Master Thesis

Semantic Segmentation on Landslide Containment Devices

Selen Akkaya

Supervised by
Bartolomeo Montrucchio
Co-Supervisors:
Stefano Mosca
Lisa Graziani

Master's Degree in Computer Engineering

Department of Control and Computer Engineering
Politecnico di Torino

2022/2023

Semantic Segmentation for Landslide Containment Devices

Selen Akkaya

Relatori:

Supervisor
Bartolomeo Montrucchio
Co-Supervisors
Stefano Mosca
Lisa Graziani

Abstract

This thesis analyzes the application of semantic segmentation models with different approaches and aims to do a model selection and find the best optimal model with a grid search. The research focuses applying semantic segmentation on main component of landslide containment devices that are mesh and wire. Then it gives brief introduction to researches that are done so far related to semantic segmentation and explains Neural networks and how deep learning models are implemented for semantic segmentation problems. Then the data set is analyzed and exploited, as well as the necessary pre-processing steps to prepare the data for model training. Pre-processing phase includes data annotation, creating different data-sets with and without data augmentation techniques employed and data splitting for generating different separate data for different purposes such that training, validation and testing. Various model structures are explored, along with the corresponding metrics and loss functions used to refine the models for comparison. The performance of the final models on test sets is evaluated, and their overall accuracy is determined. In this thesis, binary and multi-class semantic segmentation models are explored and, U-Net and U-Net++ architectures are compared. Additionally, a grid search is conducted to find the best suitable parameters for the employed models, different combination of hyper-parameters are evaluated. The research provides insights into the use of semantic segmentation for landslide containment devices, and highlights the importance of appropriate data pre-processing and model selection for achieving accurate and reliable results. Model performance is evaluated on test sets and reported as well as the prediction images are demonstrated with the original image and ground truth image for better visualize the performance of the model.

Acknowledgements

I must begin by thanking my research mentors, Professor Bartolomeo Montrucchio, my supervisors Stefano Mosca and Lisa Graziani and all the Modelway S.r.l team. I am grateful to my colleagues for welcoming me on board and supporting me during this period. Without their assistance and diligent gaudiness in each step of my thesis, this research would not have been completed. I would like to express my gratitude to you for your gaudiness over the last six months. Most importantly, none of this would have been possible without the support of my family and my friends. I express my sincere gratitude to them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Identifying the flexible slope stabilisation system components is an essential work for predictive maintenance of these systems in order to prevent vital landslide hazards. The landslide containment components identification is typically done by humans. However, this type of detection can be done by advanced autonomous systems by taking images as input to feed the system to detect and identify the objects. Used with deep learning models, these systems are suitable for image analysis tasks such as semantic segmentation models capable of pixel-level component detection.

Semantic segmentation algorithms help to understand the context of an environment and are therefore commonly used when context is required. Semantic segmentation is an important area in Computer Vision and the purpose is labelling each pixel in an image as a defined class. Therefore, this approach allows image classification at the pixel level. Assigned tags may represent objects (eg: rock, road, house) and/or living entities (eg: people, animals) in an image.

The objective of this thesis is to evaluate the performance of state-of-the-art semantic segmentation models based to detect mesh and wire in landslide containment devices. According to the results, the most reliable model will be selected for image analysis. Different combinations of model parameters will be searched with a grid search to find an optimum model. Conclusively, the models will be valuable for the maintenance of landslide containment devices. This project is part of a bigger project that is held by Modelway S.r.l. and the company is assisted in the creation of this thesis and the final result will be an optimal semantic segmentation model suitable for image analysis.

The consideration of this thesis is the landslide containment devices that are composed of two types of spikes, mesh and wire. This thesis focuses on detecting two main components, which are customarily referred to as "mesh" and "wire".

The Flexible Slope Stabilization System

Mass movements such as landslides are very important natural hazards that threaten natural resources, human life, infrastructures and property in mountainous regions around the world[8].
Landslides are defined as the movement of a mass of debris, rocks, or slope failures, which occurs during rainfall, runoff, rapid snow-melt, earthquakes, and volcanic eruptions[7].
Landslides occur for various reasons such as earthquake shocks, heavy rainfall, or road construction in hilly areas[9].

The flexible systems used in slope stabilization consist of a durable cover (cable netting or high-strength wire mesh) and anchor bolts. This technique has become widespread as it is visually inconspicuous and has minimal impact on traffic during installation[3]. To enhance the understanding of structure of landslide containment devices, it is beneficial to examine their various components. The Flexible Slope Stabilization Systems consist of several primary components, and detecting some of these components could be crucial for maintaining landslide containment systems.

- Wire mesh: triple torsion fabricated with low-resistance steel. for purpose of reducing the net grid spacing to prevent detachment of small fragments of soil or rock,
- Steel cables : the cables are fixed at the intersection points of the net weave by staples,
- Reinforcement/sewing and perimeter cables: employed to join net panels, fit the net to the ground and make the system rigid through connection with the central bolts and anchors of the perimeter cable.
- Bolts: arranged in rows and columns with a constant separation,
- Cable anchors:used at the edge of the zone to be stabilised to brace and tense the perimeter cables,
- Spike plate: to attach the intersection of the net cables and reinforcement cables to the ground by a nut thread in the bolt[4],

The Figure 1.1 demonstrates two distinct types of 'Spike plates'. The Figure 1.2 demonstrates composition of 'wire mesh', 'Steel cables', and a particular type of 'spike plate' with ring shape. Lastly, The Figure 1.3 demonstrates this composition from a broad perspective.

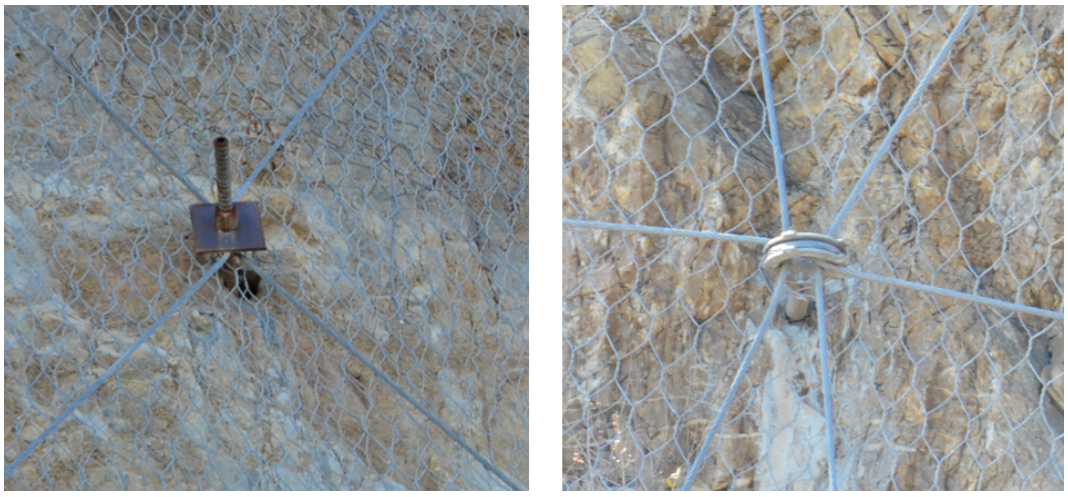


Figure 1.1: Different types of spike plates from our data-set.

Figure 1.2: Composition of the components of Flexible Slope Stabilization System

Figure 1.3: Composition from a broad perspective.

Chapter 2

Background

This chapter explains the background information on the development of semantic segmentation and Neural Networks, the technical details of deep learning and its application in semantic segmentation tasks. It covers Convolutional Neural Networks (CNNs) and their variations, particularly, the cutting-edge models such as SegNet, U-Net, U-Net++ used for Semantic Segmentation. In addition, metrics such as Dice coefficient, Intersection over Union and pixel accuracy, which are widely used for semantic segmentation, and loss functions are explained.

2.1 Artificial Intelligence

Basic knowledge of artificial intelligence (AI) and in particular machine learning and deep learning is essential to understand and successfully perform a pixel-level component recognition task.

Artificial intelligence (AI) is a branch of computer science that aims to create systems that perform tasks normally carried out by humans, such as pattern recognition, understanding of natural languages, making decisions etc. Variety of techniques are utilized including machine learning and deep learning, developing systems that perform tasks previously thought to be the exclusive domain of humans[6].

2.2 Machine Learning

Machine Learning (ML) is a branch of Artificial Intelligence (AI) that concentrates on creating algorithms and statistical models that allow machines to learn from data without human intervention. These algorithms allow systems to automatically improve their performance by detecting patterns and connections in the data, making predictions or decisions based on that knowledge. Machine Learning algorithms can be divided into three main categories: supervised, unsupervised, and reinforcement learning[6].

In supervised learning, a model is trained using labelled data. The goal of the training process is learning a mapping from inputs to outputs so that when the model receives new, unknown inputs, it can predict the correct output. Supervised learning is typically used in image classification. On the other hand, in unsupervised learning, a dataset of unlabeled examples is given to the model and the goal is learning some basic structure or distribution from the data. Unsupervised learning is widely used

in areas such as computer vision where a lot of unlabeled data is available.

2.3 Deep Learning

Deep Learning is a technique within Machine Learning (ML) that employs deep neural networks (DNNs) with multiple layers to learn representations of data, particularly in computer vision applications. These DNNs are able to learn increasingly complex representations of the input data by training on large amounts of labeled data, and they have been able to achieve state-of-the-art performance on a wide range of computer vision tasks such as image classification, object detection and semantic segmentation[6].

2.4 Neural Networks

Neural Networks are machine learning model based on the structure and function of the human brain. It consists of interconnected processing units known as artificial neurons arranged in layers. These layers process the input data and produce output through a series of mathematical operations. The neurons in the neural network are trained using a large set of labeled data, and the goal is to learn the basic relationships between input and output. An example of layer representation is shown in Figure 2.1. Neural networks can be used for a wide variety of tasks such as image classification, natural language processing speech recognition etc. The architecture of neural networks can be divided into feed forward, convolutional and recurrent neural networks, each serving a specific purpose and suitable for different types of tasks[6].

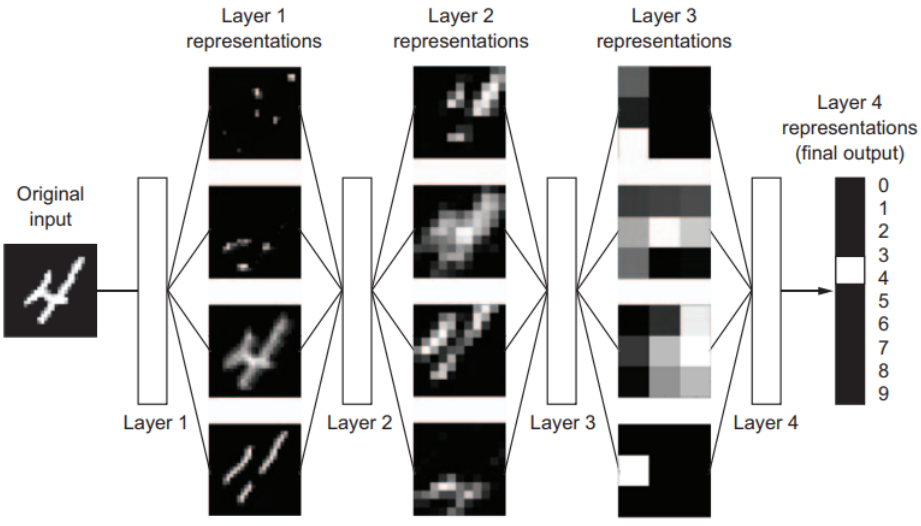


Figure 2.1: Example of deep representations that is learned by a digit classification model[6].

Layers in a neural network can be broadly classified into three types as input layers, hidden layers and output layer. The input layers are the first layers of the network and the main functionality is receiving the input data and forwarding

them to the next layer. The upcoming layers are called hidden layers and their main functionality is extracting features from the input data. These layers typically consist of several artificial neurons that are connected to each other. They are responsible for the non-linear transformation of the input data. Moreover, these layers are called hidden layers because they are not directly involved with the input or output of the data. The final layer in a network is output layer and their main task is producing the final output. This layer takes the extracted features from the hidden layers and uses these features to make a decision or prediction. The number of layers and neurons in each layer can be varied and this custom variation is depending on the task and data complexity. The more layers and neurons there are in a model, the more complex it is, and the more data must be used for training. During the training phase, the network regulates these parameters for minimizing the error that occurred between the predicted output and the ground truth. This is referred to as back-propagation. The training process is being iterated over and over again so that the network learns from the data and improves its state[6].

A representation of a neural network is shown in the Figure 2.2 that is defined by its weights, and evaluation of a loss metric which serves as a feedback signal to update the network's weights

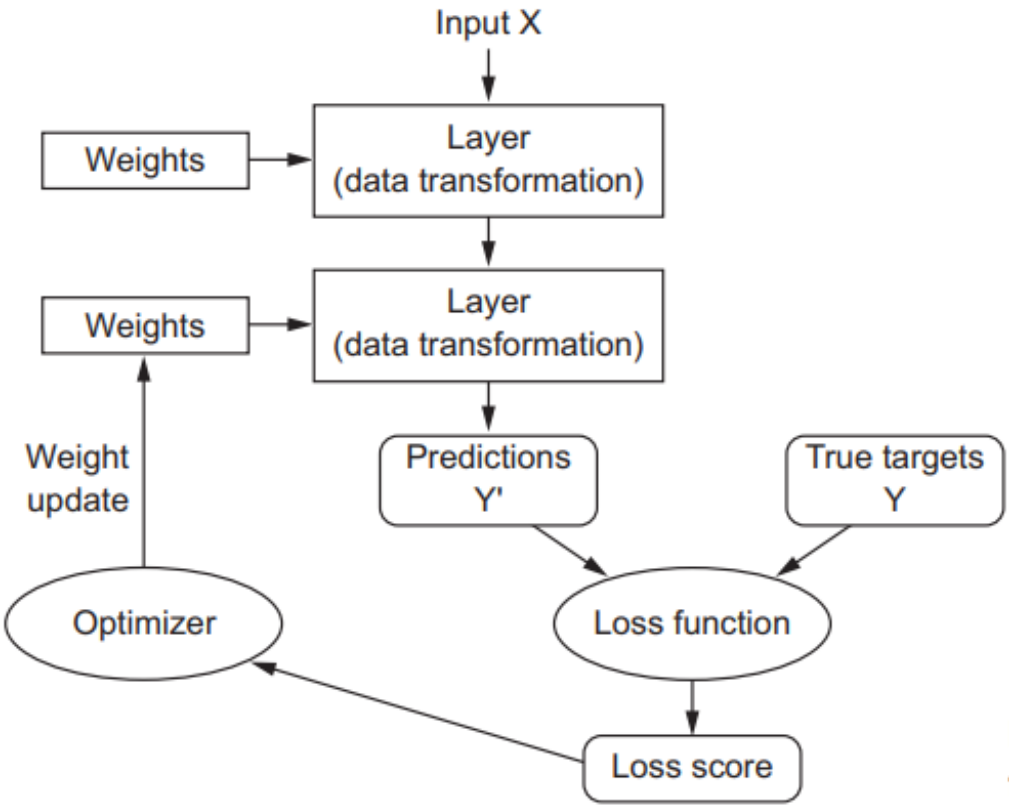


Figure 2.2: An illustration of a neural network that is parameterized with its weights and loss score calculation in order to use as a feedback signal to regulate the weights[6].

A neural network is parameterized by its weights which are used for calculations and generating the predictions. The weights are essentially the values assigned to the connections between the neurons of a network. Weights are adjusted during the

process of training in order to optimize the performance of the network. The error evaluation between predicted output and the ground truth is referred as the loss score that is used as a feedback mechanism to fine-tune the weights. The loss score is a network measure that indicates how well the network is performing. The goal of the training is to minimize the loss score which means that the network predictions are as close as possible to the ground truth. The most common loss functions used in neural networks are Cross Entropy Loss and Mean Squared Error. The training process initiates by feeding the network with input data. Then the weights of the network are used to make calculations and thus generate predictions. The estimated output (prediction) is then compared with the actual output (ground truth) by a loss function, this comparison provides a loss score. The loss score is then used as a feedback signal and the weights are adjusted, in a way to minimize the loss score, by using an optimization algorithm (such as Stochastic Gradient Descent (SGD) or Adam). This process is being iterated until the network converges in other words reaches a sufficient level of accuracy[6].

2.4.1 The History of Neural Networks

The neural networks history is traced back to the 1940s-1950s when researchers initially considered creating artificial networks of simple elements, or "neurons," to mimic the behavior of biological neurons. Warren McCulloch and Walter Pitts proposed the first artificial neural network concept in 1943, proposing a simple mathematical model of a biological neuron that could be used to execute simple logical functions[16].

In the 1950s and 1960s, researchers such as Frank Rosenblatt and Bernard Widrow continued to develop early neural network models, known as perceptrons [18]. These models were based on a single layer of artificial neurons and were able to perform simple linear classification tasks. However, they were not able to solve more complex problems due to the limitations of their architecture.

In the 1980s, David Rumelhart introduced the concept of backpropagation, an algorithm for training multi-layer neural networks[19].This allowed for the training of more complex networks, known as multi-layer perceptrons (MLPs), which were capable of solving more complex problems. However, the training of these networks was still difficult and computationally expensive, which limited their practical application.

