Master Thesis

Semantic Segmentation on Landslide Containment Devices

Selen Akkaya

Supervised by
Bartolomeo Montrucchio
Relatore 2

Final Report per la Tesi
Master in ...

Department of Control and Computer Engineering
Politecnico di Torino
Stato, Città
Mese 2023

Semantic Segmentation for Landslide Containment Devices

Sotto Titolo Tesi

Selen Akkaya

Relatori:

Supervisor 1
Relatore 1 Dipartimento Relatore
Supervisor 2
Relatore 2 Dipartimento Relatore

Abstract

This thesis presents a study on the application of semantic segmentation models. The research examines the data sets utilized, as well as the necessary pre-processing steps to prepare the data for model training. 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. Additionally, a grid search is conducted to find the best suitable parameters for the employed models. 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.

Dedication

Acknowledgements

Contents

1 Introduction **17**

2 Background **21**
2.1 Artificial Intelligence . . . 21
2.2 Machine Learning . . . 21
2.3 Deep Learning . . . 22
2.4 Neural Networks . . . 22
2.4.1 The History of Neural Networks . . . 24
2.4.2 Convolutional Neural Network (CNN) . . . 25
2.4.3 Encoder Decoder Structure . . . 27
2.5 Semantic Segmentation . . . 29
2.5.1 Comparisons of different computer vision tasks . . . 30
2.5.2 Traditional Computer Vision-based Methods for Semantic Segmentation . . . 31
2.5.3 Deep Learning-based Methods for Semantic Segmentation . . . 32
2.5.4 FCN for Semantic Segmentation . . . 33
2.5.5 SegNet . . . 34
2.5.6 U-Net . . . 36
2.5.7 U-Net ++ . . . 37
2.6 Evaluation Metrics . . . 39

3 Methodology **41**
3.1 Data processing for Semantic Segmentation . . . 41
3.1.1 Data collection and labeling . . . 41
3.1.2 Image Pre-processing . . . 42
3.1.3 Data Splitting . . . 43
3.1.4 Data Augmentation . . . 44
3.2 Experiments with U-Net in terms of Binary Semantic Segmentation for mesh and wire . . . 44
3.2.1 U-Net applied on mesh and wire images without employing data augmentation . . . 45
3.2.2 U-Net trained on mesh and wire images, data augmentation applied . . . 46
3.2.3 Gray scale images only . . . 47
3.3 U-Net++ for binary semantic segmentation on mesh and wire images 48
3.3.1 U-Net++ without data augmentation . . . 49
3.3.2 U-Net++ applied on mesh and wire images, data augmentation employed . . . 50

3.4 Multi-class Semantic Segmentation with U-Net 51

4 Evaluation **55**
4.1 Grid Search 55

5 Conclusion **61**

A Appendix **62**

List of Figures

1.1 Different types of spike plates from our data-set. 19
1.2 Composition of the components of Flexible Slope Stabilization System 19
1.3 Composition from a broad perspective. 20

2.1 Example of deep representations that is learned by a digit classification model[6]. 22
2.2 Illustration of a neural network that is parameterized by its weights and calculation of loss score to be used as a feedback signal to adjust the weights[6]. 23
2.3 An illustration of convolution operation[6] 26
2.4 ReLu and Sigmoid functions respectively[6]. 28
2.5 An illustration of several computer vision tasks [11] [10] 30
2.6 Representation of pixel-based, edge-based, graph-based image segmentation [2] 32
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]. . 33
2.8 Combining information from layers with different strides in a fully convolutional network and observation of enhanced accuracy of segmentation[15]. 34
2.9 Illustration of SegNet architecture [1] 35
2.10 Illustration of U-net architecture [17] 37
2.11 High-level overview of Unet++[23] 38
2.12 Auxiliary image to define metrics. 39

3.1 An example of semantic segmentation annotations done by LabelMe[22]. 42
3.2 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. 42
3.3 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". 43
3.4 An example of the original image (left) and the augmented image (right). 44

3.5 A sample of mesh prediction that is trained by U-Net model on input images. Input image in the left, prediction in the center and target mask in the right. 45
3.6 A sample of wire prediction that is trained by U-Net model on input images. Input image in the left, prediction in the center and target mask in the right. 46
3.7 A sample of mesh prediction that is trained by U-Net model on input images that are elaborated with data augmentation techniques. Input image in the left, prediction in the center and target mask in the right. 47
3.8 A sample of wire prediction that is trained by U-Net model on input images that are elaborated with data augmentation techniques. Input image in the left, prediction in the center and target mask in the right. 47
3.9 A sample of mesh prediction trained by U-Net on gray-scale input images. Input image in the left, prediction in the center and target mask in the right. 48
3.10 A sample of mesh prediction that is trained by U-Net++ model on input images. Input image in the left, prediction in the center and target mask in the right. 50
3.11 A sample of wire prediction that is trained by U-Net++ model on input images. Input image in the left, prediction in the center and target mask in the right. 50
3.12 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. 51
3.13 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. 51
3.14 Pixel labeling demonstration. Original image in the left, ground truth mask in the center and corresponding pixel label in the right. 52
3.15 Original image in the left and multi-class mask in the right. 52
3.16 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. 54

List of Tables

3.1 Results on test set: mesh . . . 45
3.2 Results on test set: wire . . . 46
3.3 Results on test set: mesh . . . 46
3.4 Results on test set: wire . . . 47
3.5 Results on test set: gray scale mesh images . . . 48
3.6 U-Net++ results on test set: mesh . . . 49
3.7 U-Net++ results on test set: wire . . . 49
3.8 U-Net++ results on test set: mesh . . . 50
3.9 U-Net++ results on test set: wire . . . 51
3.10 U-Net results on test set . . . 53

4.1 Test results of Adam, Binary Focal Loss . . . 55
4.2 Test results of Adam, Binary Cross Entropy . . . 55
4.3 Test results of Adadelta, Binary Focal Loss . . . 56
4.4 Test results of Adadelta, Binary Cross Entropy . . . 56
4.5 Test results of rmsprop, Binary Focal Loss . . . 56
4.6 Test results of rmsprop, Binary Cross Entropy . . . 56
4.7 Test results of Adam, Binary Focal Loss . . . 56
4.8 Test results of Adam, Binary Cross Entropy . . . 57
4.9 Test results of Adadelta, Binary Focal Loss . . . 57
4.10 Test results of Adadelta, Binary Cross Entropy . . . 57
4.11 Test results of SGD, Binary Cross Entropy . . . 57
4.12 Test results of adagrad, Binary Cross Entropy . . . 58
4.13 Test results of rmsprop, Binary Focal Loss . . . 58
4.14 Test results of rmsprop, Binary Cross Entropy . . . 58
4.15 Test results of unet model trained on wire images with base filter size 4 58
4.16 Test results of unet model trained on wire images with base filter size 16 . . . 59
4.17 Test results of unet model trained on mesh images with base filter size 4 59
4.18 Test results of unet model trained on mesh images with base filter size 16 . . . 59
4.19 Test results of unet model trained on mesh images with smaller encoder decoder blocks . . . 60
4.20 Test results of unet model trained on wire images with smaller encoder decoder blocks . . . 60

A.1 Morbi ac varius enim, ac lobortis odio. Nullam venenatis, erat in faucibus vestibulum, magna dolor faucibus sapien, ac iaculis tellus urna ut purus. Pellentesque finibus urna eget maximus cursus. Aenean mollis ante nec dolor iaculis, ac malesuada enim ultrices. Aenean vulputate felis sapien, quis rhoncus odio dictum vitae. 62

List of Contributions

- Contribution 1;
- Contribution 2;
- Contribution 3.

Chapter 1

Introduction

For the purpose of predictive maintenance of landslide containment devices and preventing crucial landslide hazards, it can be helpful to detect components of Flexible Slope Stabilization Systems. However detecting the damaged areas or any kind of anomalies on landslide containment devices is a challenging process by labor force. Pixel-level component detection can be done by deep learning models that are suitable for image analysis tasks, such as semantic segmentation models.

The objective of this thesis is to implement the most recent semantic segmentation models to assign a class label to each pixel in images to detect mesh and wire in landslide containment devices. The goal is to evaluate the performance of the models based on accuracy and pixel-level measurements. This will enable the selection of the most reliable model for image analysis, which will be beneficial for maintaining Flexible Slope Stabilization Systems. This thesis is developed in cooperation with Modelway S.r.l.

The outcome will be the state-of-the-art semantic segmentation model which can be used for images analysis of the Flexible Slope Stabilization System components.

Semantic segmentation algorithms are useful in comprehending the context of an environment, and therefore, they are extensively employed when the context of the environment is essential.

Semantic segmentation is a crucial area of Computer Vision, which seeks to label each pixel in an image with a specific class. This approach enables image classification at a pixel-level, and the labels assigned could represent objects such as glass, rock, road, house, and living beings such as humans, cats, dogs, etc. In essence, the labels represent general patterns observed in the image.

