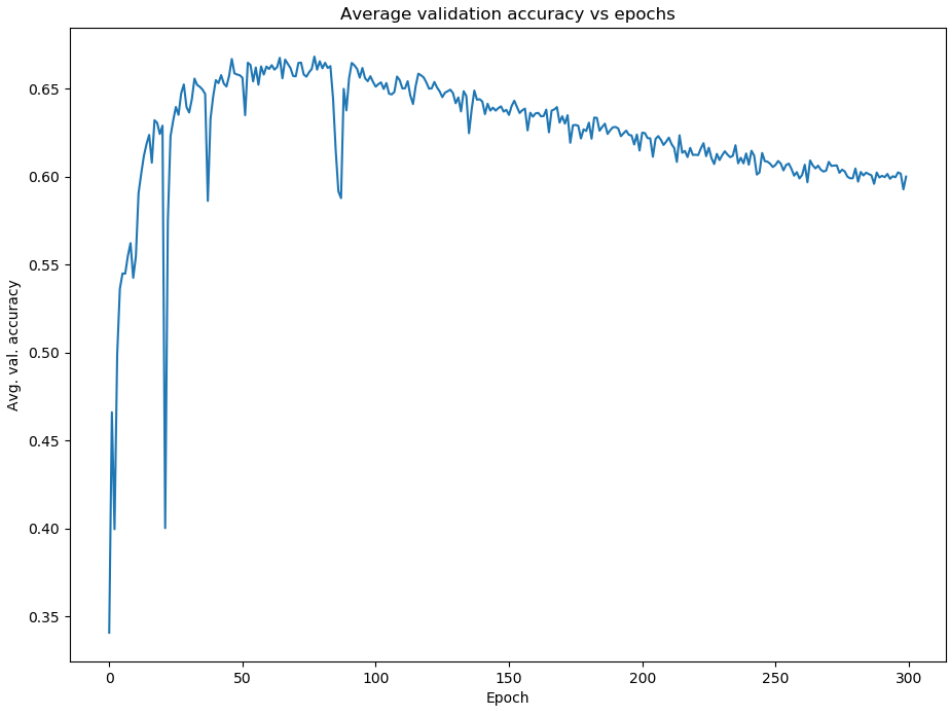


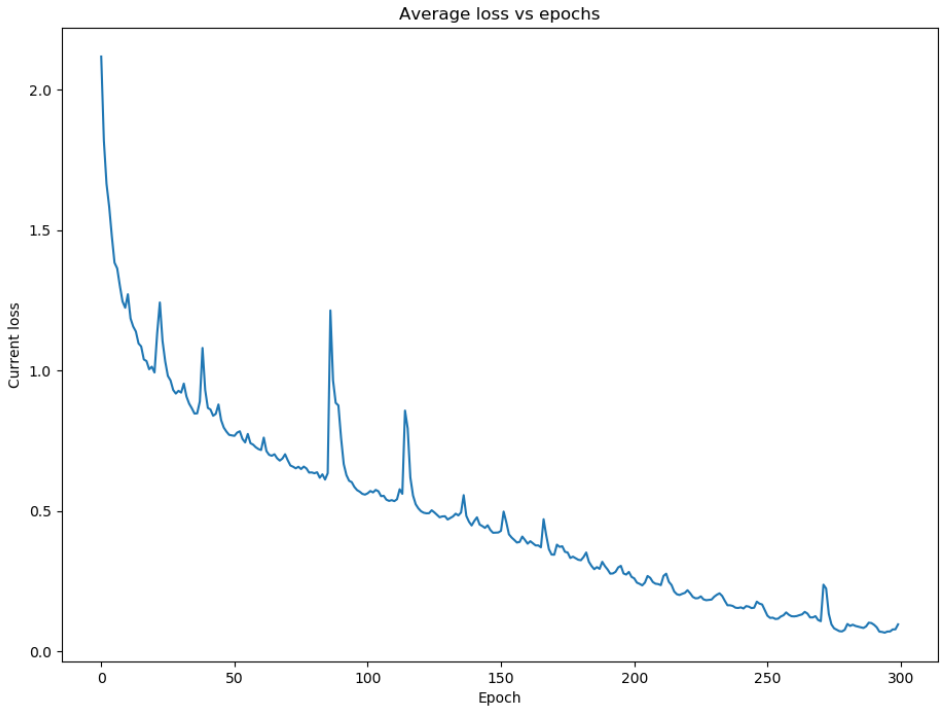
SUI_768_UNet_A



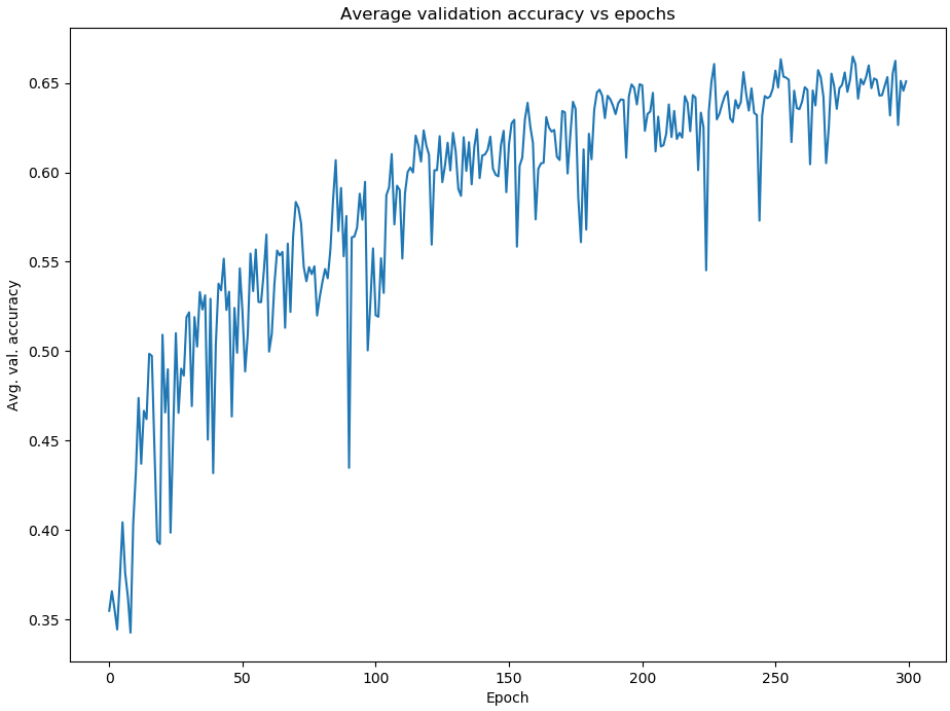


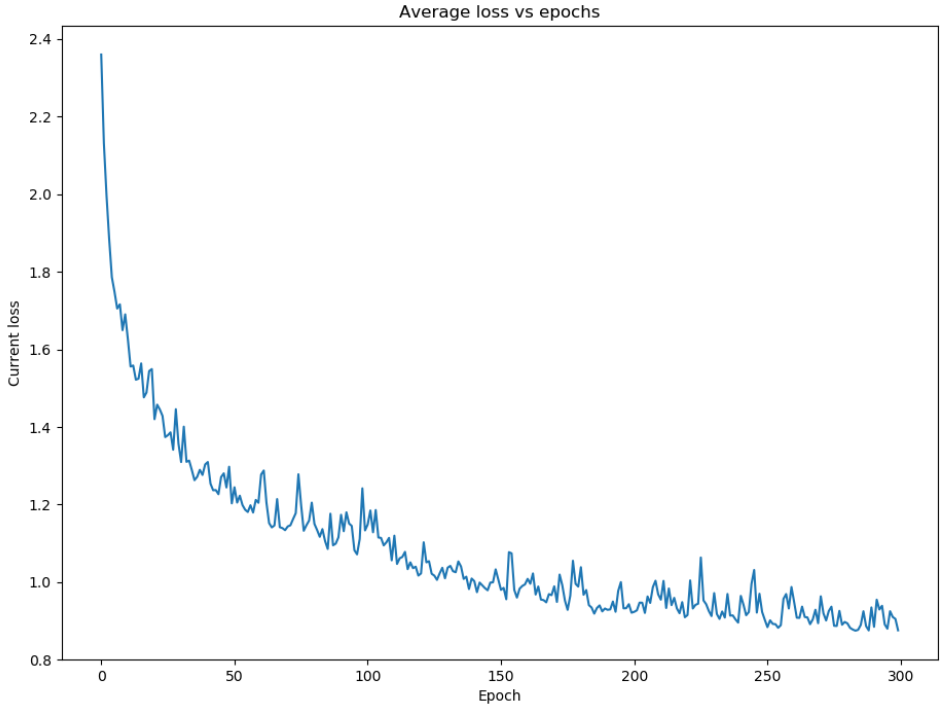
SUI_768_Seg_NA





SUI_768_Seg_A


Average validation accuracy vs epochs
Avg. val. accuracy
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0
50
100
150
200
250
300
Epoch


Average loss vs epochs
Current loss
2.4
2.2
2.0
1.8
1.6
1.4
1.2
1.0
0.8
0
50
100
150
200
250
300
Epoch