In 1998, Yann LeCun published a detailed paper on convolutional neural networks, graph transformer networks, and a discriminative training method for sequence labeling, finalizing the training algorithm[14]. However, it wasn not until 2012, with the advancements in optimized GPUs, that convolutional neural networks saw a significant breakthrough. In that year, a team led by Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, won the ImageNet Large Scale Visual Recognition Challenge with their model, AlexNet [13]. The model used a convolutional neural network and achieved a Top-5 error rate of 15.3 %, significantly outperforming the next best result of 26.2%. Ever since the breakthrough in 2012, convolutional neural networks (CNN) have consistently been the leading method in many computer vision tasks and have yet to be surpassed by other techniques. In the last decade, with the availability of large data-sets and powerful computational resources, neural networks have experienced a resurgence of interest and have been used to achieve

state-of-the-art performance in a wide range of applications such as computer vision, natural language processing, speech recognition and generation, among others.

In the course of the history of neural networks, complex and powerful models has been developed as well as new training algorithms. Therefore, wide variety of real-world problems became feasible to solve by neural networks.

2.4.2 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a sort of neural network that excels at analyzing image and video tasks. CNNs were inspired by animal vision systems and are meant to handle data having a grid-like layout, such as images. CNNs are composed of various layers, these layers are convolutional layers, pooling layers, and fully connected layers. Convolutional layers are the core of CNNs that perform convolution operations on the input data. They are used to extract features from the input data, then these features are passed on to the next layers in the network. Pooling layers are used to reduce the spatial size of the data with the given strategy. It essentially allows the network to focus on the important features while reducing the dimensionality of the data. Fully connected layers (also known as dense layers) refers to the layers whose inside neurons connect to every neuron in the preceding layer. They are used to make predictions (or decisions) based on the extracted features of the convolutional and pooling layers. CNNs can learn hierarchical visual representations. The characteristics learnt may be used to a variety of image analysis tasks, including picture classification, object recognition, and semantic segmentation. CNNs have been utilized to achieve cutting-edge performance in a variety of computer vision applications.

Convolution Operation

Convolution is a mathematical procedure that CNNs employ to extract features from incoming data. Convolution is done between input data, such as an image, and a kernel which is a small matrix and is also known as a filter. The input data is scanned by the filter (kernel). An element-wise multiplication between the filter values and the corresponding input values at each position is performed. After this element-wise multiplication, a summation operation is performed to generate a single output value for that position. This process is repeated for each position of the filter across the input data resulting in a new output matrix called activation map or feature map. A feature map or activation map is a multi-dimensional array that contains the convolution operation output for each position of the filter. A representation of convolution operation is simply shown in the Figure 2.3. Each element in the feature map represents the filter activation or response to a specific region in the input data. The feature maps are a simplified representation of the input data that emphasizes specific features such as edges, patterns etc.

Multiple features can be extracted from the input data by using multiple filters, each filter responding to different patterns in the input data, resulting in multiple feature maps. These feature maps are then passed through additional CNN layers such as pooling layers and activation layers to extract more abstract features and reduce data dimensionality.

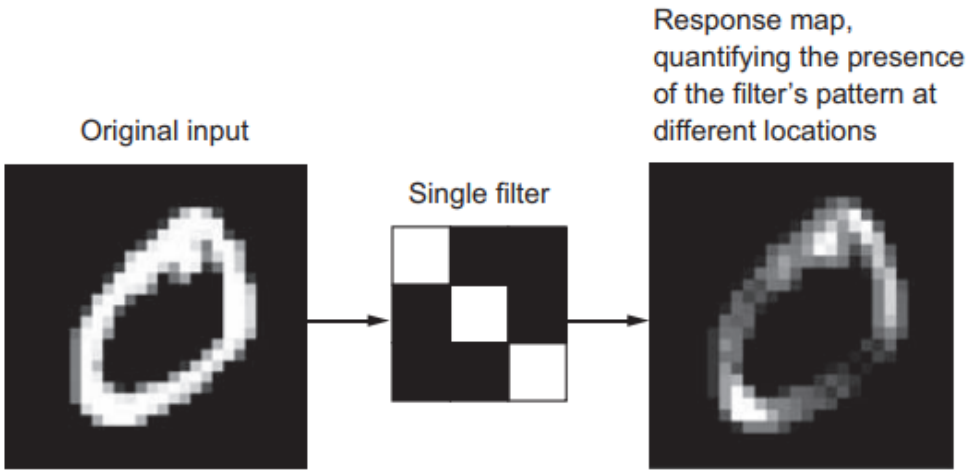


Figure 2.3: An illustration of convolution operation[6]

Convolution Layers

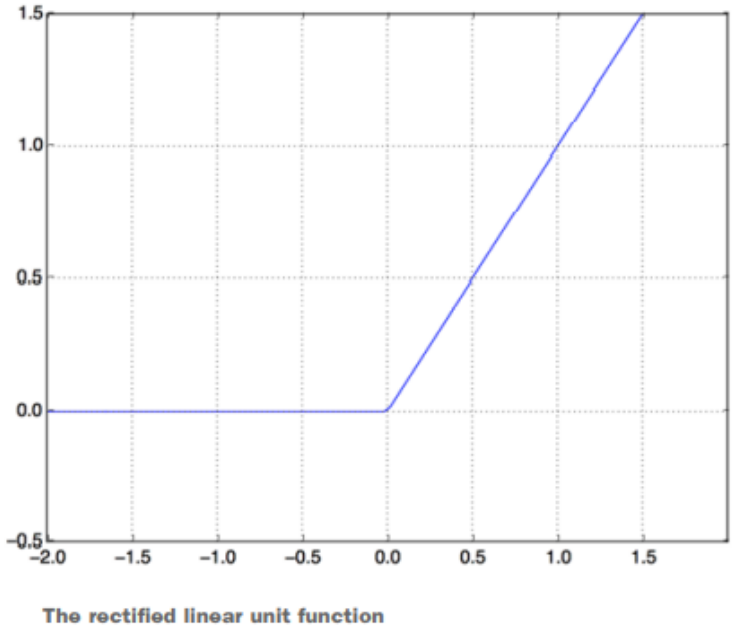
Convolutional layers are the basic building block of Convolutional Neural Networks and are used to extract features from input data, which are then passed through other layers in the network to make classifications or predictions.

The input data is passed through a series of filters, each of which is a small matrix, in convolutional layers. The input data is scanned by these filters and performing an element-wise multiplication with the corresponding input values in each position. This process is repeated for each filter position across the input data, after all, producing a new output matrix known as a feature map. The important point is that each neuron in the feature map is connected to only a small and localized region of input data that is defined by the size and stride of the kernel. This local connectivity allows the network to extract patterns or features from the input data. This structure is called as translation-invariant: regardless of their position in the input data, the features will be detected. This property is crucial in order to obtain a good performance of CNNs in tasks such as image recognition, where the objects of interest may appear in different positions in a given input image.

Non-linear Layer and Activation Functions

A nonlinear layer consists of an activation function that takes the feature map generated by the convolutional layer and generates the activation map by applying the activation function to each element-wise value of the feature map. This layer allows the network to model complex, nonlinear data relationships. It is worth to say that rectified linear unit (ReLU) function is the most widely employed non-linear function and this function converts any negative input values to 0 while leaving positive values unchanged, introducing non-linearity into the network. Since linear functions cannot model non-linear relationships, this non-linearity is critical for the network to learn complex representations of the data. Typically, Relu is typically used as activation function in the intermediate layers, and sigmoid is typically used activation function in the final layer for binary outputs that outputs a probability (a score between 0 and 1). Representation of ReLu and Sigmoid functions are shown

in the Figure. Additionally, softmax is another popular activation function that is used in final layer in multi-class classifications.



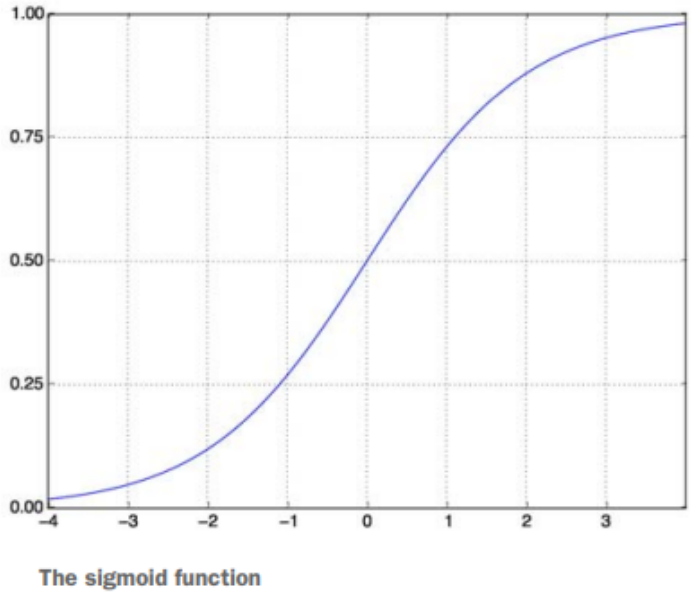


Figure 2.4: ReLu and Sigmoid functions respectively[6].

Pooling Layers

A pooling layer is a type of layer in Neural Networks which performs down-sampling operations on the incoming data while reducing spatial dimension with the chosen

strategy such as maximum pooling and average pooling. Max pooling selects the maximum value in a small window of the input data and outputs it as the new value for that position. In average pooling, the average value of the input data in a small window is calculated and the result outputs as the new value for that position. A pooling layer is typically applied after one or more convolutional layers, and the pooling operation is typically performed on the convolutional layer's output feature maps. The pooling operation reduces the size of the feature maps, resulting in a more computationally efficient network. The primary purpose of the pooling layers is utilizing the network more resistant to small translations in the input. Pooling layers are used to extract the most important features from input data and make the network more robust to minor translations of the objects in the input data.

Dense Layers

A dense layer (also known as a fully connected layer) are connected to every neuron in the preceding layer instead of connecting to just a specific subset of neurons as seen in other architectures such as convolutional neural networks. Each neuron receives incoming data from all the neurons in the previous layer and applies a set of parameters (weights and biases) to the input data to produce an output. Then the output is sent to the next layer. Dense layers are commonly used in feed-forward neural networks and are employed for extracting features from the input data. Dense layers are typically positioned between the input and output layers to extract features from the input data. They can be stacked one on top of the other to enable the extraction of high-level features from the input data.

2.4.3 Encoder Decoder Structure

Encoder-decoder structure is a type of neural network architecture that is typically used in computer vision, particularly in image and video analysis tasks.

The encoder blocks of the network extracts features from the input image (or video). On the other hand, the decoder blocks of the network uses these features to generate the output image (or video). This architecture is important since it enables the network to learn a compact representation of input data, which can then be used to produce more accurate output. Additionally, It is also used in tasks like image captioning, semantic segmentation, and object detection.

Encoder

An encoder architecture usually consists of multiple layers of convolutional neural networks (CNNs). These layers extract increasingly complex features from the data as the input image or video passes through them. Convolutional layers, pooling layers, and normalization layers are common CNN layers in encoders. By applying a set of filters to the input data, convolutional layers start extracting local features from images, pooling layers are used to reduce the spatial dimensions of feature maps while keeping the most important information intact. Normalization layers, optionally, are used to ensure that the input data is consistent and to mitigate the impact of any outliers. The final output of an encoder is a compact, fixed-length feature vector representing the input image (or video).

Decoder

Decoder architecture typically is consists of multiple decoder blocks. Each decoder block receives the features from the encoder as fixed-length, compact feature vector that os used to generate the output image (or video) by applying a set of filters to the input feature vector. Transposed convolutional layers, also known as up-convolutional layers, are used to increase the spatial dimensions of feature maps. These layers are the inverse of the encoder's convolutional layers and are used to generate a higher-resolution output image (or video).

2.5 Semantic Segmentation

Semantic segmentation is a Computer Vision task that essentially is based on interpretation and understanding the visual information from the world. It entails developing algorithms and methods for assessing photos and videos, and interpreting the scene. The objective of computer vision is building machines that can perceive and understand the visual world like humans and apply this understanding to a wide range of applications, such as autonomous cars, medical imaging, security systems etc.[12].

Semantic Segmentation is the process of assigning a semantic label, such as "road", "sky", or "car", to each pixel in an image and creating a dense prediction that classifies every pixel in the image and provides a complete representation of its semantic content. Semantic Segmentation tasks differs from other tasks of computer vision, such as Image Classification, Object Detection, Instance Segmentation so on. More specifically, image classification only assigns a single label to the whole image, whereas Object Detection involves not only classification but also the localization of objects within an image and the drawing of bounding boxes around them. Instance Segmentation takes it a step further, as it not only detects objects but also separates individual instances of the same object class. In contrast, the main focus of Semantic Segmentation is to label each pixel in the image with its semantic class, producing a dense label map that provides a more fine-grained representation of the image with respect to the sparse representation of bounding boxes in Object Detection. Moreover, semantic segmentation task is useful for several applications such as image editing, and scene understanding. Finally, it's worth noting that semantic segmentation is a very active research area and new techniques and architectures are being proposed frequently. Different approaches to the semantic segmentation problem.

Binary semantic segmentation aims classifying each pixel in an image as belonging to one of two classes. For instance, in an image of a road scene, the classes could be "road" and "not road". The model would output an image where each pixel is labeled as either one or other.

Multi-class semantic segmentation aims to classify each pixel in an image as belonging to one of multiple classes which provides to do pixel-wise labelling of multiple objects by using only one model. For example, in an image of a road scene, the classes could be "road", "vehicle", "pedestrian", and "sky". The model would output an image where each pixel is labeled with one of the classes. Multi-class semantic segmentation is relatively more challenging than binary semantic segmentation because it requires the model to distinguish among different classes, whereas

binary semantic segmentation only requires the model to distinguish between two classes. Selecting the one of these two strategies are depend on the need of the specific problem.

2.5.1 Comparisons of different computer vision tasks

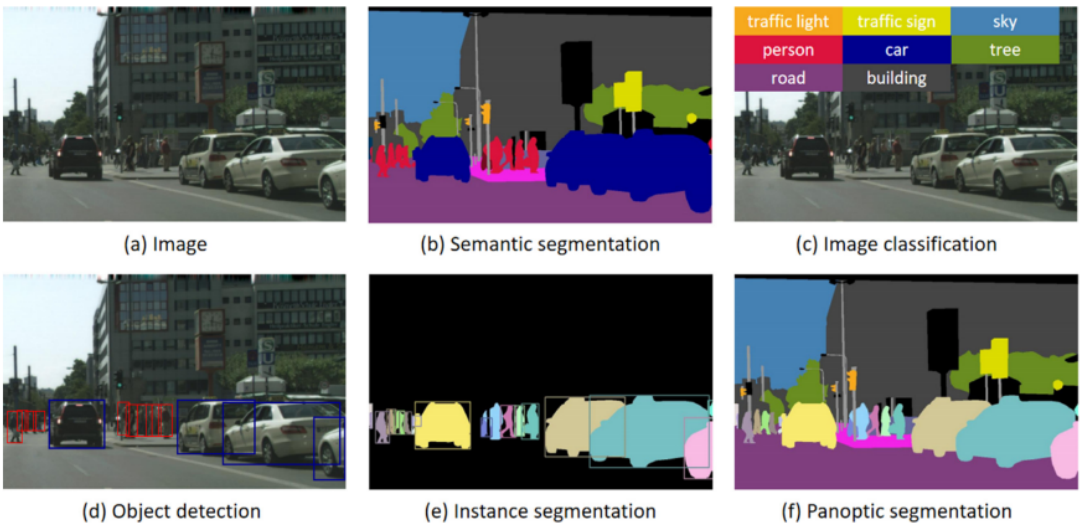


Figure 2.5: An illustration of several computer vision tasks [11] [10]

The goal of image classification is to assign a whole image to one or more category labels. In other words, an algorithm for image classification identifies the objects in a picture. The objective of Object Detection is to identify objects in an image and determine their location. This computer vision task involves both object classification and localization. The system outputs a collection of bounding boxes that surround the objects found in the image and provide their corresponding class labels that is shown in the Figure 2.5 (d). Semantic segmentation, on the other hand, has higher requirements because it aims to precisely separate each object region from the background region.

An example is given in Figure 2.5, showing that a segmentation algorithm is able to not only find all the desired objects, but also accurately define their boundaries. It is mentioned that semantic segmentation is more difficult than the previous tasks because it involves connecting low-level features with high-level meaning. Two advanced tasks, instance segmentation and panoptic segmentation, have become the focus of new research. Instance segmentation aims to identify each individual object within an image, as shown in Figure 2.5 (e) where cars are labeled with different labels representing each instance. Panoptic segmentation goes one step further by assigning both a semantic label and an instance label to every pixel. These tasks are more demanding than traditional semantic segmentation as they focus on identifying specific instances of objects and not just general class labels[10].

2.5.2 Traditional Computer Vision-based Methods for Semantic Segmentation

Before the advent of deep neural networks, semantic segmentation was primarily approached using traditional computer vision techniques such as graph-based methods and region-based methods.

Graph-based methods involve representing the image as a graph, with each pixel or super-pixel as a node and edges between neighboring pixels or super-pixels. The purpose is to divide the graph into segments (or regions), each of which corresponds to a particular object or area of the picture. Region-based methods, on the other hand, involve grouping pixels or super-pixels that are similar in terms of color, texture, or other features. Edge-based image segmentation divides an image into areas or objects by recognizing edges or boundaries in the picture. This method is based on the assumption that picture edges or borders represent the transition between various regions or objects. They serve as the foundation for dividing a picture into separate sections or items. Pixel-based image segmentation divides a picture into various areas or objects by evaluating the attributes of each pixel and it assumes that pixels of image that have similar color, texture etc. correspond to the same region or object[2].

These methods are less precise than deep learning-based solutions and are easily influenced by changes in lighting, perspective, and occlusions, among other difficulties.

Figure 2.6 illustrates the application of the mentioned methods to a sample image. In addition, these traditional methods are not able to handle large variations in object shape, size, pose and viewpoint, which makes them less robust.

Overall, traditional computer vision-based methods for semantic segmentation were less accurate and less robust than deep learning-based methods, and they were not able to handle large variations in object shape, size, pose and viewpoint.