The landslide containment devices considered in this study are composed of two types of spikes(that are customarily named associated with their respective types as "spike red" and "spike ring"), mesh and wire. This thesis focuses on detecting two main components, which are customarily referred to as "wire" and "mesh".

Landslides

Mass movements such as landslides are a crucial natural hazards that threat to natural resources, human lives, infrastructures, and properties in mountainous regions all over 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 Slope Stabilization System

The flexible systems used in slope stabilisation are formed by a covering (cable net or high-resistance wire mesh) and anchored bolts. This technique has spread substantially due to the fact that it has low visual impact and minimal influence on traffic during installation[3].

To enhance the understanding of landslide containment structures, 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.

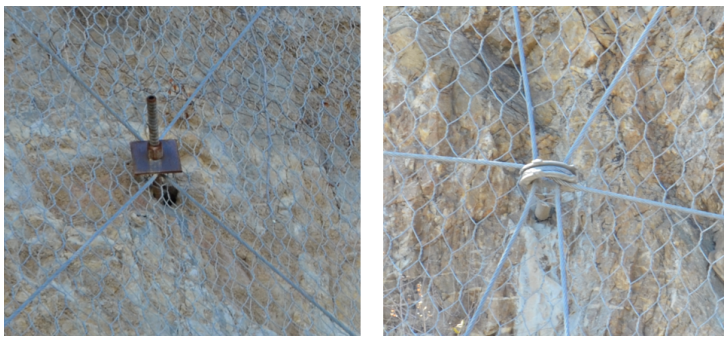


Figure 1.1: Different types of spike plates from our data-set.

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 background information with respect to semantic segmentation, and the development of Neural Networks, the technical details of deep learning and its application in the semantic segmentation tasks. It covers the use of Convolutional Neural Networks (CNNs) and their variations particularly the state-of-art models that are used for Semantic Segmentation, such as SegNet, U-Net, U-Net++. Additionally, examination of the metrics and loss functions that are commonly used for semantic segmentation is explained such as Dice coefficient, Intersection over Union and pixel accuracy.

2.1 Artificial Intelligence

To be able to successfully do pixel-level component detection, it is essential to have a fundamental knowledge of artificial intelligence (AI), machine learning, and deep learning. Artificial Intelligence (AI) is a field of computer science that aims to create machines that can perform tasks that would typically require human intelligence, such as understanding natural language, recognizing patterns, and making decisions. It encompasses various techniques, including machine learning, deep learning, and symbolic reasoning, to create systems that can perform tasks that were once thought to be exclusive to 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].

Supervised learning is a type of machine learning in which a model is trained on a labeled data-set. The goal of the training process is to learn a mapping from input space to outputs, such that when the model is presented with new, unseen inputs, it can predict the correct corresponding output. Supervised learning is widely used in image classification.

In unsupervised learning, the model is given a data-set of unlabeled examples, and the goal is to learn some underlying structure or distribution in the data. Unsupervised learning is widely used in areas such as computer vision, where there is a wealth of unlabeled data 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

A Neural Network is a machine learning model that is based on the structure and function of the human brain. It is composed of interconnected processing units, known as artificial neurons, which are organized into layers. These layers process input data and generate 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 underlying relationships between the input and output. An example of layer representation is shown in Figure 2.1.

Neural networks can be used for a wide range of tasks, including image classification, natural language processing, and speech recognition. The architecture of neural networks can be divided into feedforward, convolutional, and recurrent neural networks, each of which serves a specific purpose and is suitable for different types of tasks[6].

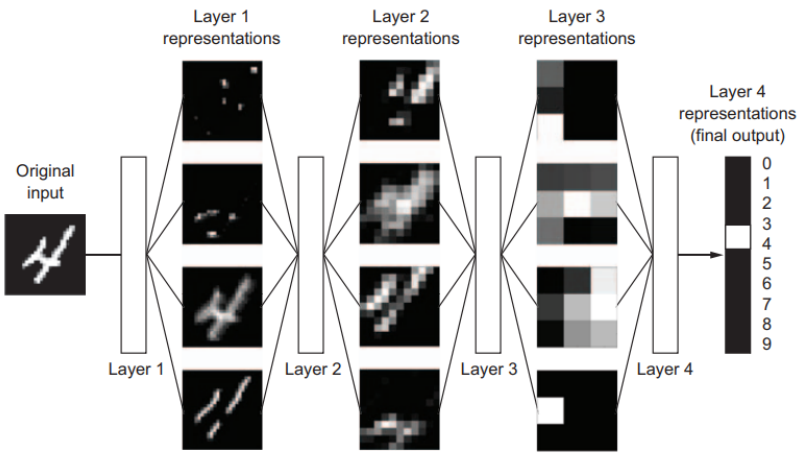


Figure 2.1: Example of deep representations that is learned by a digit classification model[6].

Neural network includes layers that can be broadly categorized into two types as input layers and hidden layers.

Input layers are the first layer of the network, and their main function is to receive the input data and pass that data to the next layer.

The next layers is called hidden layers and their main role is extracting features from the input data. These layers typically consist of multiple artificial neurons, and they are connected to each other. They are responsible with the non-linear transformation of the input data. Additionally, these layers are called hidden layers because they are not directly exposed to the input or output of the data.

Output layers are the last layer in a network, and their main role is generating the final output of a network. Output layer takes the features that are extracted by the hidden layers and use these features to make a decision or prediction.

The number of layers and the number of neurons in each layer can differ depend on the task and the data complexity. The more layers and neurons, the more complex the model is and it needs more data to be trained on.

Each layer of the network applies a set of parameters to the input data and they are also known as weights and biases. The result is passed to the next layer and this process is called as feed-forward.

During the training phase, the network adjusts these parameters for the purpose of minimizing the error that occurred between the predicted output and the ground truth. This process is called 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

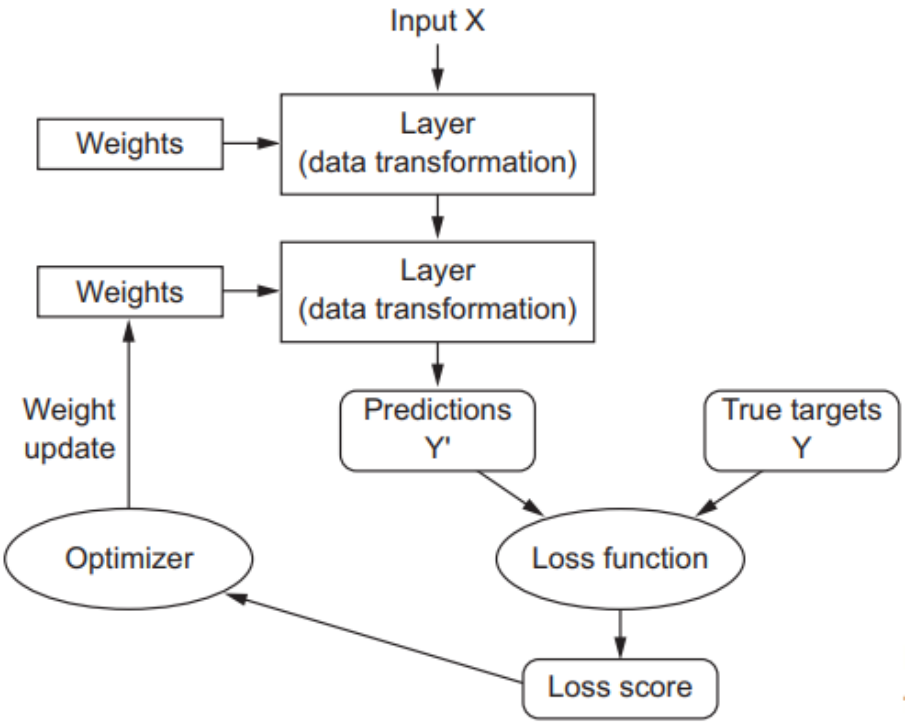


Figure 2.2: Illustration of a neural network that is parameterized by its weights and calculation of loss score to be used as a feedback signal to adjust the weights[6].

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 evaluation of the error between the predicted output and the ground truth known as the loss score. It is used as a feedback mechanism to fine-tune the weights of the network. In other words the calculation of the loss score is used as a feedback signal to adjust the weights in the network. The loss score is a measure for the network that shows how well the network is performing.

The goal of the training process is minimizing 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 Mean Squared Error (MSE) and Cross-Entropy Loss.

The training process starts with feeding the network with the input data, then the weights of the network is used to make calculations and so generate predictions. The predicted output is then compared to the true output by a loss function. This comparison provides a loss score. The loss score is then used as a feedback signal and adjust the weights by using an optimization algorithm (as Stochastic Gradient Descent (SGD) or Adam). This process is being repeated over and over again, until the network reaches a adequate level of accuracy.

2.4.1 The History of Neural Networks

The history of neural networks dates back to the 1940s and 1950s, when researchers first began exploring the idea of using artificial networks of simple elements, or "neurons," to model the behavior of biological neurons. The first concept of an artificial neural network was introduced by Warren McCulloch and Walter Pitts in 1943, in which they proposed a simple mathematical model of a biological neuron that could be used to perform simple logical operations[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.

Overall, the history of neural networks has seen the development of increasingly complex and powerful models, as well as new training algorithms, which have made it possible to apply neural networks to a wide range of real-world problems.

2.4.2 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a type of neural network which is particularly well-suited for analysis of image and video tasks. CNNs are inspired by the visual system in animals and are designed to process data with a grid-like topology, such as an image.

CNNs are composed of multiple layers that 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 the 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. It allows the network to focus on the significant features and to reduce the dimensionality of the data.

Fully connected layers, also known as dense layers, are used to make predictions (or decisions) based on the extracted features of the convolutional and pooling layers.

CNNs are able to learn hierarchical representations of images. The learned features can be used for a wide range of image analysis tasks, such as image classification, object detection, and semantic segmentation. CNNs have been used to obtain state-of-the-art performance on many tasks of computer vision.

Convolution Operation

The convolution operation is a mathematical operation used in CNNs to extract features from the input data. The convolution operation is performed between the input data, such as an image, and a kernel that is a small matrix and also called as filter.

The input data is scanned by the filter. 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 feature map element 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, textures, or patterns.

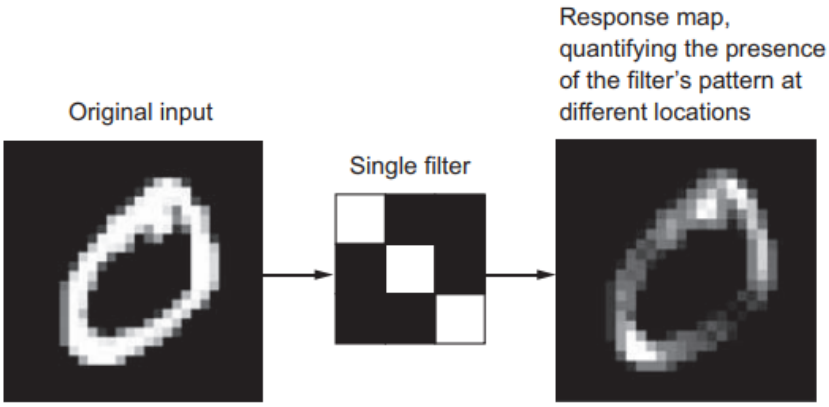


Figure 2.3: An illustration of convolution operation[6]

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.

Convolution Layers

Convolutional layers are the core building block of Convolutional Neural Networks, and they are used to extract features from the input data, that 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 the filters, 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, producing a new output matrix known as a feature map. The important point is that each neuron in the feature map is only connected to a small and localized region of the input data that is defined by the size and stride of the kernel. This local connectivity allows the network extracting features from the input data. This 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 non-linear layer in a neural network is a type of layer that applies a non-linear function to the input, allowing the network to model complex, non-linear relation-

ships in the data. The rectified linear unit (ReLU) function is the most commonly used non-linear function in neural networks.

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.

Relu is typically used 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 2.4. Additionally, softmax is another popular activation function that is used in final layer in multi-class classifications.

Pooling Layers

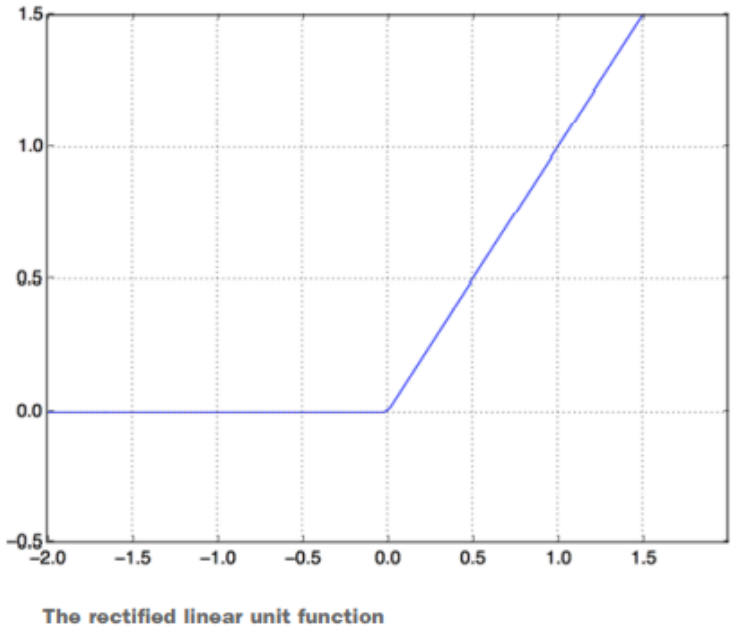
A pooling layer in a convolutional neural network (CNN) is a type of layer that performs down-sampling operations on the input data. The primary goal of pooling layer is to reduce data dimensionality and make the network more resistant to small translations in the input. The most common types of pooling operations are 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 computed and output 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. 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, is a type of layer utilized in neural networks. These layers are referred to as dense as they are connected to every neuron in the preceding layer, instead of just a specific subset of neurons as seen in other architectures such as convolutional neural networks. In this layer, each neuron receives input 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. This output is then sent to the next layer in the network. Dense layers are commonly used in feed-forward neural networks and are employed for extracting features from the input data. In order to extract features from the input data, dense layers are typically positioned between the input and output layers. To build a deep neural network, which enables the extraction of high-level features from the input data, they can be stacked one on top of the other.

2.4.3 Encoder Decoder Structure

An encoder-decoder is a type of neural network architecture that is commonly used in computer vision tasks, particularly image and video analysis tasks. The encoder component of the network extracts features from the input image or video, while the



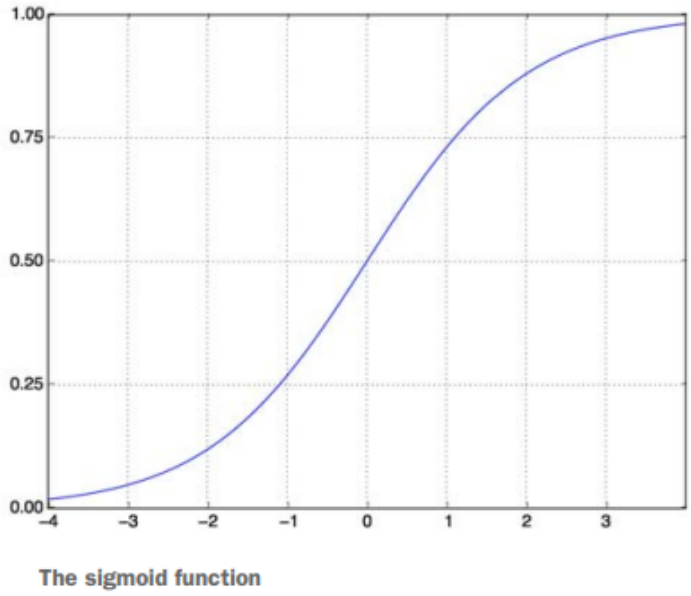


Figure 2.4: ReLu and Sigmoid functions respectively[6].

decoder component of the network uses those features to generate the output image or video. This architecture is significant because it enables the network to learn a compact representation of the input data, which can then be used to produce more accurate output. It is also used in tasks like image captioning, semantic segmentation, and object detection.

Encoder

An encoder architecture is typically composed of multiple layers of convolutional neural networks (CNNs) and/or recurrent neural networks (RNNs). These layers extract increasingly complex features from the data as the input image or video is passed 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 extract 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 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

A decoder's architecture is typically composed of multiple layers of transposed convolutional neural networks (CNNs). The decoder takes the encoder's compact, fixed-length feature vector as input and uses it 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 task of Computer Vision. Computer vision is the study of teaching computers to interpret and understand visual information from the world, like images and videos. It involves creating algorithms and methods for analyzing images and videos, identifying objects, and understanding the scene. The field draws on knowledge from image processing, computer graphics, machine learning, and artificial intelligence to develop these methods. The ultimate goal of computer vision is to build machines that can perceive and understand the visual world like humans do and apply this understanding to a wide range of applications such as self-driving cars, medical imaging, and security systems[12].

Semantic Segmentation is the process of assigning a semantic label, such as "road", "sky", or "car", to each pixel in an image. The goal is to create a dense prediction that classifies every pixel in the image and provides a complete representation of its semantic content.

This task differs from others in computer vision, such as Image Classification, Object Detection, and Instance Segmentation. 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 offers a more

fine-grained representation of the image compared 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 is a type of image segmentation that aims to classify 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 is a type of image segmentation that aims is to classify each pixel in an image as belonging to one of multiple classes. 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 more challenging than binary semantic segmentation since it requires the model to distinguish among many different classes, whereas binary semantic segmentation only requires the model to distinguish between two classes.