2.5.3 Deep Learning-based Methods for Semantic Segmentation

Deep learning-based methods for semantic segmentation have become the state-of-the-art in recent years that are based on convolutional neural networks (CNNs). Most popular deep learning-based application for semantic segmentation, including: Fully convolutional networks (FCNs) are a sort of CNN that can accept any size input picture and create an output segmentation map of the same size. FCNs may leverage the spatial information in an image by extracting features at different scales using a sequence of convolutional and pooling layers.

U-Net is a form of FCN that can accurately segment small structures in pictures, making it particularly well-suited for biomedical image segmentation applications. U-Net is built on an encoder-decoder architecture, with the encoder extracting features from the picture and the decoder reconstructing the segmentation map from the features.

These approaches outperformed various benchmarks and have been employed in

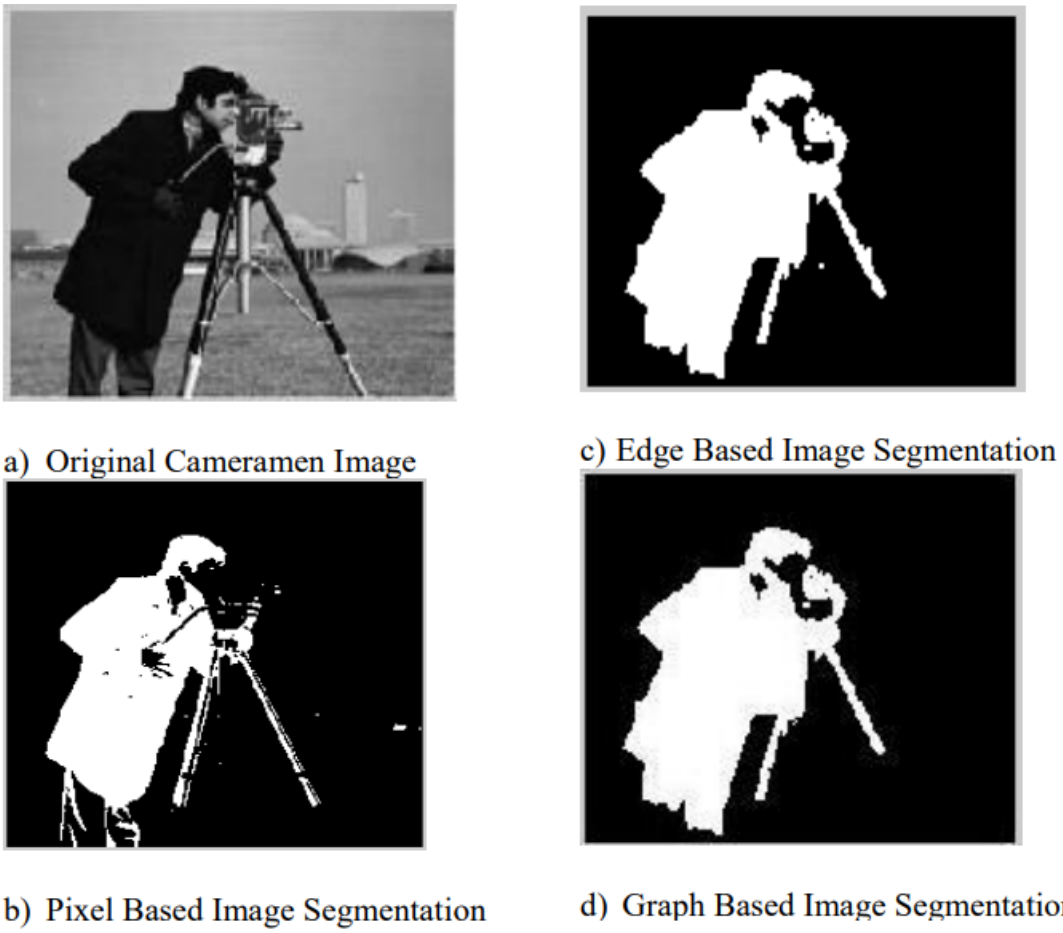


Figure 2.6: Representation of pixel-based, edge-based, graph-based image segmentation [2]

a variety of applications, including self-driving automobiles, robot navigation, picture editing, medical imaging, and scene interpretation, among others. It's worth mentioning that these systems are always evolving and improving, with new methodologies and architectures being offered on a regular basis.

2.5.4 FCN for Semantic Segmentation

Fully Convolutional Networks (FCNs), a version of CNNs, were a significant advancement in the area of image segmentation. The Fully Convolutional Network (FCN) for Semantic Segmentation was the first model that adapted and fine-tuned classification networks for direct, dense prediction of semantic segmentation[15]. A Fully Convolutional Neural Network(FCN) is represented in Figure 2.7.

A Fully Convolutional Network (FCN) is a sort of convolutional neural network (CNN) that is intended to make dense predictions across a whole picture rather than identifying a particular patch or portion of an image. This is accomplished by replacing the fully connected layers at the end of a CNN with convolutional layers, which enables the network to accept inputs of any size and provide output with commensurate spatial dimensions. In other words, a Fully Convolutional Network makes dense predictions over an image while processing the entire image at once, rather than processing small regions of the image at a time. This allows the net-

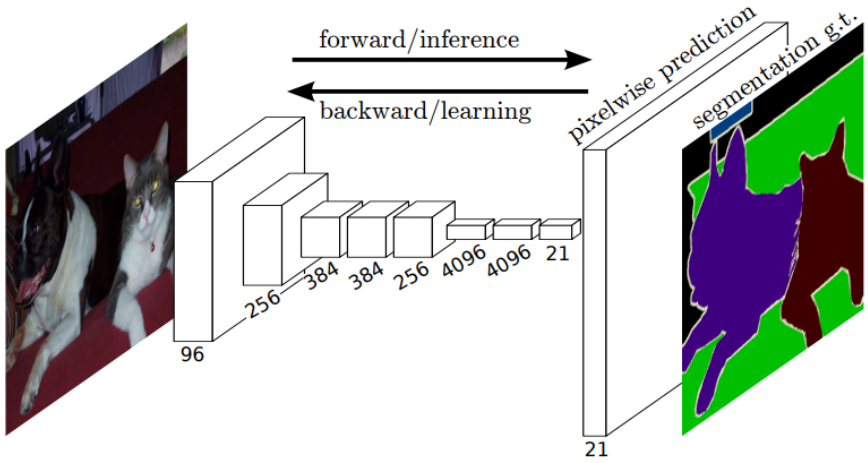


Figure 2.7: A Fully Convolutional Neural Network (FCN) make dense predictions for per-pixel tasks, allowing the network to take in inputs of arbitrary size and produce output with corresponding spatial dimensions[15].

work to consider the context of the entire image when making predictions, which is beneficial for tasks such as object detection and semantic segmentation.

In traditional classification networks, an input image is downsized and processed through convolution layers and fully connected layers to produce a predicted label. But by replacing the fully connected layers with 1x1 convolutional layers and not downsize the image, the output becomes a feature map with a smaller resolution than the input image, due to max pooling. By upsampling the output to the original resolution, a pixel-wise label map is obtained.

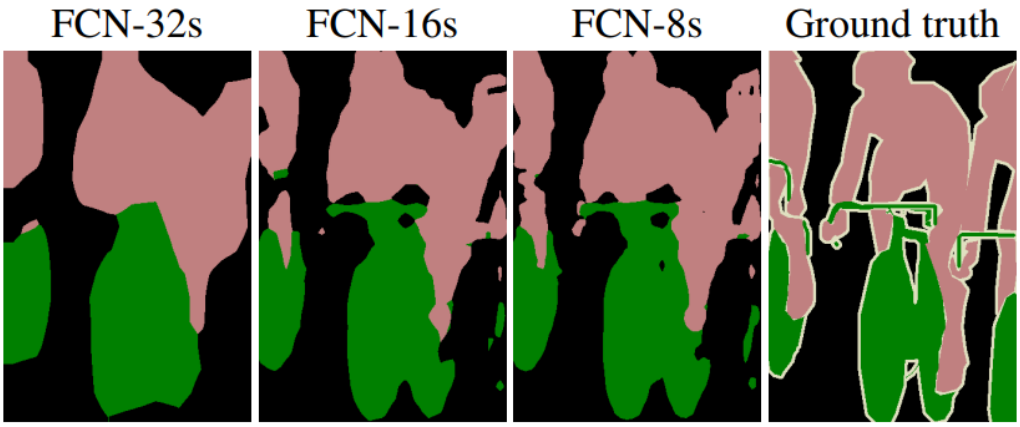


Figure 2.8: Combining information from layers with different strides in a fully convolutional network and observation of enhanced accuracy of segmentation[15].

In addition, a Fully Convolutional Network (FCN) has the effect of strides on the convolutional layers in terms of reducing the spatial resolution of the feature maps. This is done by skipping over some of the elements in the input feature maps during

the convolution operation. Larger strides result in a larger reduction of spatial resolution, while smaller strides result in a smaller reduction. A representation is shown in Figure 2.8. This can be useful for reducing the computational cost of the network and preventing overfitting, but it also means that the network may lose some fine-grained information about the input.

2.5.5 SegNet

SegNet, developed by the University of Cambridge, [1] was initially submitted to the 2015 CVPR conference but was not officially published until 2017 in the TPAMI journal.

SegNet is a convolutional neural network architecture designed for semantic image segmentation tasks. It is an encoder-decoder type of architecture where the encoder is used to extract features from the input image and the decoder is used to make predictions based on these features.

It consists of an encoder network and a corresponding decoder network, followed by a final pixel-wise classification layer as shown in Figure 2.9. The encoder is made up of 13 convolutional layers, which are based on the VGG16 network[20]. VGG16 network is a pre-trained network that has been trained on a large dataset of images. The encoder extracts features from the input image and compresses them into a lower-dimensional feature space. The decoder is an up-sampling network that expands the compressed features back to the original resolution of the input image. Each layer in the encoder has a corresponding layer in the decoder, resulting in a total of 13 layers in the decoder network and the final output from the decoder is then fed into a multi-class soft-max classifier to produce class probabilities for each pixel independently.

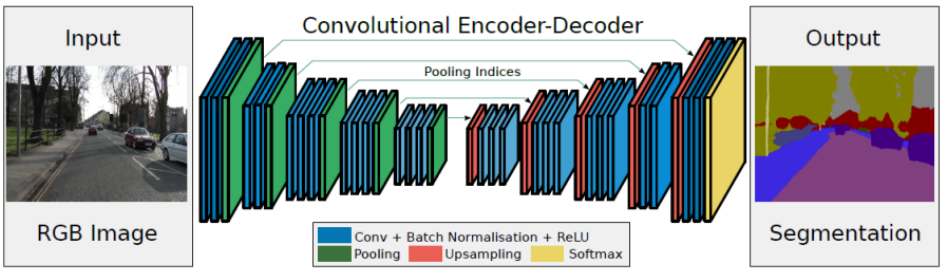


Figure 2.9: Illustration of SegNet architecture [1]

The encoder network in SegNet uses a filter bank to produce a set of feature maps. These feature maps are batch-normalized and then an element-wise rectified linear non-linearity (ReLU) is applied. After that, max-pooling with a 2x2 window and a stride of 2 (non-overlapping window) is performed to reduce the resolution of the feature maps by a factor of 2. This is done to achieve translation invariance over small spatial shifts in the input image. However, as a result, the spatial resolution of the feature maps is reduced. This is not ideal for segmentation, where boundary delineation is important. To address this, the authors of SegNet capture and save boundary information in the encoder feature maps before subsampling is performed.

However, due to memory limitations, the stored information was reduced to only max-pooling indices, i.e., the locations of the maximum feature value in each pooling window is recorded for each encoder feature map.

The decoder network is in charge of upsampling the encoder network feature maps to the original picture resolution and constructing the final segmentation mask.

SegNet decoder network employs a reverse design of the encoder network, with pooling indices from the corresponding encoder layer utilized to accomplish non-linear up-sampling of feature maps. This permits the decoder network to preserve spatial information that was lost during the encoder network's down-sampling operation. During upsampling, the feature maps are processed through a succession of convolutional and non-linear activation layers to generate the final segmentation mask. SegNet decoder network is meant to reconstruct the spatial resolution of the original image while keeping the encoder high-level semantic information of the network .

The performance of SegNet was evaluated on two scene segmentation benchmarks. The first task was road scene segmentation with CamVid dataset [5] which has practical applications in various autonomous driving-related problems. The second task was indoor scene segmentation with SUN RGB-D dataset [21] which is relevant to several augmented reality tasks. The model performance was compared to several other widely adopted deep architectures for segmentation such as FCN, DeepLab, and DeconvNet. All the architectures were trained with the same hyperparameters and output the same resolution output feature maps. According to the results of the original paper, SegNet outperforms all other architectures in all metrics (Class average, Global average, mean Intersection over Union, and Boundary F1 measure) on the CamVid dataset. The experimental results on the SUN RGB-D dataset reveal some interesting points. All the deep architectures have very low mean IoU and boundary metrics, as well as small global and class averages. In terms of mIoU, SegNet outperformed FCN and DeconvNet but had a slightly lower mIoU than DeepLab-LargeFOV, which is probably due to the CRF post-processing used in the DeepLab model. The authors attribute the overall poor performance to a large number of classes in this segmentation task, many of which occupy a small part of the image and appear infrequently.

2.5.6 U-Net

The U-Net is a modified fully convolutional network (FCN) that is mostly used in semantic segmentation tasks. Ronneberger and Fischer originally identified the U-Net in 2015, and it was especially built for image segmentation and localization in biomedical applications[17].

U-Nets have an advantage over regular CNNs because U-Net can both identify where an object is in an image and what it is. Thus, it provides both localization and classification. In this context, localization means that each pixel in an image is assigned a corresponding class label.

U-Nets have an advantage over Fully Convolutional Networks (FCNs) because they can produce accurate segmentation even with a limited number of training images. This is achieved by using up-sampling layers with a large number of fea-

ture channels, which helps conveying more contextual information to the higher resolution layers.

The U-Net segmentation network is the one of the best choice for this thesis application because it is a scalable and flexible Fully Convolutional Network (FCN) that does not compromise on localization and context information.

Network Architecture

It has a specific structure that includes a contracting path on the left and an expansive path on the right, which is essentially an encoder-decoder architecture. The contracting path follows a typical convolutional network format, including repeated use of 3x3 convolutions, ReLU activation, and 2x2 max pooling with a stride of 2. The network also contains skip connections that is in between the encoder and decoder. Skip connections allow the decoder to learn the high-level features from the encoder, and allows the network to retain spatial information (height and width) throughout the network. More specifically, skip connections contribute the decoder blocks with the intermediate encoder activations that is composed of high resolution details.

At each downsampling step, the number of feature channels is doubled. Each step in the expansive path includes an up-sampling of the feature map, followed by a 2x2 convolution that reduces the number of feature channels by half. This is combined with the correspondingly cropped feature map from the contracting path, and two 3x3 convolutions with a ReLU activation function are applied. The cropping is necessary due to the loss of border pixels during each convolution. The final layer uses a 1x1 convolution to map each component feature vector to the desired number of classes. The network architecture is illustrated in the Figure 2.10.

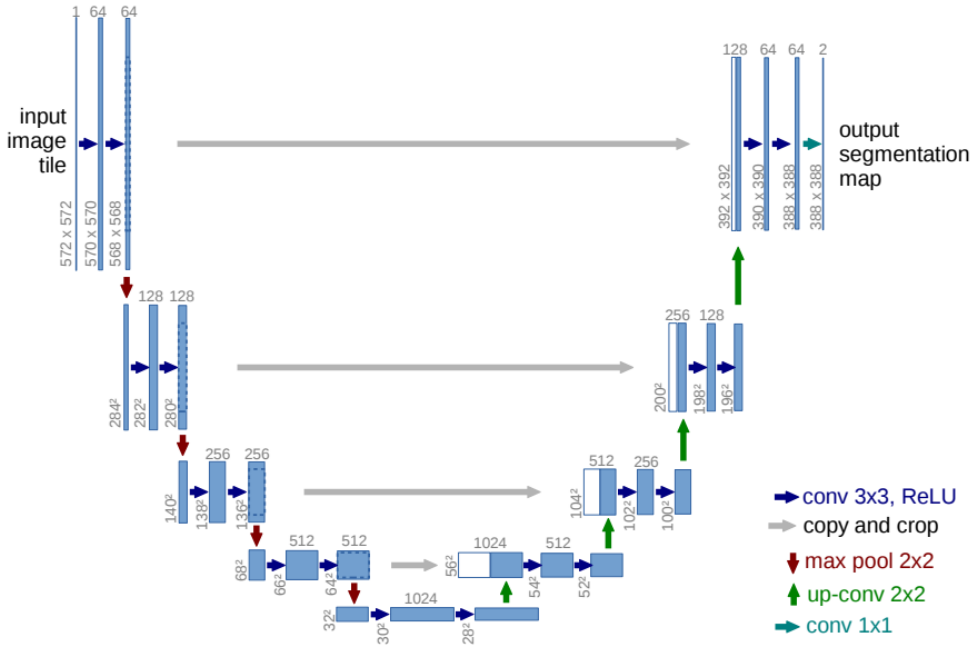


Figure 2.10: Illustration of U-net architecture[17].

The U-Net model is able to capture the overall context of an image by using

a contracting path, while at the same time it allows for precise localization by using an expanding path that is symmetric to the contracting path. The high resolution features from the contracting path are combined with the upsampled output to enable localization. Additionally, the upsampling section of the model has a large number of feature channels which helps the network to propagate context information to higher resolution layers.