2.5.1 Comparisons of different computer vision tasks

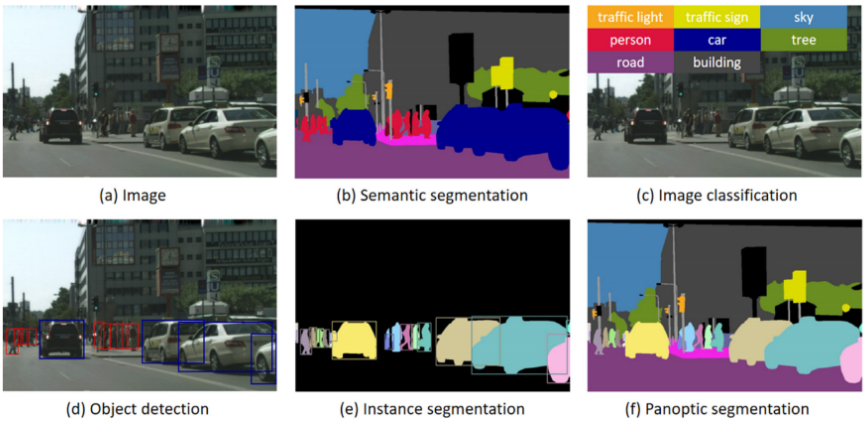


Figure 2.5: An illustration of several computer vision tasks [11] [10]

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 superpixel as a node and edges between neighboring pixels or superpixels. The goal is to partition the graph into different segments or regions, each of which corresponds to a different object or part of the scene.

Region-based methods, on the other hand, involve grouping pixels or superpixels that are similar in terms of color, texture, or other features. The goal is to segment the image into different regions, each of which corresponds to a different object or part of the scene.

Edge-based image segmentation relies on detecting edges or boundaries in the image to segment the image into different regions or objects. This approach is based on the idea that edges or boundaries in an image correspond to the transition between different regions or objects. They are used as the basis for segmenting the image into different regions or objects.

Pixel-based image segmentation relies on analyzing the properties of each pixel in the image to segment the image into different regions or objects. This approach is based on the idea that pixels in an image that are similar in terms of color, texture, or other features belong to the same region or object[2].

These methods are typically less accurate than deep learning-based methods and they can be easily affected by the changes in lighting, viewpoint, and occlusions, among other factors.

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.

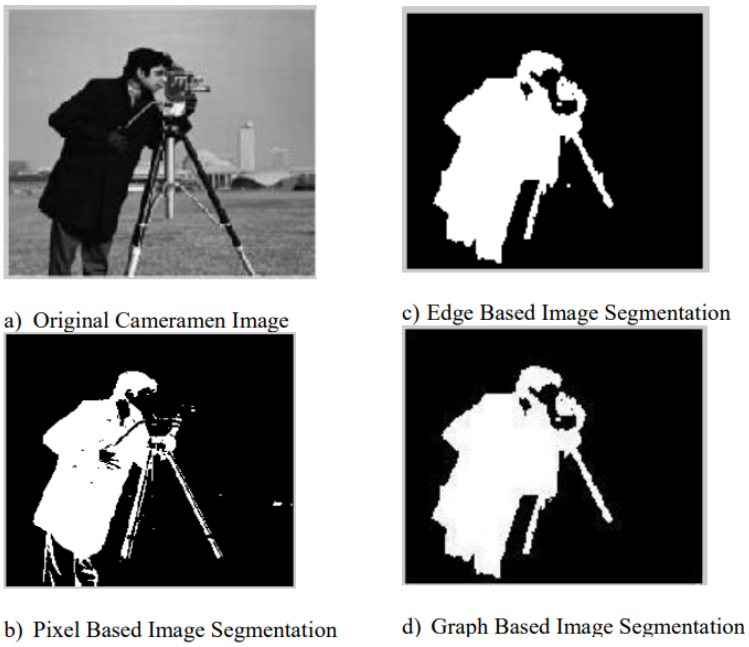


Figure 2.6: Representation of pixel-based, edge-based, graph-based image segmentation [2]

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. These methods are based on convolutional neural networks (CNNs).

There are several types of deep learning-based methods for semantic segmentation, including:

Fully convolutional networks (FCNs): These are a type of CNN that are able to take an input image of any size and produce an output segmentation map of the same size. FCNs are able to make use of the spatial information in the image by using a series of convolutional and pooling layers to extract features at different scales.

U-Net: This is a type of FCN that is particularly well-suited for biomedical image segmentation tasks, as it is able to accurately segment small structures in images. U-Net is composed of an encoder-decoder architecture, where the encoder is used to extract features from the image, and the decoder is used to reconstruct the segmentation map from the features.

These methods have shown great performance on several benchmarks and have been used in a wide range of applications, such as self-driving cars, robot navigation, image editing, medical imaging and scene understanding, among others. It's worth noting that these architectures continue to evolve and improve, with new techniques and architectures being proposed frequently.

2.5.4 FCN for Semantic Segmentation

A significant development in the field of image segmentation came in the form of Fully Convolutional Networks (FCNs), a variation of CNNs. 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.

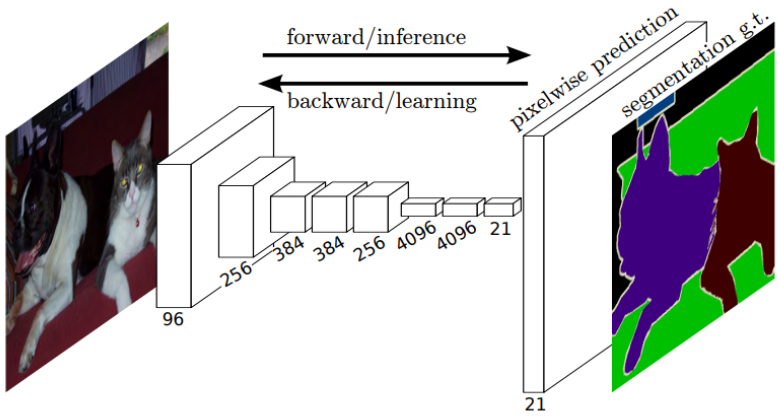


Figure 2.7: A Fully Convolutional Neural Network (FCN) make dense predictions for per-pixel tasks, allowing the network to take in inputs of arbitrary size and produce output with corresponding spatial dimensions[15].

A Fully Convolutional Network (FCN) is a type of convolutional neural network (CNN) that is designed to perform dense predictions over an entire image, rather than classifying a single patch or region of an image. This is achieved by replacing the fully connected layers at the end of a CNN with convolutional layers, allowing the network to take in inputs of arbitrary size and produce output with corresponding spatial dimensions. In other words, an FCN makes dense predictions over an image by processing the entire image at once, rather than processing small regions of the image at a time. This allows the network 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.

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

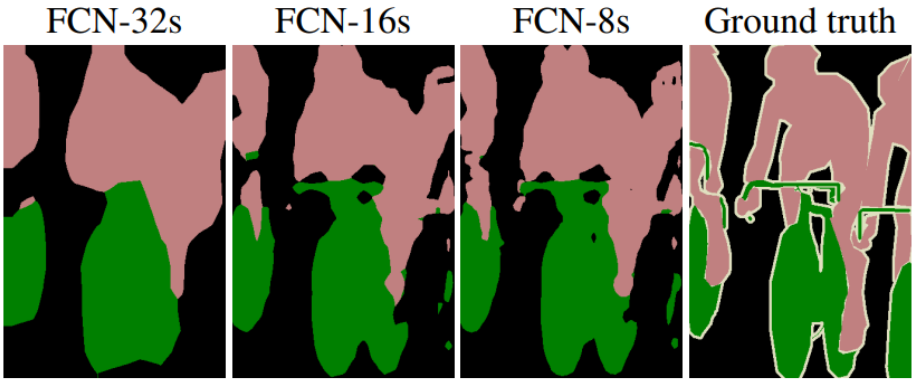


Figure 2.8: Combining information from layers with different strides in a fully convolutional network and observation of enhanced accuracy of segmentation[15].

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.

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

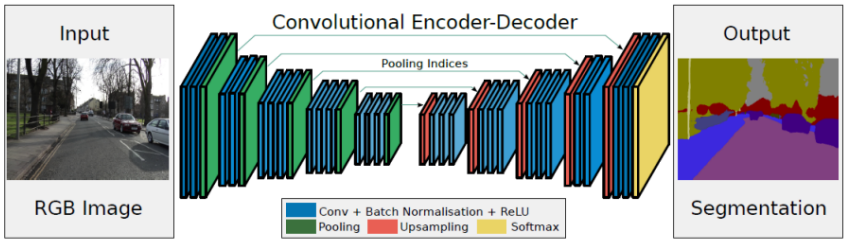


Figure 2.9: Illustration of SegNet architecture [1]

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 up-sampling the encoder network feature maps to the original picture resolution and creating 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 version of a fully convolutional network (FCN) and is commonly used for semantic segmentation tasks. The U-Net was first introduced by Ronneberger and Fischer in 2015 and it was specifically designed for image segmentation and localization in biomedical applications[17].

U-Nets have an advantage over regular CNNs because they can both identify where an object is in an image and what it is. Thus it provides 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 segmentations even with a limited number of training images. This is achieved by using up-sampling layers with a large number of feature channels, which helps to convey more context information to the higher resolution layers.

The U-Net segmentation network is the best choice for this application because it is a scalable and flexible Fully Convolutional Network (FCN) that doesn't compromise on localization and context information. Additionally, U-Nets are commonly used in current state-of-the-art techniques for semantic segmentation of satellite imagery.

U-Net segmentation networks are the most suitable choice for this application because they are highly scalable and can be easily adjusted to different sizes. They also provide a balance between localization and context preservation, and are commonly used in state-of-the-art techniques for semantic segmentation in general.

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.

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.

The U-Net model is able to capture the overall context of the image by using a contracting path, while at the same time it allows for precise localization 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

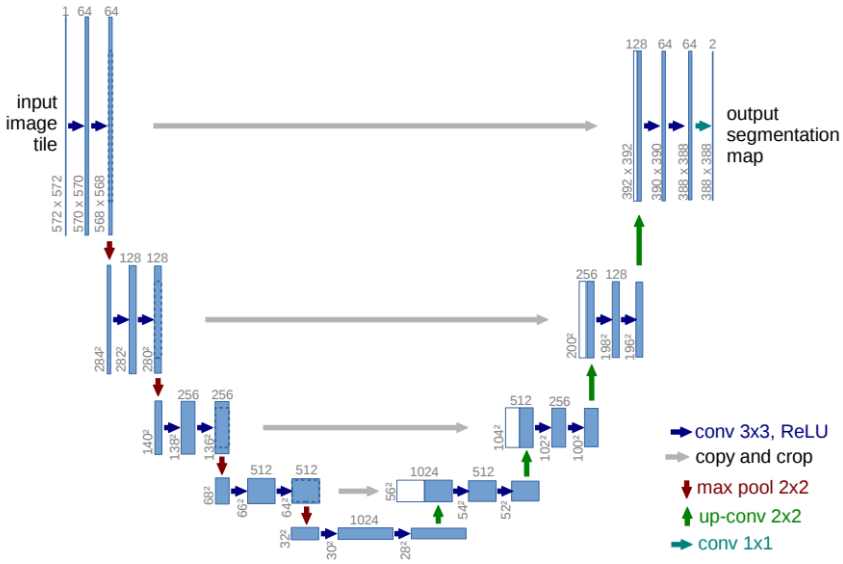


Figure 2.10: Illustration of U-net architecture [17]

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 specifically designed to minimize the difference between the feature maps generated by the encoder and decoder, thus improving the final output of the segmentation.

UNet++ uses a hierarchical architecture, where multiple levels of encoders and 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

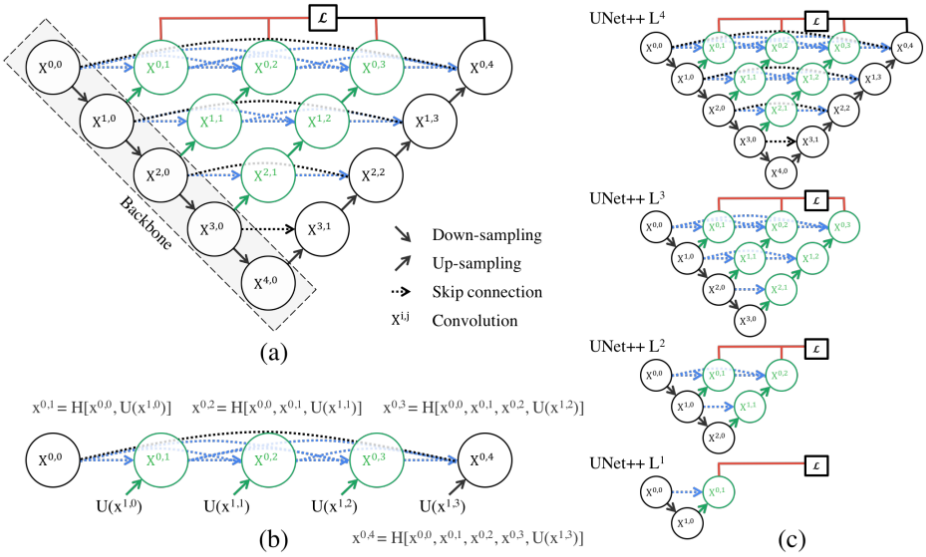


Figure 2.11: High-level overview of Unet++[23]

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 improved version of the UNet architecture that addresses some of its limitations and allows to achieve better performance in image segmentation tasks.

Difference Between UNet and UNet++

The main difference between UNet and UNet++ is in their architecture and the way they handle feature extraction and feature fusion.

UNet uses a single encoder-decoder structure, while UNet++ uses a hierarchical architecture with multiple levels of encoders and decoders.

UNet uses the same set of filters at all scales, while UNet++ uses different sets of filters at different scales.

UNet uses skip connections to combine features from different levels of the encoder, while UNet++ uses skip connections and attention gates to selectively propagate information from the encoder to the decoder.

UNet++ is designed to extract more contextual and fine details information than UNet.

UNet++ is designed to handle feature extraction and feature fusion in a more efficient and effective way than UNet.

UNet++ 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.

2.6 Evaluation Metrics

To assess the performance of various network architectures for semantic segmentation, a proper metric that accurately describes the network's ability to identify a class is required. Three metrics are employed in this study to evaluate models that are widely used semantic segmentation and that are mean Intersectıon over union, dice coefficient and pixel accuracy.

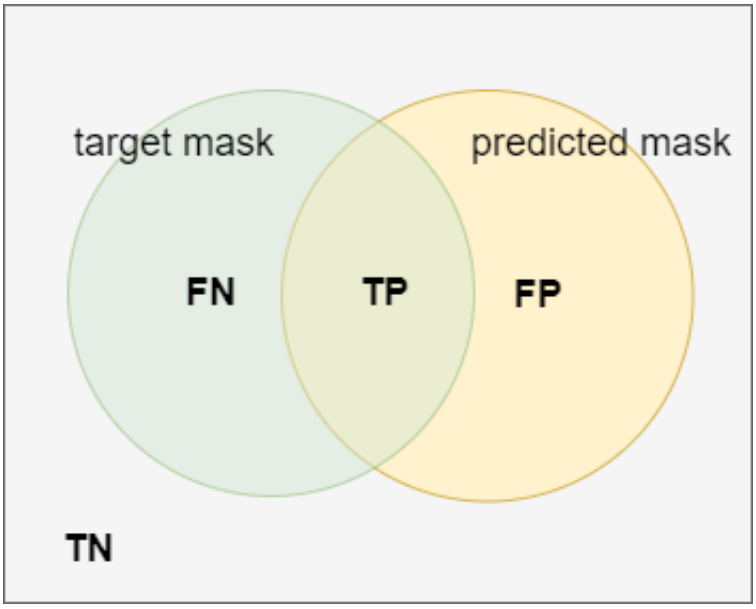


Figure 2.12: Auxiliary image to define metrics.

mIoU

mIoU stands for "mean Intersection over Union". It is a commonly used metric in computer vision, particularly for semantic segmentation tasks.

The Intersection over Union (IoU) is a metric that is commonly 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 intersection of the predicted and target masks and dividing it by the union of the two masks.

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 a similarity metric that can be used to compare the overlap between two sets of data, such as the predicted output of a semantic segmentation model and the ground truth segmentation. It has a value between 0 and 1, with 1 indicating that the two sets are identical and 0 indicating that the two sets have no elements in common. Therefore, values closer to 1 indicating 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, also known as overall accuracy, is a simple and widely used metric for assessing image classification and semantic segmentation model performance. It is defined as the proportion of correctly classified pixels to total pixels in an image.

It is a scalar value between 0 and 1, with 1 representing a perfect match and 0 representing no match at all. It is a very simple metric that only considers whether pixels are correctly classified or not, but it does not take into account class imbalance or the location of misclassifications.
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

The focus of this chapter is on the techniques employed in the research for performing semantic segmentation on mesh and wire. First, information on the data sets utilized in the study is examined and the pre-processing steps necessary to prepare the data and run the models effectively. Next section delves into the different model structures utilized, as well as the metrics and loss functions used to refine the models for comparison. The following section covers the procedure for evaluating the performance of the final models on test sets and determining their overall accuracy. Finally grid search is employed to find the best suitable parameters for the employed models.

3.1 Data processing for Semantic Segmentation

The preparation of data for use in training and evaluating semantic segmentation models is referred to as data processing for semantic segmentation. Semantic segmentation process typically includes gathering images and annotations for specific classes, preprocessing the data by cleaning and transforming it into a format suitable for use in the model, augmenting the data to increase diversity and prevent overfitting, and dividing the data into training, validation, and test sets. These steps are crucial for preparing the data in a format 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 involves acquiring the images and the annotations for each class of interest. To construct the dataset, roughly 500 high-resolution photos (4928 x 3264) were captured at three locations in Sicily. These photos depict rocky walls that are covered with meshes and, in many cases, also contains spikes and wires.