2.5.7 U-Net ++

UNet++[23] is a type of deeply-supervised encoder-decoder network that utilizes an encoder-decoder structure with multiple layers of nested and dense connections, called skip pathways, that link the encoder and decoder sub-networks. These pathways are designed specifically to minimize the difference between the feature maps generated by the encoder and decoder, thus improving the final output of the segmentation. The experiments that is done in "UNet++: A Nested U-Net Architecture for Medical Image Segmentation" study , the suggested architecture provided an important performance gain compared to standard U-Net model on medical images. UNet++ uses a hierarchical architecture, where multiple levels of encoders-decoders are stacked on top of each other to extract features at different scales. Each level of the encoder extracts features at a different resolution, and each level of the decoder uses these features to make predictions at a corresponding resolution. This allows the network to incorporate both low-level and high-level features, which improves the accuracy of the segmentation.

The network also uses a skip connection between the encoder and decoder, which allows the network to combine features from different levels of the encoder. This helps to preserve fine details in the segmentation while still incorporating contextual information from the encoder. Additionally, UNet++ uses attention gates to selectively propagate information from the encoder to the decoder, which helps to reduce noise and improve the accuracy of the segmentation.

With respect to the Figure 2.11: (a) UNet++ is an architecture that uses an encoder and a decoder that are connected through a series of nested dense convolutional blocks. The aim of UNet++ is to reduce the semantic gap between the feature maps generated by the encoder and the decoder before they are combined. As seen in the graphical abstract, UNet++ is composed of the original U-Net represented by black blocks, the dense convolution blocks on the skip pathways which are represented by green and blue blocks, and deep supervision represented by red blocks. These red, green, and blue components distinguish UNet++ from U-Net.

(b) Analysis of the first skip pathway detailed for UNet++.

(c) UNet++ can be more efficient during the inference phase if it is trained with deep supervision, by removing less important parts of the network..

Overall, UNet++ is an elaborated version of the UNet architecture that addresses some of its limitations and in some cases it allows to achieve better performance in image segmentation tasks.

Difference Between UNet and UNet++

The primary distinction between UNet and UNet++ is in their design and approach to feature extraction and feature fusion. UNet employs a single encoder-decoder structure, whereas UNet++ employs a hierarchical design with many encoder and

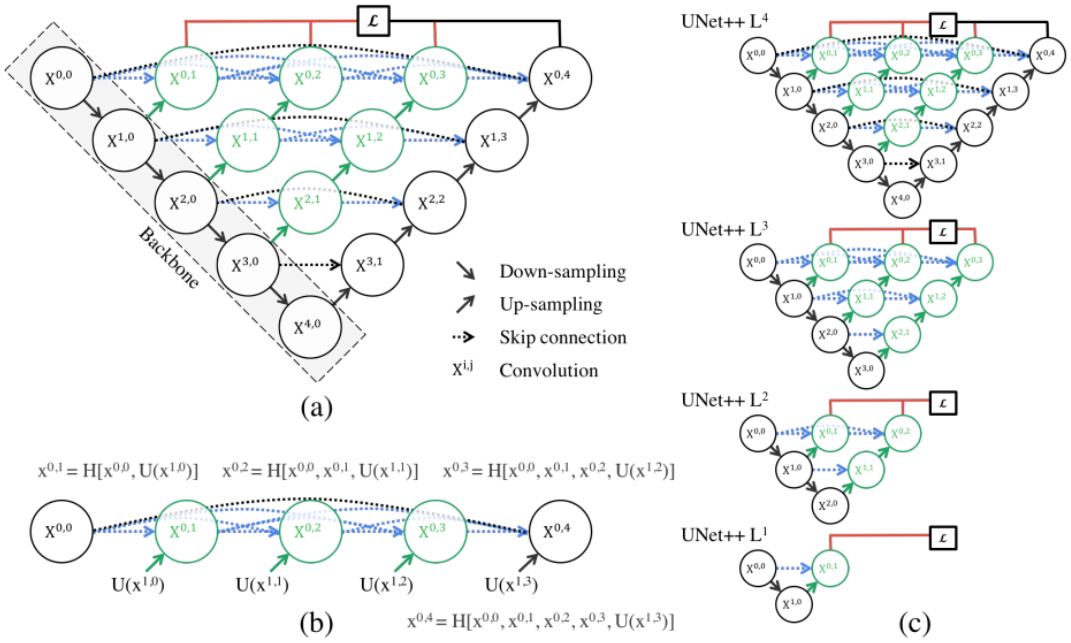


Figure 2.11: High-level overview of Unet++[23]

decoder layers. UNet employs the same set of filters at all scales, whereas UNet++ employs multiple sets of filters at various sizes.

UNet combines features from several layers of the encoder, whereas UNet++ selectively propagates information from the encoder to the decoder via skip connections and it is designed to extract more contextual and fine details information than UNet. It is designed to reduce the semantic gap between the feature maps of the encoder and decoder sub-networks, while UNet uses skip connections to preserve fine details in the segmentation. The re-designed skip pathways also connect the use of deep supervision which gives possibility to network to work in two different modes. First mode is called as accurate mode that averages outputs from all segmentation branches. The second mode is called the fast mode and provides the final segmentation map selected from only one of the segmentation branches. The mode is selected according to the need.

2.6 Evaluation Metrics

When assessing the performance of various semantic segmentation network architectures, it is necessary to use an convenient metric to precisely describe the network capability of identifying a class. In this study three different metrics are employed to evaluate model performance that are widely used in semantic segmentation. These are mean Intersection over union, dice coefficient and pixel accuracy.

mIoU

mIoU stands for "mean Intersection over Union" which is a commonly used metric in computer vision, particularly for semantic segmentation tasks where the evaluation is based on the overlapping between the ground truth image and the predicted

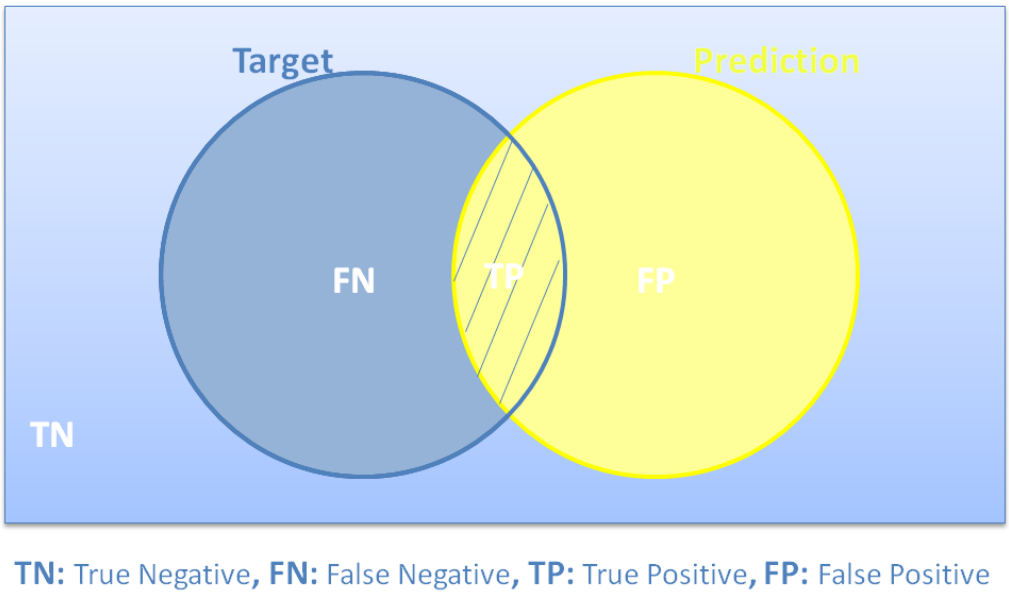


Figure 2.12: Auxiliary image to define metrics.

image. mIoU is in fact, used to evaluate the degree of overlap between the predicted segmentation map produced by a model and the ground truth segmentation map. It is calculated by taking the overlap of the predicted and target masks and dividing it by the union of the two masks and taking the average of this value of all classes.

Regarding the image shown in Figure 2.12, it highlights three different regions, namely FN, TP, and FP, which correspond to False Negative, True Positive, and False Positive, respectively:

$$IoU = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

This is the formulation of Intersection over Union and mean of IoU is:

$$mIoU = \frac{1}{N} \sum_{i=1}^{N} \frac{\mathrm{TP}_i}{\mathrm{TP}_i + \mathrm{FP}_i + \mathrm{FN}_i}$$

where N is the number of classes and mIoU is the average Intersection over Union for all classes.

mIoU is a scalar value between 0 and 1, with 1 representing a perfect match and 0 representing no overlap.

It is used to assess the model's overall performance by considering the performance of all classes, not just the most common ones.

Dice Coefficient

The Dice coefficient is used to compare the overlap between two image (ground truth and the prediction). It provides a scalar value between 0 and 1. Value 1 indicates identical images and 0 indicates no elements in common. Therefore, values closer to 1 means a better match.

With respect to the image in Figure 2.12 Dice Coefficient:

$$DC = \frac{2 \cdot \mathrm{TP}}{2 \cdot \mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$

It is calculated by taking twice the intersection of the two masks and dividing it by the total number of pixels in both masks.

Pixel accuracy

Pixel accuracy is a widely used metric for evaluating the performance of image classification and semantic segmentation models and is defined as the ratio of correctly classified pixels in an image to the total pixels. It provides a scalar value between 0 and 1. While value 1 is representing a absolute match, value 0 is representing zero-match (no match at all). Therefore, this metric only considers whether pixels are correctly classified or not However, it does not take class imbalance into account. With respect to the image in Figure 2.12:

$$PA = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

Chapter 3

Methodology

This chapter helds the techniques used to perform semantic segmentation on mesh and wire. First, the data-set is analyzed, and the necessary pre-processing steps to prepare the data and run the models effectively will be explained. Then, different model structures are utilized, along with the metrics and loss functions used to refine the models for comparison. The following section covers the procedure to evaluate the performance of the final models on test data and determine the overall accuracy of the models. Finally, grid search is employed to find the optimal parameters for the employed models.

3.1 Data processing for Semantic Segmentation

Pre-processing stands for employed techniques that assures the data used in training and evaluation of the model is suitable for the given task. The aim of this step is improving to quality of data and removing unwanted features and doing feature extraction from raw data and so improving the reliability of the model. The semantic segmentation pre-process typically involves collecting images and annotations to assign pixel-level labels, cleaning data and converting it into a suitable format to use in the model. Optionally, augmenting data to increase diversity and possibly avoid overfitting, and splitting data as training, validation, and test sets (typically). These steps are essential to prepare the data to feed the deep models so that can be effectively utilized by semantic segmentation models, leading to more accurate and reliable results.

3.1.1 Data collection and labeling

Data collection contains acquiring images and annotations for each class of interest. Approximately 500 high-resolution images with dimension of 4928 x 3264 were captured at three different locations in Sicily in order to construct the dataset. These images are representing rocky walls covered by meshes and, in many cases, also contain spikes and wires.

The annotations define the region of interest of the images belong to a particular class. They can be done manually by some devolopment tools that have a user interface for manually annotating image data. In this thesis, the labeling process is carried out using a Python library named *LabelMe*[22]. For each image in the dataset, the masks corresponding to wires and meshes are created using polygons,

thus the data that is suitable for semantic segmentation is obtained. LabelMe is an open-source image annotation tool and it is designed for image segmentation and object recognition tasks. Users can manually label and annotate objects in images with this tool. Creating segmentation masks and annotated data can be used for model training such as semantic segmentation and object detection models. Moreover, LabelMe is easy to use due to its graphical interface for annotating images, which makes it a widely adopted tool for preparing data in computer vision tasks.

Figure 3.1 shows an example of the LabelMe interface used for annotation and the labelMe interface.
Figure 3.2 shows the corresponding mask of an image that is annotated by labelMe.

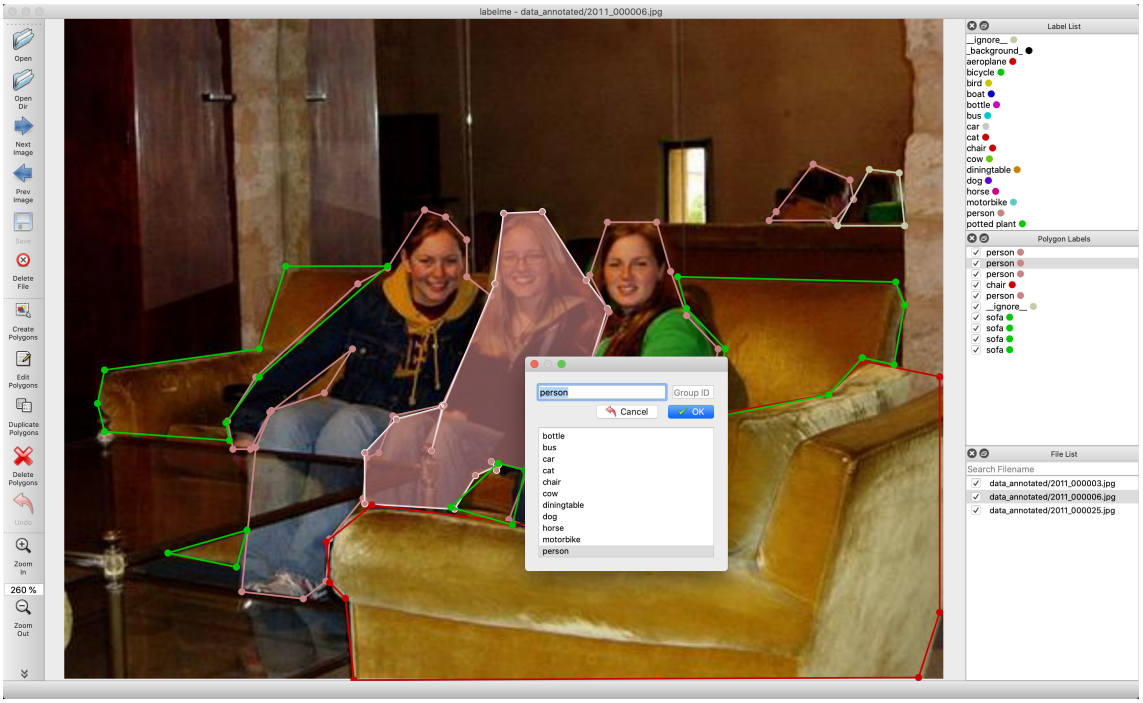


Figure 3.1: An example of how semantic segmentation annotations done by LabelMe[22].

Figure 3.2: Semantic segmentation mask creation is demonstrated with its corresponding image after annotations done by LabelMe[22].

In this study, only wire and mesh are considered since they are more suitable for semantic segmentation application. Figure 3.3 displays an image with annotations to demonstrate the process of annotation.

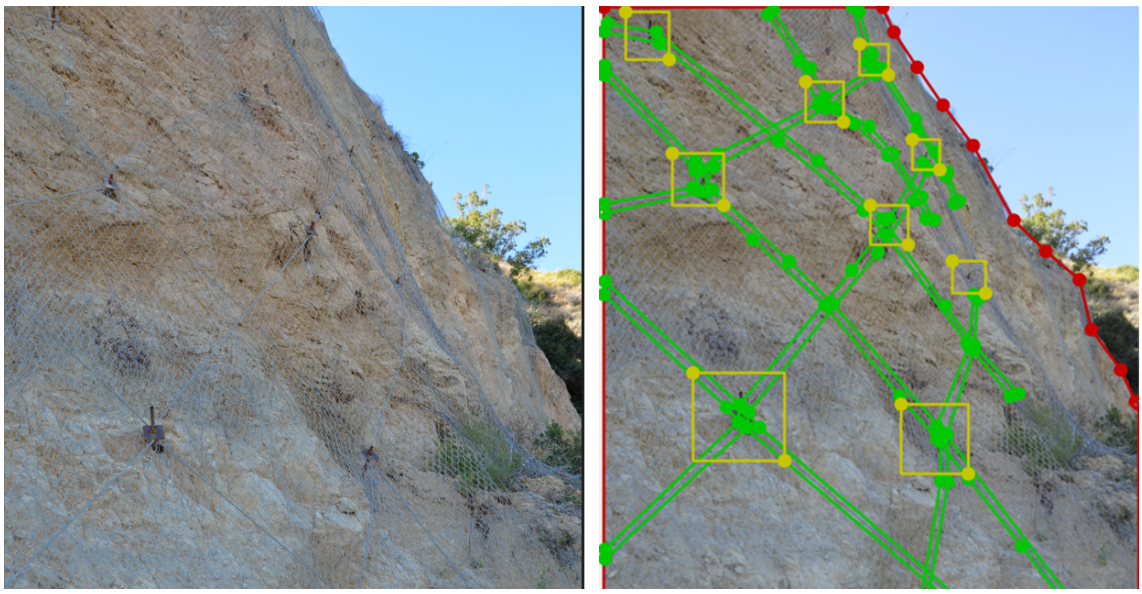


Figure 3.3: A demonstration of the labeling process performed using the LabelMe[22] annotation tool is given. Original image and annotated version is shown respectively. The yellow boxes indicate spikes(that are not considered in this thesis), the green polygons represent wires, and the red polygons are for mesh.

3.1.2 Image Pre-processing

Pre-processing includes cleaning and transforming the data so that it can be fed into model and this can involve image resizing, pixel values normalization,encoding class labels to numerical values etc. In this thesis, acquired images are captured in both horizontal and vertical orientations. Since the models used require a fixed input size, three cuts for each horizontal image (left, middle, right) and two cuts for each vertical image (bottom, middle) are taken to create dataset. The vertical orientation was limited to two crops because the top portions were often too distant from the point the pic was taken (likely because the images were taken from the ground) and resulting in unclear visibility of the elements in image. This cropping operation is shown in the Figure 3.4. These cropped images are square in shape with a size of 3264x3264 pixels. The mesh and wire images are then resized as 512x512.