The annotations define the regions of interest in the images that belong to the class. Annotations can be done by annotation tools that have a user interface for manually annotate the data. In this study, the process of labeling was carried out using a Python library named *LabelMe* [22]. For each image in the dataset, the masks corresponding to wires and meshes were created using polygons. LabelMe is an open-source image annotation tool that is designed for image segmentation and object recognition purposes. With LabelMe, users can manually label and annotate

objects in images, creating segmentation masks and annotated data that can be used for computer vision model training such as semantic segmentation and object detection models. 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.

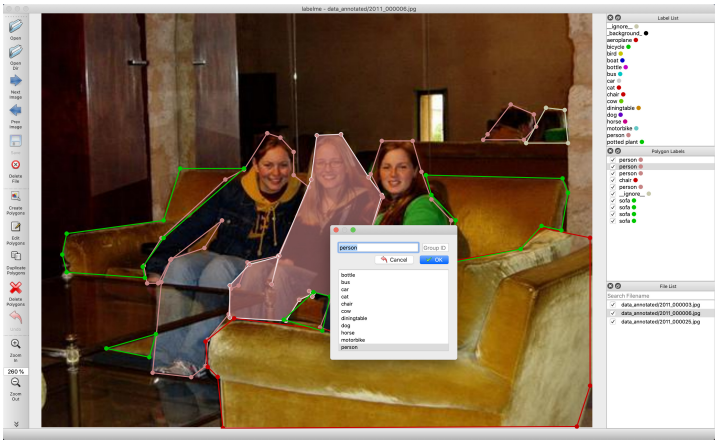


Figure 3.1: An example of semantic segmentation annotations done by LabelMe[22].

In this study, only wire and mesh are considered since they are more suitable for semantic segmentation. Figure 3.2 displays an image with annotations to demonstrate the process of annotation.

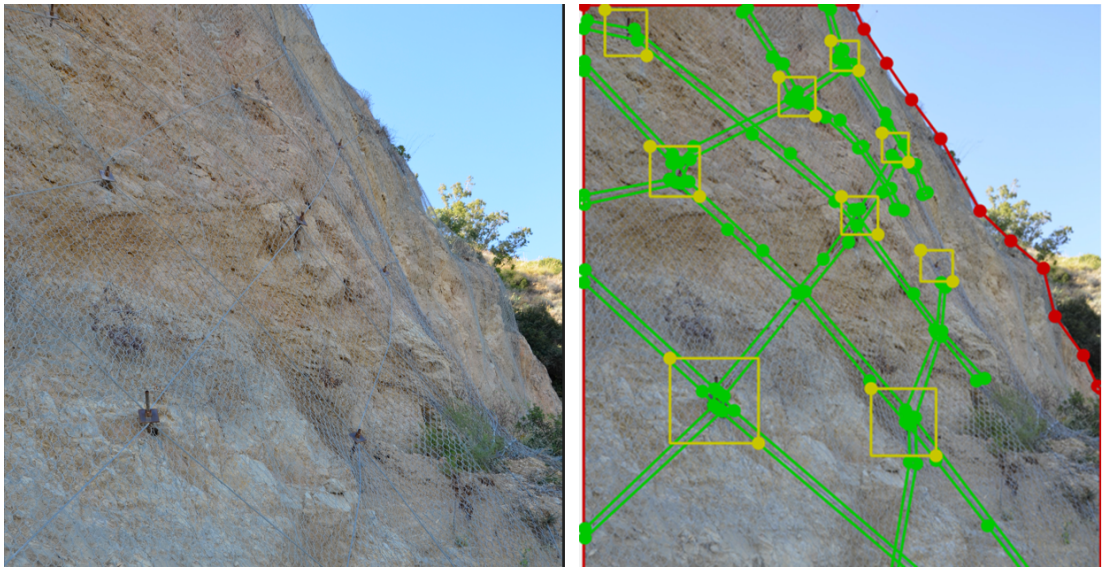


Figure 3.2: A demonstration of the labeling process performed using the LabelMe[22] annotation tool is given. Original image and annotated version is shown respectively. The yellow boxes indicate spikes(that are not considered in this thesis), the green polygons represent wires, and the red polygons are for mesh.

3.1.2 Image Pre-processing

Pre-processing includes cleaning and transforming the data into a suitable format for model to use. This can include image resizing, pixel values normalizing, and

encoding class labels to numerical values.
The acquired images were captured in both horizontal and vertical orientation. The models used necessitate a fixed input dimension, so to to create data-sets, three cuts were made for each horizontal image (left, center, right) and two cuts for each vertical image (bottom, center). The vertical images were limited to two crops because the top portions were often too distant and resulting in unclear visibility of the elements, likely because the images were taken from the ground. This operation is shown in the Figure 3.3. These cropped images were square in shape with a size of 3264x3264 pixels. The mesh and wire images are then resized as 512x512.

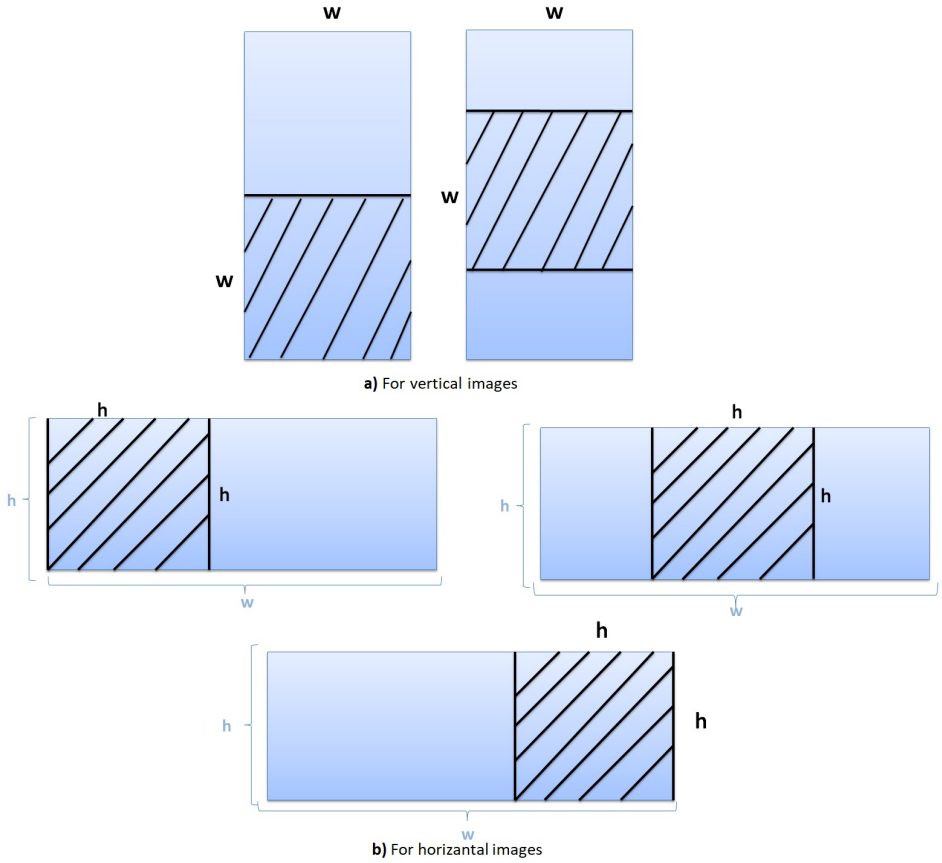


Figure 3.3: A demonstration is provided of the horizontal and vertical cuts applied to the input images. It shows the vertical cuts identified as "center" and "bottom" and the horizontal cuts identified as "center," "left," and "right".

3.1.3 Data Splitting

Splitting involves dividing the data as training, validation, and test sets. To train the model, the training set is used. For model selection, the validation set is used. Lastly, the test set is used to evaluate the performance of the model.
The data for meshes and wires were 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.

3.1.4 Data Augmentation

Data augmentation involves generating additional training data to training set by applying transformations to the existing images. This transformation can include flipping, rotation, scaling, adding random noise into images etc. Data augmentation helps preventing overfitting and increases the diversity of the training data.
Data augmentation was employed before training to assess its impact on accuracy, and the same method was used for both wires and meshes. 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 3.4 shows an example of data augmentation that is employed for the dataset.

Figure 3.4: An example of the original image (left) and the augmented image (right).