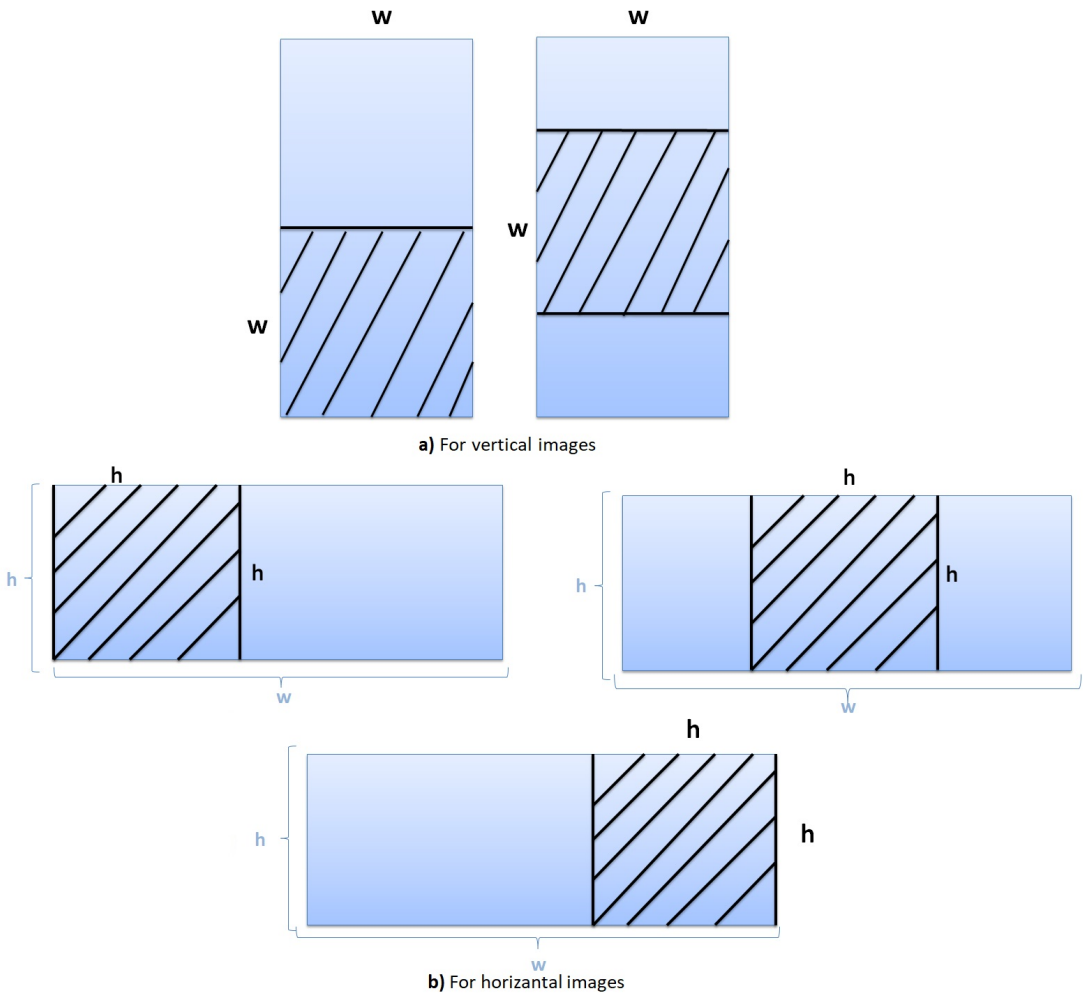


Figure 3.4: A demonstration is provided of the horizontal and vertical cuts applied to the input images. It shows the vertical cuts identified as 'center' and 'bottom' and the horizontal cuts identified as 'center', 'left', and 'right'.

3.1.3 Data Splitting

At the beginning, there were 451 images (horizontal and vertical), these images are composed of thee different sites of a region, in this thesis these sites are named as site 1 (154 images), site 2 (111 images) and site 3 (186 images). Then, these images are cropped, from these pieces two different data-set are obtained for wire and mesh separately. Mesh data-set contains 1137 images and wire data-set contains 1036 images because the images that does not contain any piece of wire and mesh are discarded. Therefore the number of images are different for mesh and wire. The Figure 3.5 shows the process of how the mesh and wire data-set obtained.

Splitting data-set involves dividing the data as training, validation, and test sets (typically). The training set is to train the model, the validation set is for the model selection and lastly, the test set is used for evaluating the performance of the model.

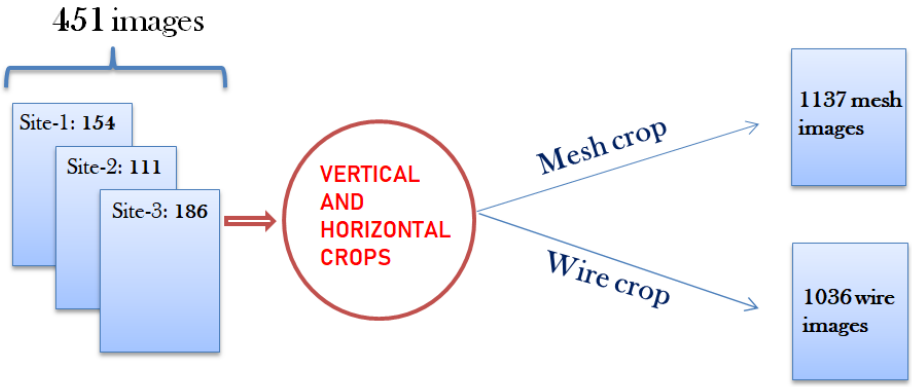


Figure 3.5: Data-set creation step by step and indication of the number of images of each data-set.

For the following experiments the data for mesh and wire are evenly split into three sets: 75% for training, 15% for validation, and 10% for testing. The images were divided in such a way that the cuts of the same image were not placed in the same set and were distributed fairly among the sites. At the end for mesh data-set: training set contains 854 images, validation set has 163 and test set has 120 images. For wire set: training set contains 772 images, validation set has 152 and test set has 112 images.

3.1.4 Data Augmentation

Data augmentation involves typically generating additional data to add training set by applying transformations to the existing images. This transformation may include flipping, rotation, scaling, adding random noise into images etc. Data augmentation most possibly help preventing overfitting and increases the diversity of the training data.

Data augmentation is employed to assess its impact on accuracy, and the same method used for both wire and mesh. For each image in the training and validation sets, a modified version was created by randomly altering lighting and with a specified probability, adding noise and flipping the image horizontally. This doubled the number of examples in the training and validation sets. The Figure 3.6 shows an example of data augmentation that is employed for the dataset.

Figure 3.6: An example of the original image (left) and the augmented image (right).

The Figure 3.7 shows the process of how the mesh and wire data-set with data augmentation employed are obtained.

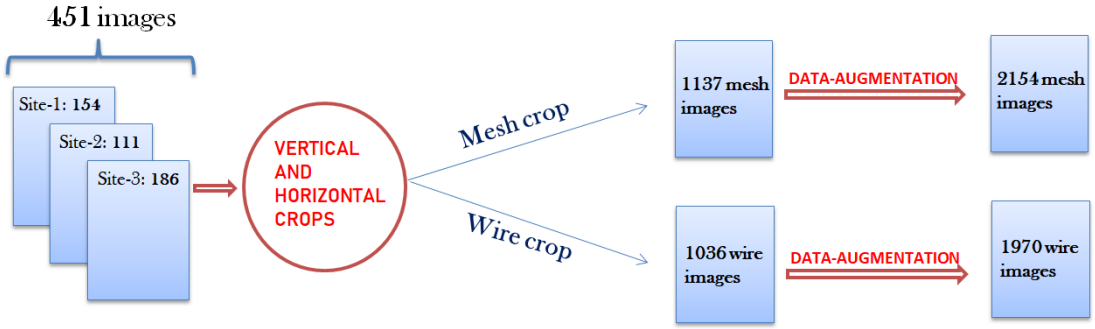


Figure 3.7: Data-set creation with data augmentation step by step and indication of the number of images of each data-set.

After data augmentation employed on validation and training sets, two different data set is created for mesh and wire separately. At the end for mesh, training set contains 1708 images, validation set has 326 and test set has 120 images. For wire set: training set contains 1544 images, validation set has 304 and test set has 112 images.

3.2 Binary Semantic Segmentation with U-Net

In this thesis, separate U-Net models are used to conduct further research on semantic segmentation of wire images and mesh images using a neural network.

Standard U-Net model that was proposed in the original paper [17] is taken as base model. The proposed model is starting with 64 filters in the first down-sampling block and in each block it doubles the filter size until bottleneck layer.

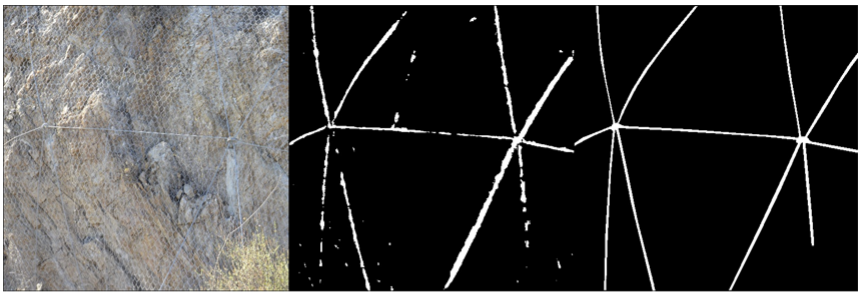
After bottleneck layer, number of filters are halved in each layer until it gets a symmetric correspondence of expansive path. Standard U-Net model is composed of two paths: contracting path and expansive path. The down-sampling (contracting) consists of 4 blocks that contain 2 convolutional layers followed by max-pool layer. After the contracting path, there is a bottleneck layer and it is followed by the expansive path (up-sampling blocks). In the expansive path, the feature maps are upsampled by transposed convolutions. Then, the up-sampled feature maps are concatenated with the corresponding feature maps from the contracting path to preserve the spatial details. The final part of expansive path contains 3 convolutional layers where the last layer is an activation function acting on each pixel. As loss function binary cross entropy and for optimizer Adam is employed.

In this experimental setup, sigmoid is chosen as activation function in the output layer. Thus, each pixel of the output is a value between 0 and 1. In the prediction phase, values lower than 0.5 are set to 0 and those higher are set to 1.

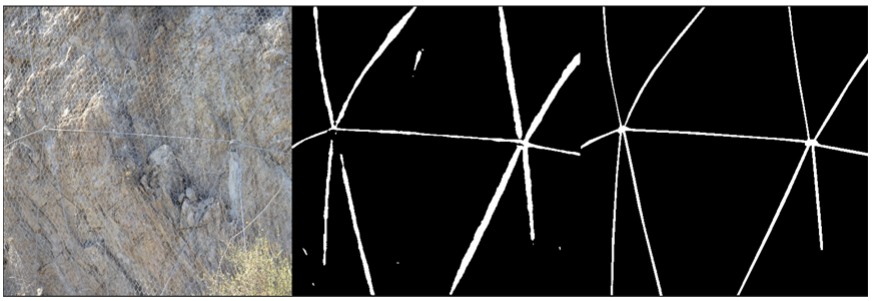
The model takes as input an image of size (512,512,3) with target the mask of size (512,512,1). Since the problem is binary segmentation mask has one color channel. The standard U-Net that is proposed in the paper written by Ronneberger[17] has dimension 572x572. Additionally there are cropping layer to build a robust network because the spacial dimension varies in each up-sampling down-sampling blocks and if the dimension of the images are not power of two the output image that is obtained at the end does not give the same spacial dimension. In our experiment to prevent this issue 512x512 images are employed and cropping layers (crop2D) are removed. Output returns the predicted mask with the same size as the target. Image values are normalized to [0,1].

As evaluation metrics, mean intersection over union (IoU), Dice Coefficient and pixel accuracy is indicated. However selecting the best model will be only considered on mIoU. Since it gives the overlapping rate of the ground truth label and the prediction label (also dice coefficient works in a similar way), it is the most widely used and reliable evaluation metric. It is worth to mention that, pixel accuracy may be miss-leading since the amount of background in an image may be unbalanced and mostly much higher compared to wire and mesh. For instance if the amount of background pixels in an image are higher than mesh/wire amount, and the model predicts that all the image is composed of only background then pixel accuracy would give high result that is not reliable. That is why pixel accuracy only for mesh/wire is considered that gives that accurately predicted pixels only for mesh/wire which provides a robust evaluation.

This experiment utilized a data augmentation employed data-set of images, rotation, flipping,adding random noise and saturation adjustment are applied randomly to images too obtain new data-set with data augmentation.

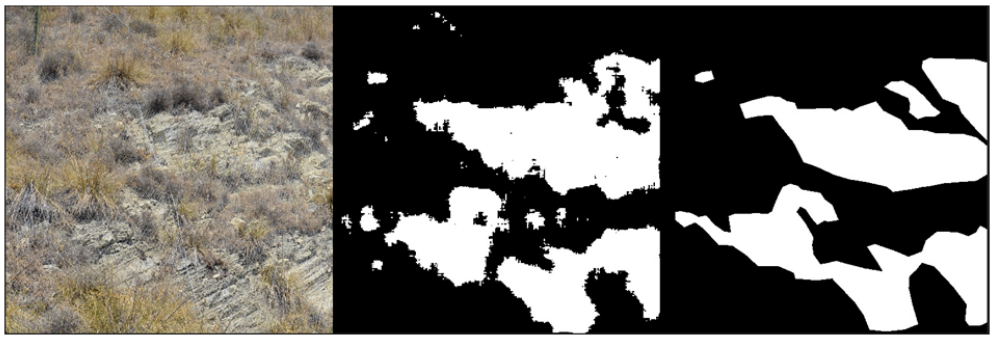


a. A sample of wire prediction by standard U-Net

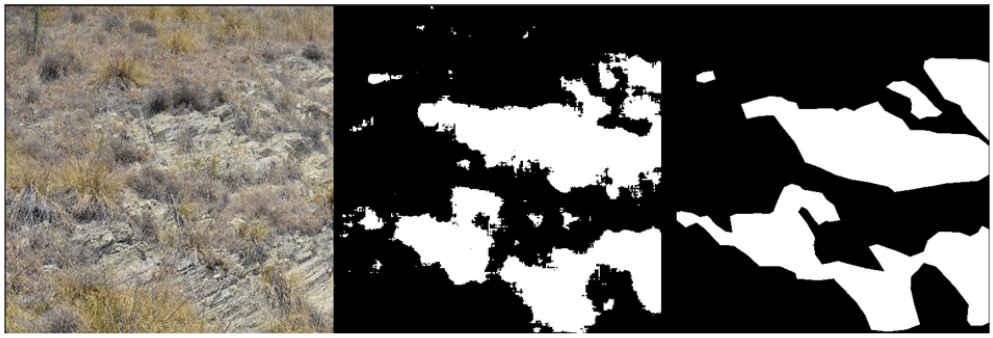


b. A sample of wire prediction by standard U-Net on data augmentation applied wire images

Figure 3.8: Samples of wire prediction that is trained by U-Net model with and without data augmentation employed wire images. Input image in the left, prediction in the center and target mask in the right.



a. A sample of mesh prediction by standard U-Net



b. A sample of mesh prediction by standard U-Net on data augmentation applied wire images

Figure 3.9: Samples of mesh prediction that is trained by U-Net model with and without data augmentation employed mesh images. Input image in the left, prediction in the center and target mask in the right.

	Wire	Wire with augmentation	Mesh	Mesh with augmentation
Mean IoU	51.1%	**52.1%**	87.2%	**87.8%**
Dice coeff.	67.7%	68.5%	93.1%	93.5%
Pixel accuracy	96.9%	96.8%	91.7%	92.1%
Pixel accuracy wire or mesh only	60.4%	63.3%	90.0%	94.4%

Table 3.1: standard U-Net results on test set of mesh and wire images separately for mesh and wire models.

The obtained results on test set of standard U-Net model trained on wire and mesh images, with and without data augmentation employed are reported in Table 3.1. Model performance on a sample composition of original image, prediction and ground truth mask is shown in the Figure 3.8 for wire images with and without data augmentation employed. In the Figure 3.9 for mesh images with and without data augmentation employed is demonstrated.

In conclusion, this binary semantic segmentation experiment showed that data augmentation improves performance of the model and the results are reasonably good. Although evaluation results are high the prediction sample images are noisy. In order to reduce this noisy view, model improvement is crucial. A dropout layer may help model to be less complex and regularize it. Dropout is a regularization technique that is widely used in deep neural Networks. For reducing the neurons

co-adaptation effect, dropout layers are working in a way that it randomly removes a specified ratio of neurons so that remaining neurons can learn more robust features in a neural network during the training procedure. Thus this technique is used for reducing the effect of overfitting the training data and to obtain less complex models.

In the following experiment, the same data-sets will be exploited with dropout layer added standard U-Net model.

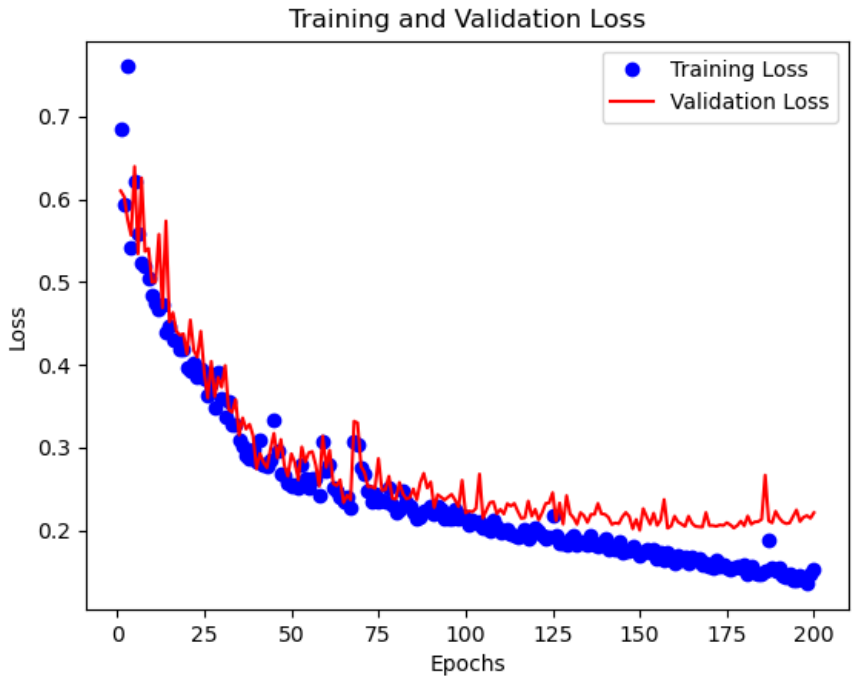


Figure 3.10: Training and validation loss behavior during the training on mesh images.

For observation purpose the validation loss - training loss plot will be compared after employing dropout layers, previous model behaviour is visualized as a graph in the figure 3.10 on mesh images. Since early stopping procedure was employed overfitting is prevented but with dropout layers it is expected to have longer training and model to learn more features. The graph shows that after 150th epoch model started to give higher distance between training loss and validation loss.

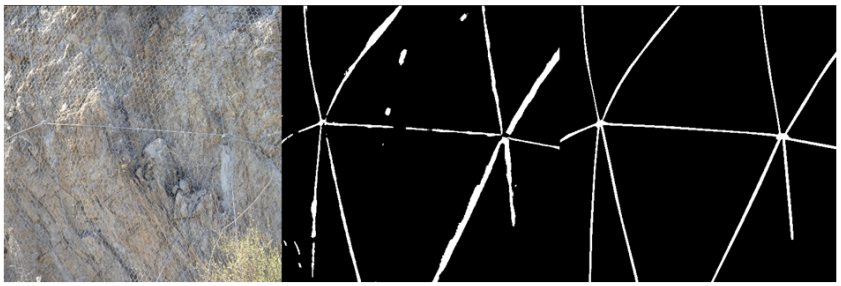
3.3 Binary Semantic Segmentation with U-Net, with dropout layers

In this experiment the previous model is taken as base and added dropout layers in each encoder decoder blocks. First and second block have dropout rate 0.1, third and forth blocks have 0.2. Bottleneck layer has dropout rate : 0.3 and expansive path have the same values symmetrically (respectively 0.2, 0.2, 0.1, 0.1).

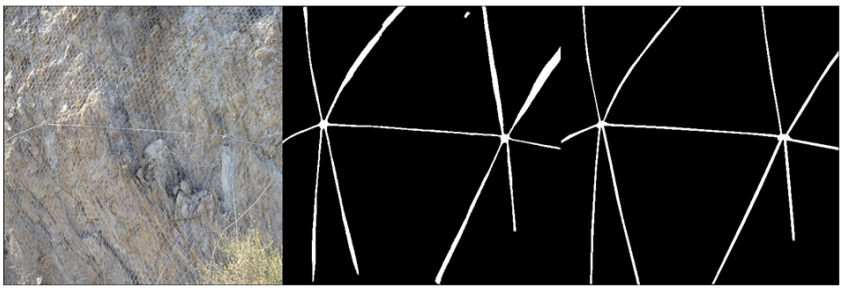
	Wire	Wire with augmentation	Mesh	Mesh with augmentation
Mean IoU	58.1%	**58.8%**	81.1%	**89.2%**
Dice coeff.	73.4%	74.1%	89.6%	94.3%
Pixel accuracy	97.3%	97.4%	87.4%	93.1%
Pixel accuracy wire or mesh only	68.2%	69.1%	90.6%	95.2%

Table 3.2: results of standard U-Net with dropout layers on test set of mesh and wire images separately for mesh and wire models.

The obtained results on test set of standard U-Net model with dropout layers added, trained on wire and mesh images, with and without data augmentation employed are reported in Table 3.2. Model performance on a sample composition of original image, prediction and ground truth mask is shown in the Figure 3.11 for wire images with and without data augmentation employed. In the Figure 3.12 for mesh images with and without data augmentation employed is demonstrated.

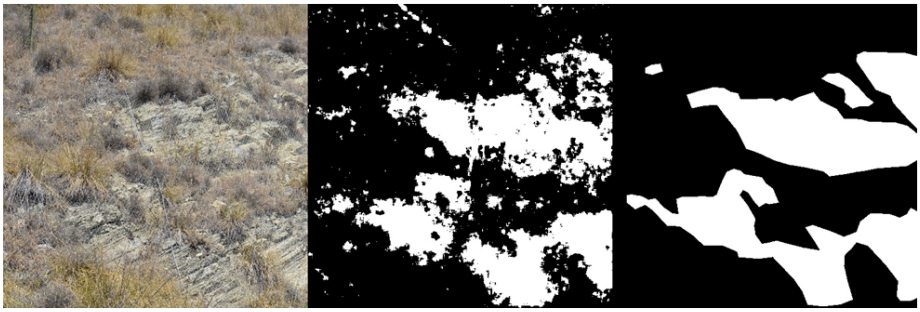


a. A sample of wire prediction by standard U-Net with dropout layers

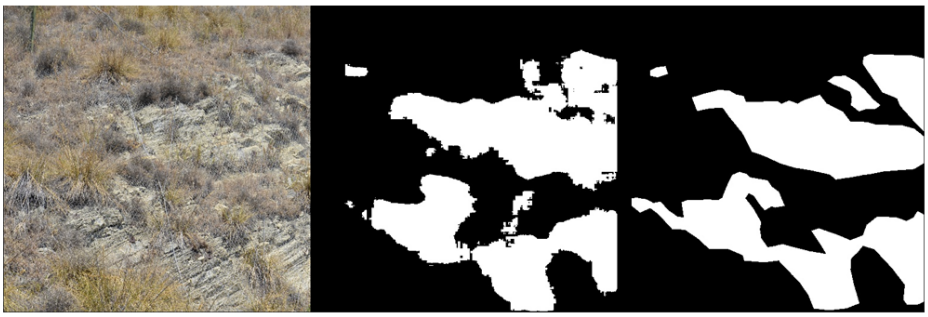


b. A sample of wire prediction by standard U-Net with dropout layers on data augmentation applied wire images

Figure 3.11: Samples of wire prediction that is trained by standard U-Net model with dropout layers on with and without data augmentation employed wire images. Input image in the left, prediction in the center and target mask in the right.



a. A sample of mesh prediction by standard U-Net with dropout layers



b. A sample of mesh prediction by standard U-Net with dropout layers on data augmentation applied wire images

Figure 3.12: Samples of mesh prediction that is trained by standard U-Net model with dropout layers on with and without data augmentation employed mesh images. Input image in the left, prediction in the center and target mask in the right.

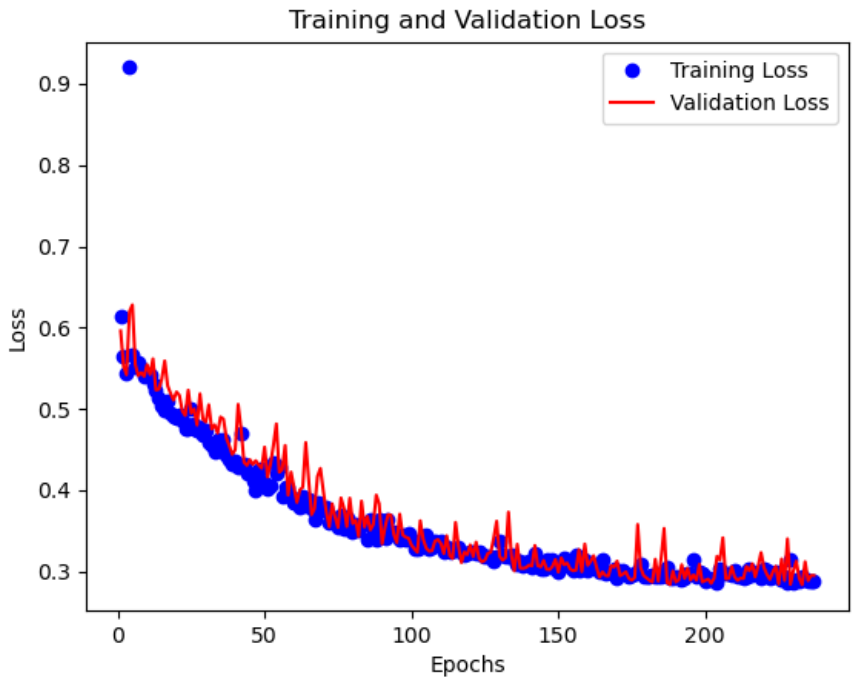


Figure 3.13: Training and validation loss behavior of U-Net with dropout layers during the training on mesh images.

Figure 3.13 shows the validation loss - training loss plot for U-Net model on mesh images with dropout layers. Training procedure took longer time (since there was early stopping procedure and it did not stop the network during training), and model learnt more features without over-fitting the training data.The model is more robust compared to previous model (without dropout layers) as expected because the gap between training loss and validation loss is smaller in the second model. The inference from this plots is that, dropout layers worked well in terms of regularization which is also visible on the results table 3.2.

In conclusion, this binary semantic segmentation experiment with dropout layers employed showed that dropout layers improved model performance with data augmentation applied data-set. This improvement is obviously seen in mesh image samples and the evaluation results are better than previous experiment. In the figure 3.12 image b seems reasonably accurate for mesh sample, as well as figure 3.11 image b for wire sample. As it is intended, the noise in the image prediction is reduced. Thus, dropout layer helped the model to be less complex and model could learn better.

3.4 Binary Semantic Segmentation with U-Net++

U-Net++ model consists of an encoder and decoder path. The encoder path involves a series of convolutional blocks with max pooling layers (that is resulting in a down-sampled output) and that down-samples the input image. On the other hand the decoder path involves a series of upsampling layers that concatenates the corresponding encoder layer (concatenation layer merges multiple input tensors along a specified axis) followed by transpose convolutional block (used for upsampling the input data that has 2x2 filters with stride 2 and same padding), and the final output layer has a sigmoid activation function.

U-Net++ have a deep supervision flag, if it set to True, it adds additional output layers with sigmoid activation for each skip connection, allowing intermediate outputs at different scales. However in this study Deep supervision is set as disables and consider U-Net++ model as default setting. In this architecture a Batch Normalization layer is used that normalizes the output of previous layer by subtracting the mean and dividing it by the standard deviation of the output and so making the model more robust and stable during training. As activation function ReLU (Rectified Linear Unit) is used, which sets all negative values to zero and leaves positive values unchanged.

The proposed UNet++ model starts with filter size 32 that is used as standard U-Net++ model application in practice by the U-Net++ paper[23]. This model increases the filter size up to 512 filters in each encoder block. It is different that the standard UNet model which starts with 64 filters and goes up to 1024. (This thesis focuses on comparing standard models.)

3.4.1 U-Net++ model performance results

The data-set that is exploited in this experiment consists of images with a resolution 512x512. The data-set is divided into three subset in pre-processing step as it was in U-Net experiment; For no data augmentation techniques employed data-set, for mesh: training set contains 854 images, validation set contains 163 images, and test set contains 120 images. For wire set: training set contains 772 images, validation set has 152 and test set has 112 images.

For data augmentation applied set, for mesh: the data-set is divided into three sub-sets: a training set consisting of 1708 images, a validation set consisting of 326 images, and a test set consisting of 120 images. For wire, data-set split is: Training 1544, validation 304, test 112.

The U-Net model is trained for 250 epochs and Adam optimizer is used, with a batch size of 32 and a binary cross-entropy loss function. The early stopping process is monitored by calculating the mean Intersection over Union (IoU) on the validation set, and the patience was set to 50.

The obtained results on test set of U-Net++ model trained on mesh and wire images with and without data augmentation employed approaches are reported in Table 3.3.

U-Net ++	Wire	Wire with augmentation	Mesh	Mesh with augmentation
Mean IoU	54.5%	**56.1%**	88.8%	**89.3%**
Dice coeff.	70.6%	71.8%	94.1%	94.4%
Pixel accuracy	97.2%	97.2%	93.0%	93.2%
Pixel accuracy wire or mesh only	63.0%	66.6%	93.5%	93.6%

Table 3.3: Report of U-Net++ results on test set on wire and mesh images with and without data augmentation

A sample of each wire and mesh prediction that is trained by separate U-Net++ model in the scope of Binary Semantic Segmentation is shown in the figure 3.14

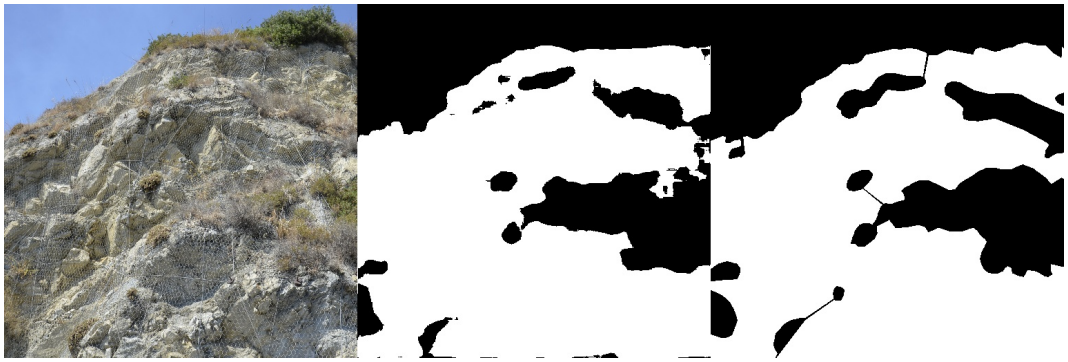


Figure 3.14: A sample of **mesh** prediction that is trained by U-Net++ model on **no data augmentation** employed mesh images. Input image in the left, prediction in the center and target mask in the right.

A sample of mesh prediction that is trained by U-Net++ model on no data augmentation employed mesh images is shown in the figure 3.14.

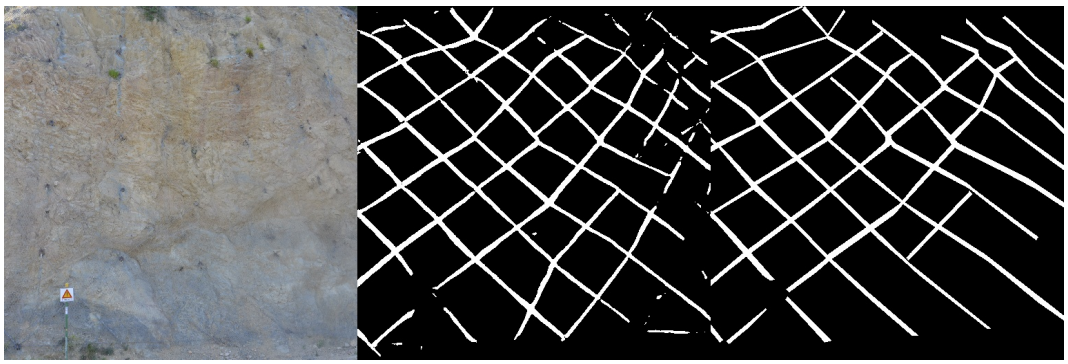


Figure 3.15: A sample of **wire** prediction that is trained by U-Net++ model on **no data augmentation** employed wire images. Input image in the left, prediction in the center and target mask in the right.

A sample of wire prediction that is trained by U-Net++ model on no data augmentation employed wire images, is shown in figure 3.15

Figure 3.16: A sample of **mesh** prediction that is trained by U-Net++ model on input images that is utilized **with augmentation** techniques. Input image in the left, prediction in the center and target mask in the right.

A sample of mesh prediction that is trained by U-Net++ model on input images that is utilized with augmentation techniques, is shown in figure 3.16

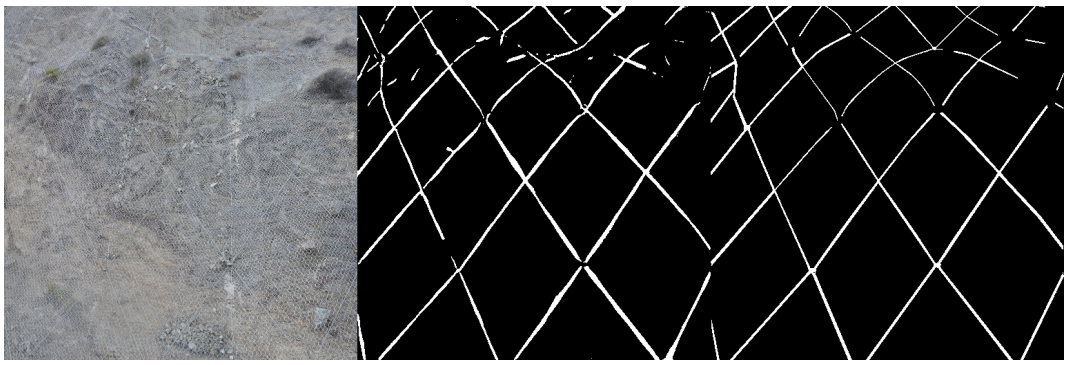


Figure 3.17: A sample of **wire** prediction that is trained by U-Net++ model on input images that is utilized **with augmentation** techniques. Input image in the left, prediction in the center and target mask in the right.

A sample of wire prediction that is trained by U-Net++ model on input images that is utilized with augmentation techniques, is shown in figure 3.17

U-Net vs U-Net++	U-Net wire	U-Net++ wire	U-Net mesh	U-Net++ mesh
Mean IoU	52.1%	**56.1%**	87.8%	**89.3%**
Dice coeff.	68.5%	71.8%	93.5%	94.4%
Pixel accuracy	96.9%	97.2%	92.1%	93.2%
Pixel accuracy wire or mesh only	63.3%	66.6%	94.4%	93.6%

Table 3.4: comparison of U-Net and U-Net++ models trained on data augmentation applied data.