3.2 Experiments with U-Net in terms of Binary Semantic Segmentation for mesh and wire

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. 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. Output returns the predicted mask with the same size as the target. Image values are normalized to [0,1]. The neural network has a rather complex structure, which is divided into the downsampling, upsampling and classification parts. Therefore there are two paths: contracting path and expansive path. The downsampling consists of 6 blocks containing 2 convolutional layers followed by max pooling. The bottleneck layer is created after the contracting path, which is followed by the expansive path.In the expansive path, the feature maps are upsampled using transposed convolutions. The upsampled feature maps are then concatenated with their corresponding feature maps from the contracting path to preserve the spatial information. In the upsampling part there are 5 blocks, formed by 2 convolutional layers and one deconvolutional layer. The final part contains 3 convolutional layers, where the last layer as an activation function has the sigmoid acting on each pixel. So for each pixel 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.

3.2.1 U-Net applied on mesh and wire images without employing data augmentation

In this experiment, a dataset of images with a resolution of 512x512 pixels was used. Each image in the dataset contained mesh for binary semantic segmentation on mesh, and it contained wire for wire segmentation. The dataset was divided into three sets: a training set consisting of 854 images, a validation set consisting of 163 images, and a test set consisting of 120 images. The deep learning model was trained for 250 epochs using the 'adam' optimizer and a batch size of 32. The loss function used was binary cross-entropy, and early stopping was employed with a patience of 50. The mean Intersection over Union (IoU) on the validation set was used to monitor the early stopping process.

U-Net applied on mesh images

The obtained results on test set of U-Net model trained on mesh images are reported in Table 3.1 and a sample composition of original image, prediction and ground truth mask is shown in the Figure 3.5

Mean IoU	86,28%
Dice coefficient	92,63%
Pixel accuracy(all image)	91,01%
Pixel accuracy mesh only	94,46%

Table 3.1: Results on test set: mesh

Figure 3.5: A sample of mesh prediction that is trained by U-Net model on input images. Input image in the left, prediction in the center and target mask in the right.

U-Net applied on wire images

The obtained results on test set of U-Net model trained on wire images are reported in Table 3.2 and a sample composition of original image, prediction and ground truth mask is shown in the Figure 3.6

Mean IoU	49.56%
Dice coefficient	66.27%
Pixel accuracy(all image)	96.66%
Pixel accuracy wire only	61.04%

Table 3.2: Results on test set: wire

Figure 3.6: A sample of wire prediction that is trained by U-Net model on input images. Input image in the left, prediction in the center and target mask in the right.

3.2.2 U-Net trained on mesh and wire images, data augmentation applied

This experiment utilized a data augmentation employed dataset of images with a resolution of 512x512 pixels. Every image in the dataset had mesh for binary semantic segmentation on mesh and wire for wire segmentation. For mesh, the dataset is divided into three 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 is divided as training 1544, validation 304, test 112. The deep learning model was trained for 250 epochs using the 'adam' optimizer and a batch size of 32. The binary cross-entropy loss function was used, and early stopping was implemented with a patience of 50. The mean Intersection over Union (IoU) on the validation set was used as a monitoring metric for the early stopping process.

U-Net model trained on mesh images using data augmentation

The obtained results on test set of U-Net model trained on augmentation employed mesh images are reported in Table 3.3 and a sample composition of original image, prediction and ground truth mask is shown in the Figure 3.7.

Mean IoU	85.46%
Dice coefficient	92.16%
Pixel accuracy(all image)	90.51%
Pixel accuracy mesh only	93.21%

Table 3.3: Results on test set: mesh

Figure 3.7: A sample of mesh prediction that is trained by U-Net model on input images that are elaborated with data augmentation techniques. Input image in the left, prediction in the center and target mask in the right.

U-Net model trained on wire images using data augmentation

The obtained results on test set of U-Net model trained on augmentation employed wire images are reported in Table 3.4 and a sample composition of original image, prediction and ground truth mask is shown in the Figure 3.8

Mean IoU	49.66%
Dice coefficient	66.36%
Pixel accuracy(all image)	96.73%
Pixel accuracy wire only	59.96%

Table 3.4: Results on test set: wire

Figure 3.8: A sample of wire prediction that is trained by U-Net model on input images that are elaborated with data augmentation techniques. Input image in the left, prediction in the center and target mask in the right.

3.2.3 Gray scale images only

The U-Net model is tested on gray-scale versions of the same images to evaluate the effect of color on the model's performance.

The dataset used in this experiment consisted of images with a resolution of 512x512 pixels, and each image contained at least one piece of mesh. The dataset

is divided into three sets: a training set consisting of 854 images, a validation set consisting of 163 images, and a test set consisting of 120 images. The U-Net model is trained for 250 epochs using the 'adam' optimizer, with a batch size of 32 and a binary cross-entropy loss function. The early stopping process was 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 gray-scale mesh images are reported in Table 3.5 and a sample composition of original image, prediction and ground truth mask is shown in the Figure 3.9

Mean IoU	67.89%
Dice coefficient	80.87%
Pixel accuracy(all image)	75.77%

Table 3.5: Results on test set: gray scale mesh images

The findings showed the significance of the colors in the input images. The average Intersection over Union (IoU) of the Unet model applied on colored images was 86.28%, while when gray-scale images were used, the rate dropped to 67.89%.

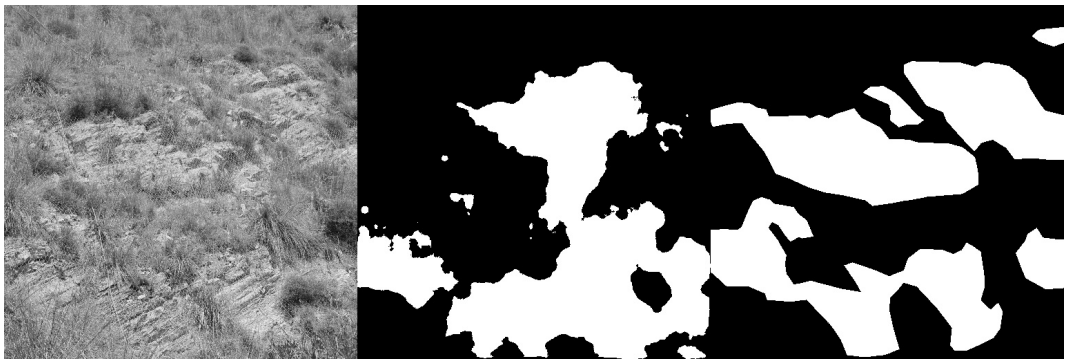


Figure 3.9: A sample of mesh prediction trained by U-Net on gray-scale input images. Input image in the left, prediction in the center and target mask in the right.

3.3 U-Net++ for binary semantic segmentation on mesh and wire images

The model consists of an encoder path and a decoder path. The encoder path consists of a series of convolutional blocks with max pooling layers, which downsample the input image. The decoder path consists of a series of upsampling layers with concatenation to the corresponding layer in the encoder path, followed by convolutional blocks. The final output layer has a sigmoid activation function.

The using deep supervision flag, if set to True, would add additional output layers with sigmoid activation for each skip connection, allowing for intermediate outputs at different scales. In this study Deep supervision is set as disables.

The UNet++ model consists of several layers, each with its own function in the network. The Input layer is the starting point for the model and defines the shape of

the input data. The Transpose convolutional layer is used for upsampling the input data, which is achieved through a transposed convolution operation that has 2x2 filters with stride 2 and same padding. The Concatenate layer merges multiple input tensors along a specified axis. The Pooling layer average pooling of the input data, resulting in a downsampled output. The BatchNormalization layer normalizes the output of the previous layer by subtracting the mean and dividing by the standard deviation of the output, making the model more robust and stable during training. The Activation layer applies an activation function to the output of the previous layer. In this model, the ReLU (Rectified Linear Unit) activation function is used, which sets all negative values to zero and leaves positive values unchanged. Finally, the Merge layer concatenates multiple input tensors along a specified axis, similar to the Concatenate layer.

3.3.1 U-Net++ without data augmentation

The dataset used in this experiment consisted of images with a resolution of 512x512 pixels. The dataset was divided into three sets a training set contains 854 images, validation set contains 163 images, and test set contains 120 images.