To conclude this experiment, it is worth to say that the U-net++ model gave good results, even better than U-Net model. However, there is no big difference between two model performances. Since the standard U-Net model is a lighter weight and less complex network compared to U-Net++, and the performance of both models is good enough, it is better to investigate the U-net model rather than UNet++ to search for the optimal parameter selection phase in the following chapter.

Comparison of these two models on data without data augmentation employed is reported in table 3.4 to better compare the results.

3.5 Multi-class Semantic Segmentation with U-Net

Following experiment aims to determine if multi-class semantic segmentation approach leads to improve the detection of mesh and wires. Annotated data was pre-processed by labeling each pixel with a value of 0 indicating the background, 1 for the mesh, and 2 for the wire.
This application is shown in Figure 3.18 This annotated data is then converted into a numerical format by one-hot encoding.

One-hot encoding is used to represent categorical data in a format that can be interpretable by algorithms. In multi-class semantic segmentation, it is used for converting pixel labels into a format that can be used to feed the deep learning model during training and evaluation processes. In multi-class semantic segmentation, there are several labels that are assigned to each pixel in an image. One-hot encoding plays a role to represent these labels as a binary vector that c corresponds to a particular class. For instance, considering a multi-class semantic segmentation with three classes: wire, mesh, and background, each pixel is assigned to one of these labels. In this experiment, if a pixel is labeled as a wire, one-hot encoded representation would be [1, 0, 0], whereas if it's labeled as a mesh, it is [0, 1, 0] and lastly, if a pixel is labeled as background, its representation is [0, 0, 1]. In summary, this technique represents categorical labels as binary vector, where each element of the vector is corresponding to a possible label. The element is set as 1 if the pixel belongs to that class, or 0 otherwise. This allows the model to easily process each label as a separate class. For evaluation, metrics are adjusted for multi-class semantic segmentation with a pixel-by-pixel comparison used to identify intersecting pixels.

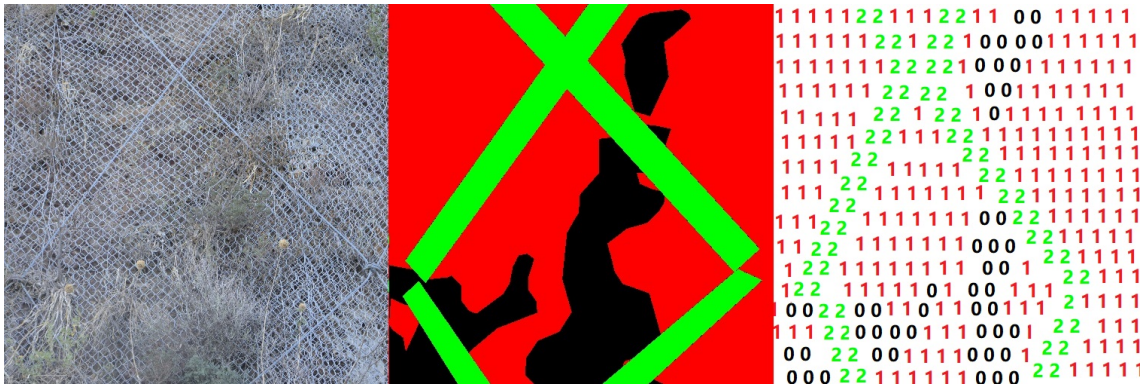


Figure 3.18: Pixel labeling is demonstrated; Original image in the left, ground truth mask in the center and corresponding pixel label in the right.

U-Net model without data augmentation is trained on data-set composed of 858 training images 167 validation images and 118 test images. Data preparation and splitting is shown in Figure 3.19

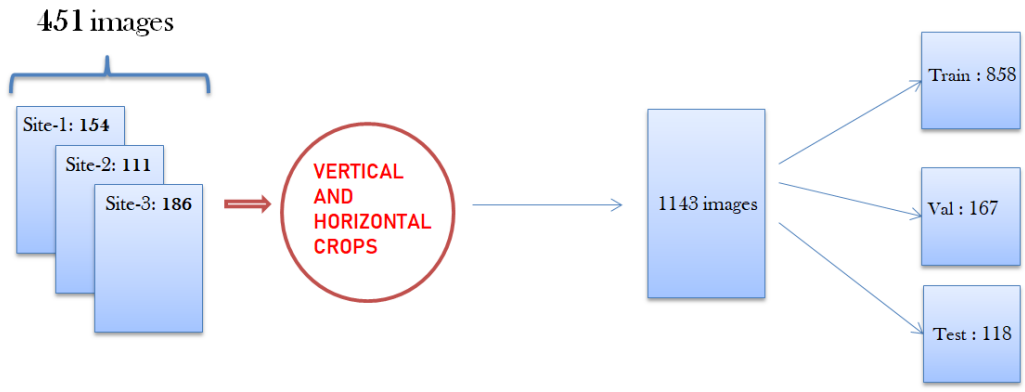


Figure 3.19: Data-set creation and splitting for multi-class semantic segmentation experiments is shown step by step.

In Figure 3.20, multi-class semantic segmentation mask is shown with the original image. Green pixels are the demonstration of wire wheres red pixels refers to mesh and black is for background.

Figure 3.20: Original image in the left and multi-class mask in the right.

The data-set used in this experiment consists of images with 512x512 pixel resolution. The U-Net model is trained for 250 epochs using the Adam optimizer, with a batch size of 32 and a Sparse Categorical Crossentropy loss. The reason for using sparse categorical entropy loss in this experiment is that each image has more pixels in the mesh category than in the wire category. To deal with the imbalance between mesh and wire pixels, the sparse categorical entropy loss function is employed which can penalize the model more for incorrectly classification for the pixels of the minority class (or classes). By taking into account the frequency of each class in the training data and assigning higher weights to the minority class, loss function encourages the model focusing on accurately classifying the pixels in the less common classes. Thus, this is leading to potentially better performance in overall. It is worth to mention that in order to avoid over-fitting the early stopping strategy was

monitored by calculating the mean Intersection over Union (IoU) on the validation set, and the patience value is set to 50.

U-Net applied on images and multi-class masks

The obtained results on test set of U-Net model trained on images are reported in Table 3.5 and a sample composition of original image, prediction and ground truth mask is shown in the Figure 3.21

Multiclass	Average	Background	Mesh	Wire
Mean IoU	63.9%	79.9%	80.8%	30.8%
Dice coeff.	75.1%	88.8%	89.4%	47.1%
Pixel accuracy	72.0%	90.8%	89.9%	35.3%

Table 3.5: U-Net results on test set with average evaluation value in the firs column, background only in the second, mesh only in the third and wire only in the last column. Optimizer employed: Adam (lr=0.0001))

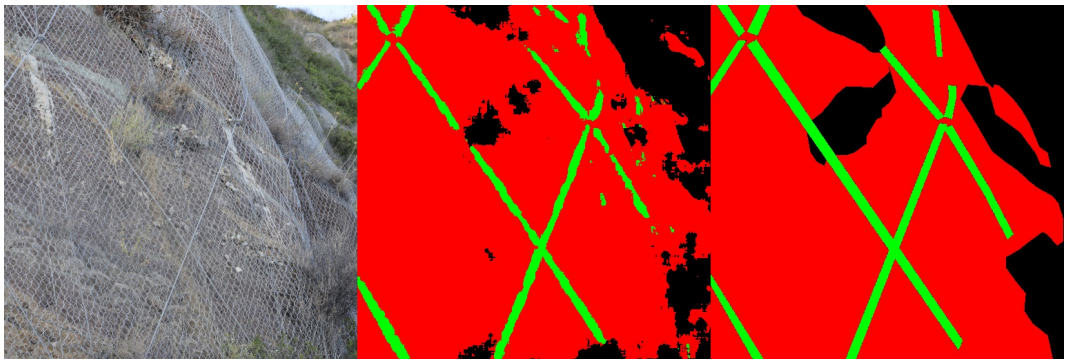


Figure 3.21: A sample of prediction that is trained by U-Net model on input images for multi-class semantic segmentation. Input image in the left, prediction in the center and target mask in the right. Model trained with Adam (lr=0.0001)

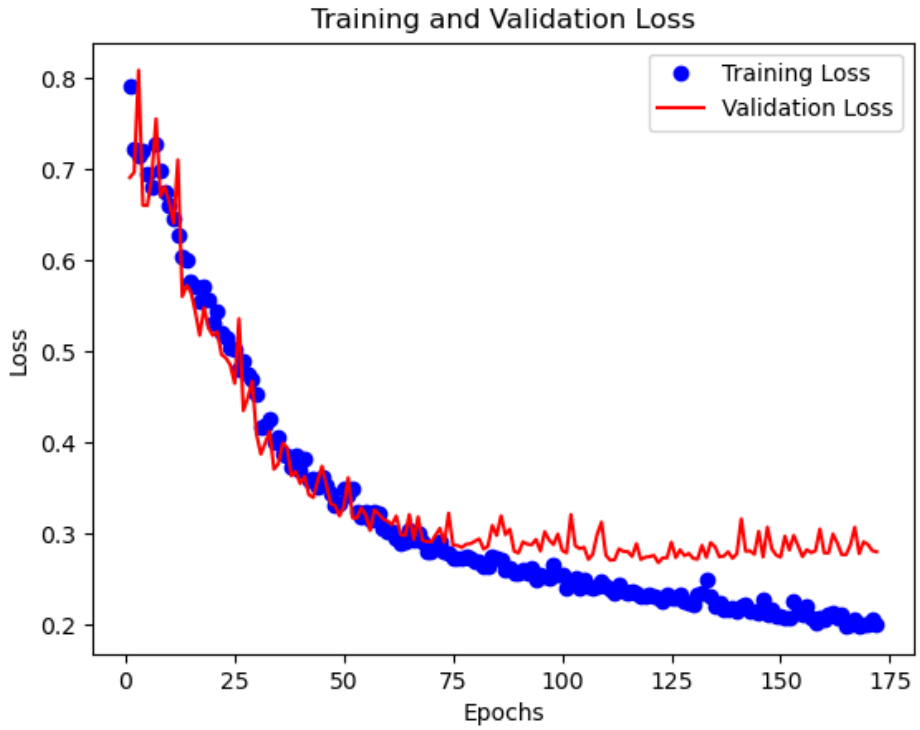


Figure 3.22: Training and validation loss behavior of U-Net model for multi-class semantic segmentation during the training

The validation loss - training loss plot is shown in the figure 3.22 and it shows that model is trained until 200th epoch even though it was set until 250 epochs. For obtaining less complex and more robust network dropout layers will be employed in the following.

3.6 Multi-class Semantic Segmentation with U-Net, with dropout layers

To improve the performance different parameters are employed and this is started with different learning rate for Adam optimizer. Therefore, same configuration is tested with default values of Adam is reported in table 3.6 and a sample composition of original image, prediction and ground truth mask is shown in the Figure 3.23

Multiclass	Average	Background	Mesh	Wire
Mean IoU	**69.1%**	80.6%	82.6%	44.2%
Dice coeff.	**80.3%**	89.3%	90.5%	61.3%
Pixel accuracy	**77.1%**	88.8%	92.1%	50.5%

Table 3.6: U-Net results on test set with avarege evaluation value in the firs column, background only in the second, mesh only in the third and wire only in the last column. Optimizer employed: Adam (lr=0.001)

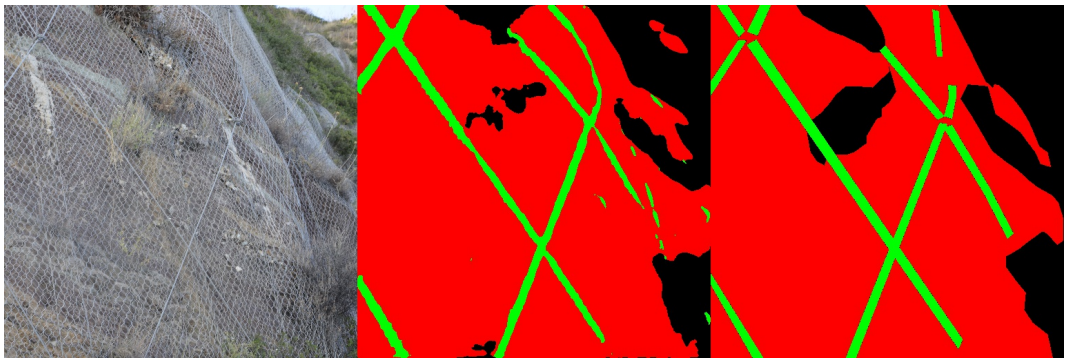


Figure 3.23: A sample of prediction that is trained by U-Net model on input images for multi-class semantic segmentation. Input image in the left, prediction in the center and target mask in the right. Model trained with Adam (lr=0.001)

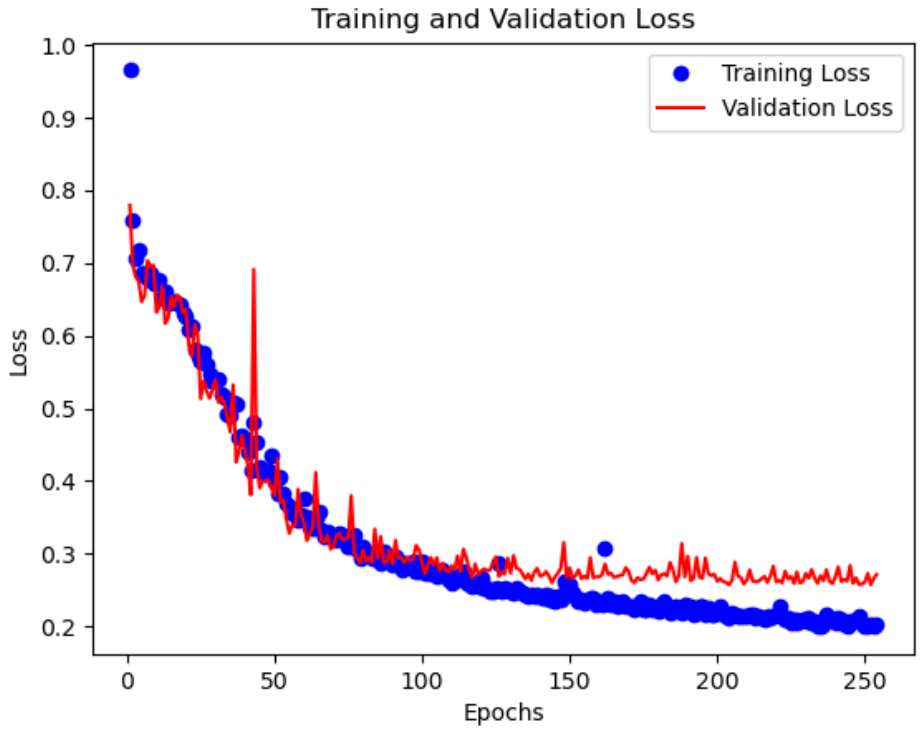


Figure 3.24: Training loss - Validation loss plot that shows training and validation loss behavior thorough the increasing epoch numbers of U-Net model with dropout layers for multi-class semantic segmentation during the training process.

Figure 3.24 shows the validation loss - training loss plot for U-Net model with dropout layers. It shows that training took longer time compared to the model without dropout layers that gave result in the table 3.5 (since there was early stopping procedure and it did not stop the network during the training), and model learnt more features without over-fitting the training data. Model is more robust with dropout layers compared to previous model as expected.
The gap between training loss and validation loss is not higher compared to the graph 3.22. Thus, as expected, dropout layers worked well with the effect of regu-

larization which is also reported at the results table 3.6.

Lastly, data augmentation applied data-set which involves 1716 training, 334 validation and 118 test images is used (data augmentation techniques are only applied on training and validation sets). How the data is prepared and split is shown in the figure 3.25. It is expected to obtain better results since data augmentation employed data-set gave better results so far in the previous experiments.

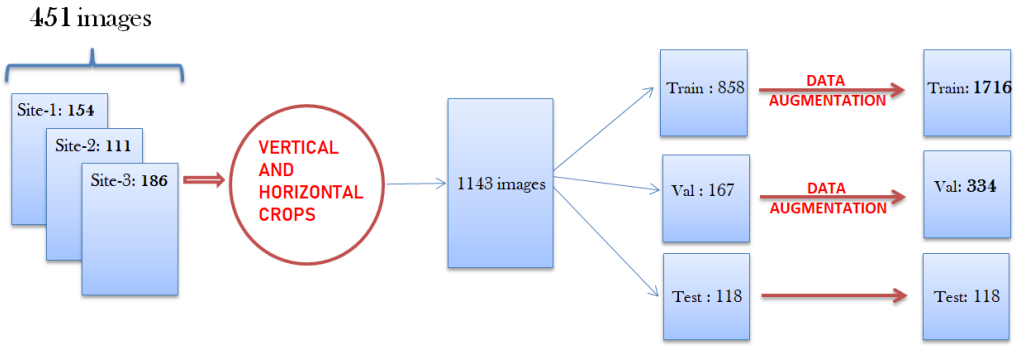


Figure 3.25: Data-set creation with data augmentation step by step and indication of the number of images of each data-set

Multiclass, Data Augmentation	Average	Background only	Mesh only	Wire only
Mean IoU	**70.7%**	81.1%	82.7%	48.3%
Dice coeff.	**81.7%**	89.5%	90.6%	65.2%
Pixel accuracy	**79.6%**	90.6%	90.7%	57.5%

Table 3.7: Multiclass-Semantic Segmentation experiment with data augmentation employed dataset (loss: sparse categorical cross entropy, Optimizer: Adam(lr=0.001)

Table 3.7 reports the results of multi-class U-Net model on data-set that is obtained with data augmentation. It is obvious that the performance of the model is increased. However still not as good as binary semantic segmentation for U-Net.

Figure 3.26 shows the prediction that is made by multiclass U-Net on data augmentation employed data.

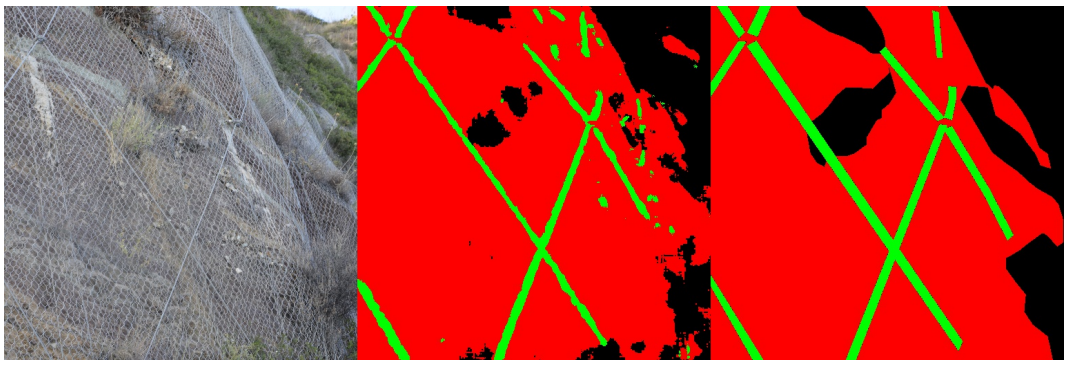


Figure 3.26: A sample of prediction that is trained by U-Net model on input images extended by data augmentation for multi-class semantic segmentation

In this multi-class semantic segmentation experiment, the goal was detecting both mesh and wire in a single model implementation. However, the performance of the model decreased significantly (compared to binary semantic segmentation application), especially in the detection of wire. Therefore, there is a compromise between employing a single model to detect both objects at pixel-level and accurately obtaining the segmentation results. As this thesis aims to identify an optimal model with high evaluation scores on test set, it will continue to focus on the binary semantic segmentation approach.

Chapter 4

Evaluation

4.1 Grid Search

This section aims to find an optimal combination of parameters such as optimization and loss function for the previously used U-Net model by exploring different options. This investigation involves training the standard U-Net model and finding the best combination for both mesh and wire separately in terms of binary semantic segmentation. Once the reasonably good combination is found among the chosen parameters, the focus shifts to determine the best filter size and number of convolution blocks for the up-sampling and down-sampling blocks.

Trial of different loss functions and optimizers

Different combination of loss functions and optimizers are investigated in this section. For binary semantic segmentation most widely used functions are taken into consideration. For optimizer Adam and RMSprop are going to be considered with their default values and for loss functions Binary focal loss and Binary Cross Entropy loss will be taken into account with their default parameters.

Results on test set of U-Net model trained on wire images with combinations of Adam and RMSprop optimizers, and Binary-focal Loss and Binary Cross Entropy loss are shown in the table 4.1 for wire and for mesh it is shown in table 4.2.

Wire	Adam(lr=0.0001) + BCE	Adam(lr=0.0001) + BFL	RMSprop + BCE	RMSprop + BFL
Mean IoU	54.1%	55.3%	56.5%	**56.8%**
Dice coeff.	70.2%	71.2%	72.2%	72.4%
Pixel accuracy	97.1%	97.2%	97.2%	97.3%
Pixel accuracy wire-only	63.1%	65.2%	67.8%	65.8%

Table 4.1: Results on test set of U-Net model trained on wire images with combinations of Adam and RMSprop optimizers, and Binary-focal Loss and Binary Cross Entropy loss.

In the combination of Adam with both Binary Cross-Entropy and Binary Focal Loss, the U-Net model was fitted to the local minimum and thus the learning rate was adjusted to obtain the results. (In default learning rate = 0.001, adjusted form has learning rate = 0.0001)

In conclusion, this grid search to find optimal parameters among the ones that is chosen, has shown that the RMSprop + Binary Focal Loss gives best results for

Mesh	Adam + BCE(lr=0.0001)	Adam(lr=0.0001) + BFL	RMSprop + BCE	RMSprop + BFL
Mean IoU	**88.4%**	88.3%	87.6%	86.6%
Dice coeff.	93.8%	93.8%	93.4%	92.8%
Pixel accuracy	92.5%	92.6%	92.0%	91.1%
Pixel accuracy mesh-only	95.1%	94.1%	95.1%	95.5%

Table 4.2: Results on test set of U-Net model trained on mesh images with combinations of Adam and RMSprop optimizers, and Binary-focal Loss and Binary Cross Entropy loss.

wire among all others. On the other hand, for mesh the best results are obtained with combination of Adam and Binary Cross entropy (with learning rate 0.0001).

Once the optimal optimizer and loss function have been selected for the model and data, the next step is to determine the best filter sizes. While the initial experiments with standard U-Net model were conducted using a starting filter size of 64, we are now examining the impact of filter size on model performance by testing with starting filter smaller filter size as base filter size is 32. This search will provide if the optimal filter size for the model is smaller.

Trial of different filter sizes for wire

Starting with the best parameters that are obtained by the previous experiments, the focus is turned into changing the number of filters. The obtained results on test set of U-Net model trained on wire images with RMSprop optimizer and Binary Focal Loss with base filter size 32 and the model that gave the best result so far for wire ımages during grid search previously(with base filter size 64) are reported in Table 4.3

Wire	RMSprop + BFL (beginning fs:32)	RMSprop + BFL(beginning fs:64)
Mean IoU	54.5%	**56.8%**
Dice coeff.	70.6%	72.4%
Pixel accuracy	97.1%	97.3%
Pixel accuracy wire-only	63.7%	65.8%

Table 4.3: Different filter sizes (fs) comparison on wire images for U-Net model with base filter size 32 and 64.

This search shows that bigger filter sizes are more suitable to obtain better results on test sets for wire images.

The obtained results on test set of U-Net model trained on mesh images with Adam optimizer and Binary Cross Entropy (best optimizer loss function combination on mesh images during grid search so far) with base filter size 32 and the previous model with base filter size 64 are reported in Table 4.4

Mesh	Adam + BCE(lr=0.0001) (filter size starts with 32)	Adam + BCE(lr=0.0001) (filter size starts with 64)
Mean IoU	87.7%	**88.4%**
Dice coeff.	93.4%	93.8%
Pixel accuracy	92.1%	92.5%
Pixel accuracy mesh-only	94.3%	95.0%

Table 4.4: Different filter sizes comparison on mesh images for U-Net model with base filter size 32 and 64.

This search shows that bigger filter sizes are more suitable also for obtaining better results on mesh images.

Experimenting with varying numbers of up-sampling and down-sampling layers.

After having identified the best optimizer, the best loss function, and the base filter size among the chosen parameters, the focus now turns on examining how the performance is being affected by different number of the up-sampling and down-sampling blocks in U-Net model.

In this experiment original U-Net architecture is compares with smaller one with one encoder and one decoder block eliminated from both sides of the network. For the parameter selection the best results given parameters are selected and indicated in the Table 4.5 with the all results for wire and results for mesh images are indicated in table 4.6

Wire	RMSprop + BFL **smaller**	RMSprop + BFL **standard**
Mean IoU	54.3%	**56.8%**
Dice coeff.	70.4%	72.4%
Pixel accuracy	97.2%	97.3%
Pixel accuracy wire only	61.0%	65.8%

Table 4.5: Different size of encoder decoder blocks are compared on wire images for U-Net model. 3 encoder blocks - 3 decoder blocks are reported in the left and standard U-Net model with 4 encoder - 4 decoder in the right.

Mesh	Adam + BCE **smaller**	Adam + BCE **standard**
Mean IoU	**89.1%**	88.4%
Dice coeff.	94.3%	93.8%
Pixel accuracy	93.0%	92.5%
Pixel accuracy mesh only	95.6%	95.1%

Table 4.6: Different size of encoder decoder blocks are compared on mesh images with the same configuration. Adam optimizer and Binary cross entropy loss with learning rate=0.0001 is used for both settings.

The experiment shows that smaller encoder-decoder blocks results in less accurate solution for wire images. However for mesh images the result is the opposite. After obtaining all the results, the best model for wire is U-Net model with dropout

layers, Adam optimizer and binary cross entropy loss. For mesh the best result that is achieved is the U-Net model with smaller encoder decoder blocks, Adam optimizer Binary cross entropy loss with learning rate 0.0001.

Last experiment is employing data augmentation applied data-set on mesh images for best model that is obtained so far for mesh since it is certain that data augmentation always improved the performance. The results are reported in the table 4.7

Mesh Adam + BCE (lr=0.0001)	smaller enc-dec	smaller enc-dec + dropout	**smaller enc-dec + dropout + augmentation**
Mean IoU	89.1%	89.6%	**90.2%**
Dice coeff.	94.3%	94.5%	94.8%
Pixel accuracy	93.0%	93.4%	93.8%
Pixel accuracy mesh-only	95.6%	95.3%	95.1%

Table 4.7: Best mesh results trained on data augmentation employed mesh images by U-Net model with dropout layers, smaller encoder decoder blocks

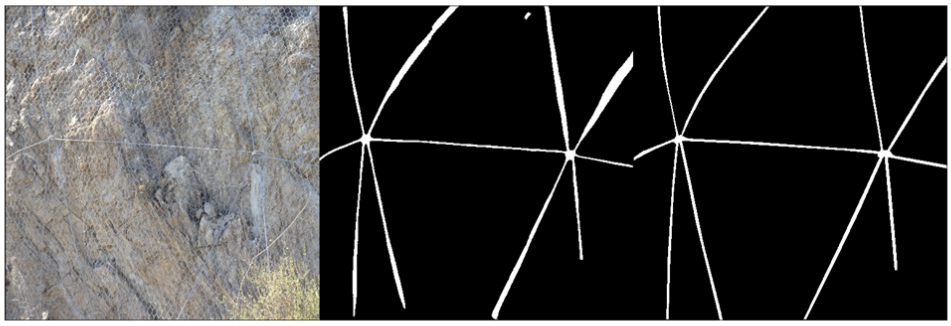
Finally, the best models for mesh and wire images that is obtained after all these experiments are reported in the table 4.8 and model selection phase is concluded with these results.

Best Model	**Best model for mesh**	**Best model for wire**
Mean IoU	**90.2%**	**58.1%**
Dice coeff.	94.8%	73.5%
Pixel accuracy	93.8%	97.3%
Pixel accuracy wire or mesh only	95.1%	68.2%

Table 4.8: Best model results for mesh and wire are reported.
Best model for mesh: smaller network with dropout layers on data augmentation employed data-set, optimizer: Adam, Loss: Binary Cross Entropy (learning rate = 0.0001).
Best model for wire: standard model with dropout layer on data augmentation employed data-set. Adam, Loss: Binary Cross Entropy (learning rate = 0.001).

To conclude this grid search section, it is worth to mention that 90.17% mIoU result on test set is an outstanding performance rate for a data that is manually labeled by humans because manually labeled data can vary depending on human annotators expertise, complexity of objects in an image, etc. This situation is valid also for wire images. Almost 60% of mIoU ratio is very good for smaller objects in an image (in this case wire objects in the input images have smaller pixel values compared to meshes). Smaller objects are more challenging to detect or segment accurately due to the fact that smaller objects may involve less distinct boundaries and/or may be overlapped by other objects in an image and this may lead unexpected predictions. It is harder to detect and segment the smaller object pixels for computer vision models. Therefore it negatively affects the performance. The best models predictions on mesh and wire images are shown in the figure 4.1

a. A sample of mesh prediction by smaller encoder-decoder block U-Net with dropout layers on data augmentation applied wire images



b. A sample of wire prediction by standard U-Net with dropout layers on data augmentation applied wire images

Figure 4.1: A sample of prediction that is trained by **best** U-Net models on input images for mesh and wire. Input image in the left, prediction in the center and target mask in the right.

Chapter 5

Conclusion

In conclusion, this thesis concentrated on analyzing wire and mesh images at the pixel level for the detection of the landslide containment components. The research collected a data-set of landslide containment device images from various locations in Sicily and conducted various pre-processing techniques on it. The Unet and Unet++ models were trained on the data-set, and binary segmentation tasks were performed in order to do a model selection. Additionally multi-class semantic segmentation approach is also employed and obtained results are compared to do a model selection between multi-class and binary semantic segmentation model. The best model was determined by testing different optimization algorithms and loss function combinations. Further experiments were carried out by varying the filter sizes and using smaller encoder-decoder blocks to assess the impact on performance.

U-Net and U-Net ++ architectures are exploited and experiments are done on wire and mesh images separately. Even-though U-net++ model gave slightly better results, the standard U-Net model is selected due to the fact that it is a lighter weight and less complex network compared to U-Net++, and there is a slight improvement that is negligible considering the complexity of the networks. Multi-class semantic segmentation gave poorer results as it is expected because conducting a binary segmentation is less complex task for models to conduct. Considering the compromise between employing a single model to detect both objects at pixel-level at the same time and accurately obtaining the segmentation results, model selection is decided on Binary U-Net model. After model selection, the focus turned on the grid search to with different combination of hyper-parameters. During grid search different loss functions, optimizers, different number of encoder decoder blocks and different filter sizes are employed. The results showed that the best model for wire is U-Net with dropout layers, Adam optimizer and binary cross entropy loss. The best model for mesh is achieved by U-Net with smaller encoder decoder blocks, Adam optimizer and binary cross entropy loss with learning rate 0.0001 on data augmentation applied data-set.

Possible future development is using the best model with best parameters to identify the objects in landslide containment devices and applying an anomaly detection technique to detect abnormal regions accordingly. This process generally focuses on analyzing the shape, texture, color and other features of the region of interest of the objects, and making a comparison with the normal state of the objects to see the deviation from the normal state. This can be done by determining a predefined thresholds to do inference whether an anomaly is crucial for further

investigation and to decide if it needs a repair. Thus, achieving robust and accurate object anomaly detection is possible by combining semantic segmentation with anomaly detection techniques. Thereby, harmful damages by landslide hazards or rockfall situations can be prevented.

Bibliography

[1] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "Segnet: A deep convolutional encoder-decoder architecture for image segmentation". In: *IEEE transactions on pattern analysis and machine intelligence* 39.12 (2017), pp. 2481–2495.

[2] B Basavaprasad and M Ravi. "A comparative study on classification of image segmentation methods with a focus on graph based techniques". In: *International Journal of Research in Engineering and Technology* 3.03 (2014), pp. 310–314.

[3] E. Blanco-Fernandez et al. "Flexible systems anchored to the ground for slope stabilisation: Critical review of existing design methods". In: *Engineering Geology* 122.3 (2011), pp. 129–145. ISSN: 0013-7952. DOI: https://doi.org/10.1016/j.enggeo.2011.05.014. URL: https://www.sciencedirect.com/science/article/pii/S0013795211001712.

[4] Elena Blanco-Fernandez et al. "Field measurements of anchored flexible systems for slope stabilisation: evidence of passive behaviour". In: *Engineering Geology* 153 (2013), pp. 95–104.

[5] Gabriel J Brostow, Julien Fauqueur, and Roberto Cipolla. "Semantic object classes in video: A high-definition ground truth database". In: *Pattern Recognition Letters* 30.2 (2009), pp. 88–97.

[6] Francois Chollet. *Deep learning with Python*. Simon and Schuster, 2021.

[7] D. M. Cruden. "A simple definition of a landslide". In: *Bulletin of the International Association of Engineering Geology - Bulletin de l'Association Internationale de Géologie de l'Ingénieur* 43 (1991). ISSN: 1435-9537. DOI: 10.1007/BF02590167. URL: https://doi.org/10.1007/BF02590167.

[8] Omid Ghorbanzadeh et al. "Evaluation of Different Machine Learning Methods and Deep-Learning Convolutional Neural Networks for Landslide Detection". In: *Remote Sensing* 11.2 (2019). ISSN: 2072-4292. DOI: 10.3390/rs11020196. URL: https://www.mdpi.com/2072-4292/11/2/196.

[9] Fausto Guzzetti et al. "Landslide inventory maps: New tools for an old problem". In: *Earth-Science Reviews* 112.1 (2012), pp. 42–66. ISSN: 0012-8252. DOI: https://doi.org/10.1016/j.earscirev.2012.02.001. URL: https://www.sciencedirect.com/science/article/pii/S0012825212000128.

[10] Shijie Hao, Yuan Zhou, and Yanrong Guo. "A brief survey on semantic segmentation with deep learning". In: *Neurocomputing* 406 (2020), pp. 302–321.

[11] Alexander Kirillov et al. “Panoptic segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 9404–9413.

[12] Scott Krig. *Computer vision metrics: Survey, taxonomy, and analysis*. Springer nature, 2014.

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[14] Y. Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. “Fully convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[16] Warren S McCulloch and Walter Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133.

[17] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-net: Convolutional networks for biomedical image segmentation”. In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pp. 234–241.

[18] Frank Rosenblatt. “The perceptron: a probabilistic model for information storage and organization in the brain.” In: *Psychological review* 65.6 (1958), p. 386.

[19] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[20] Karen Simonyan and Andrew Zisserman. “Very deep convolutional networks for large-scale image recognition”. In: *arXiv preprint arXiv:1409.1556* (2014).

[21] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. “Sun rgb-d: A rgb-d scene understanding benchmark suite”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 567–576.

[22] Kentaro Wada. *labelme: Image Polygonal Annotation with Python*. https://github.com/wkentaro/labelme. 2018.

[23] Zongwei Zhou et al. “Unet++: A nested u-net architecture for medical image segmentation”. In: *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 2018, pp. 3–11.