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

U-Net++ model trained on mesh images

The obtained results on test set of U-Net++ model trained on mesh images are reported in Table 3.6 and a sample composition of original image, prediction and ground truth mask is shown in the Figure 3.10

Mean IoU	88.86%
Dice coefficient	94.10%
Pixel accuracy(all image)	92.99%
Pixel accuracy(mesh only)	93.56%

Table 3.6: U-Net++ results on test set: mesh

U-Net++ applied on wire images

The obtained results on test set of U-Net++ model trained on wire images are reported in Table 3.7 and a sample composition of original image, prediction and ground truth mask is shown in the Figure 3.11

Mean IoU	54.56%
Dice coefficient	70.60%
Pixel accuracy(all image)	97.18%
Pixel accuracy(wire only)	63.00%

Table 3.7: U-Net++ results on test set: wire

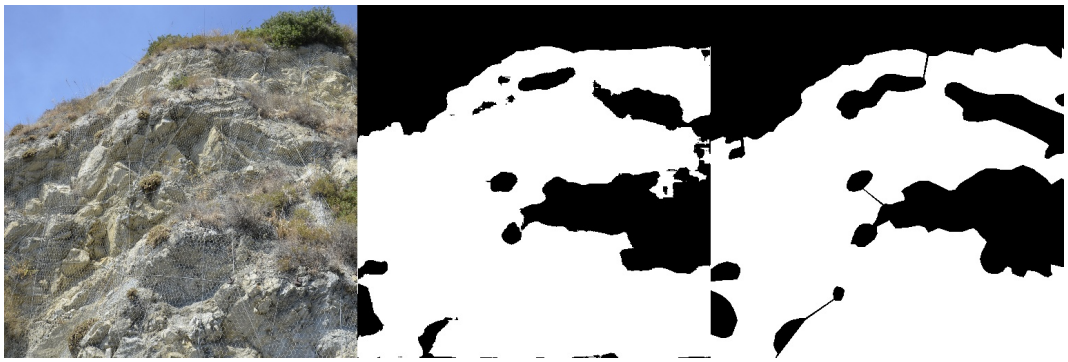


Figure 3.10: A sample of mesh prediction that is trained by U-Net++ model on input images. Input image in the left, prediction in the center and target mask in the right.

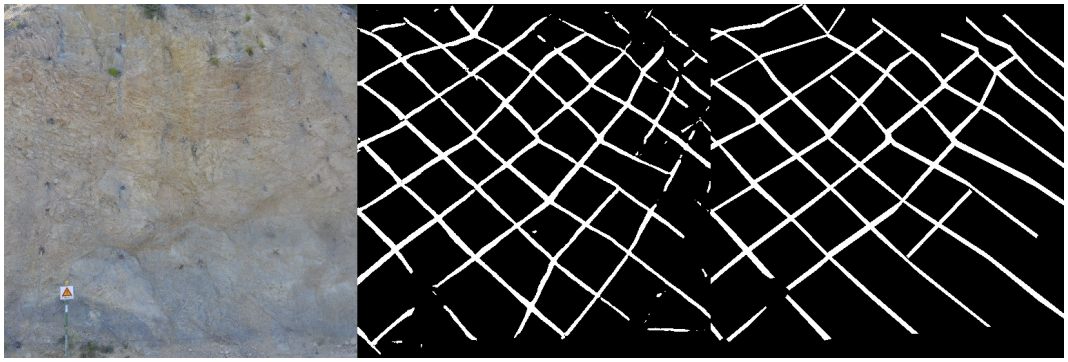


Figure 3.11: A sample of wire prediction that is trained by U-Net++ model on input images. Input image in the left, prediction in the center and target mask in the right.

3.3.2 U-Net++ applied on mesh and wire images, data augmentation employed

For mesh: the dataset was divided into three 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.

U-Net++ applied on mesh images, data augmentation employed

The obtained results on test set of U-Net++ model trained on wire images are reported in Table 3.8 and a sample composition of original image, prediction and ground truth mask is shown in the Figure 3.12

Mean IoU	89.33%
Dice coefficient	94.36%
Pixel accuracy(all image)	93.19%
Pixel accuracy(mesh only)	93.56%

Table 3.8: U-Net++ results on test set: mesh

Figure 3.12: 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.

U-Net++ applied on wire images, data augmentation applied

The obtained results on test set of U-Net++ model trained on augmentation applied wire images are reported in Table 3.9 and a sample composition of original image, prediction and ground truth mask is shown in the Figure 3.13

Mean IoU	56.07%
Dice coefficient	71.85%
Pixel accuracy(all image)	97.19%
Pixel accuracy(wire only)	66.59%

Table 3.9: U-Net++ results on test set: wire

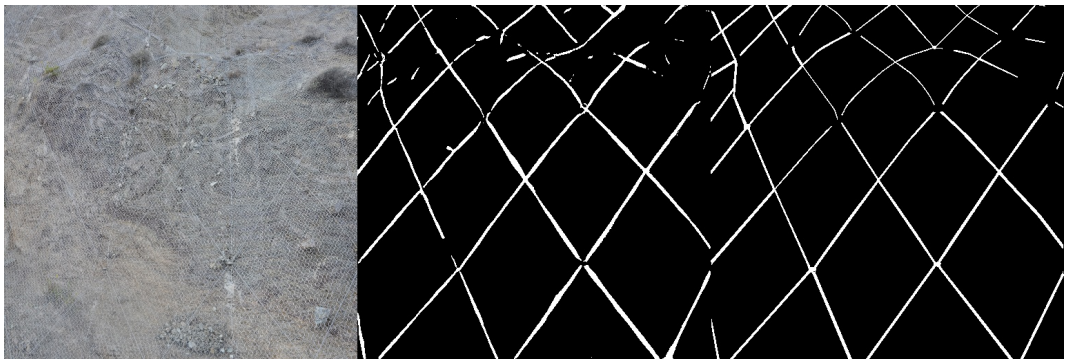


Figure 3.13: A sample of wire prediction that is trained by U-Net++ model on input images that is utilized with augmentation techniques. Input image in the left, prediction in the center and target mask in the right.

3.4 Multi-class Semantic Segmentation with U-Net

The experiment aimed to determine if using a multiclass semantic segmentation approach would lead to improved detection of mesh and wires.

The annotated data underwent pre-processing by labeling each pixel, with a value of 0 indicating the background, 1 for mesh, and 2 for wire. This application is shown in Figure 3.14

This annotated data was then converted into a numerical format using a one-hot encoding strategy. This technique represents categorical labels as a binary vector, where each element of the vector corresponds to a possible label, and the element is set to 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, the metrics were adjusted for multi-class semantic segmentation, with a pixel-wise comparison used to identify intersected pixels.

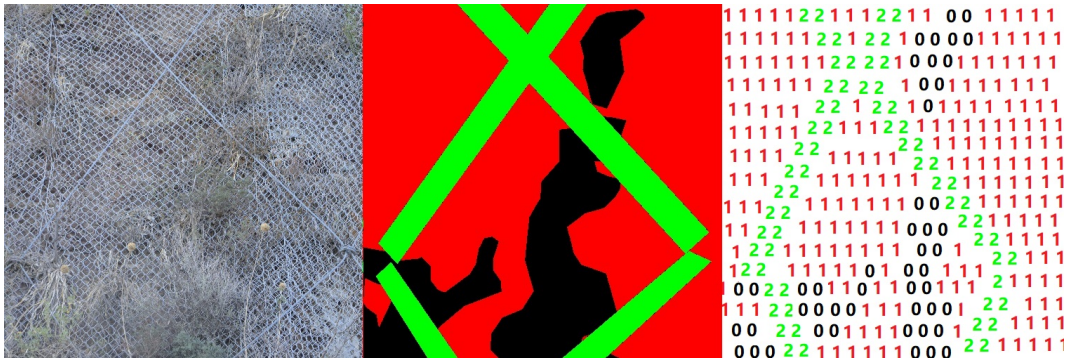


Figure 3.14: Pixel labeling demonstration. Original image in the left, ground truth mask in the center and corresponding pixel label in the right.

U-Net model without data augmentation is trained on dataset that is composed of 858 training images 167 validation images and 118 test images.

In Figure 3.15, 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 3.15: Original image in the left and multi-class mask in the right.

U-Net model without data augmentation is trained on dataset that is composed of 858 training images 167 validation images and 118 test images.

The dataset used in this experiment consisted of images with a resolution of 512x512 pixels. 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 function. The reason for using sparse categorical entropy loss in this experiment is that there are more pixels in the mesh category compared to the wire category in each image. To deal with this imbalance, the sparse categorical entropy loss function is employed, which can penalize the model more for incorrectly classifying the pixels in the minority classes. By taking into account the frequency of each class in the training data and assigning higher weights to the less frequent classes, the loss function encourages the model to focus on accurately classifying the pixels in the less common classes, leading to potentially better overall performance. The early stopping process was monitored by calculating the mean Intersection over Union (IoU) on the validation set, and the patience was set to 50.

U-Net applied on images and multiclass masks

The obtained results on test set of U-Net model trained on images are reported in Table 3.10 and a sample composition of original image, prediction and ground truth mask is shown in the Figure 3.16

Mean IoU	63.05%
Mean IoU for wire only	31.78%
Mean IoU for mesh only	79.75%
Mean IoU for background only	77.61%
Dice coefficient	87.39%
Dice coefficient for wire only	48.24%
Dice coefficient for mesh only	88.73%
Dice coefficient for background only	87.39%
Pixel accuracy(all image)	71.37%
Pixel accuracy for wire only	36.49%
Pixel accuracy for mesh only	91.43%
Pixel accuracy for background only	86.19%

Table 3.10: U-Net results on test set

In this multi-class semantic segmentation experiment, the goal is to detect both mesh and wire in a single model implementation. However, the performance of the model decreased significantly, 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 the state-of-the-art model for the dataset, it will continue to focus on the binary semantic segmentation approach.

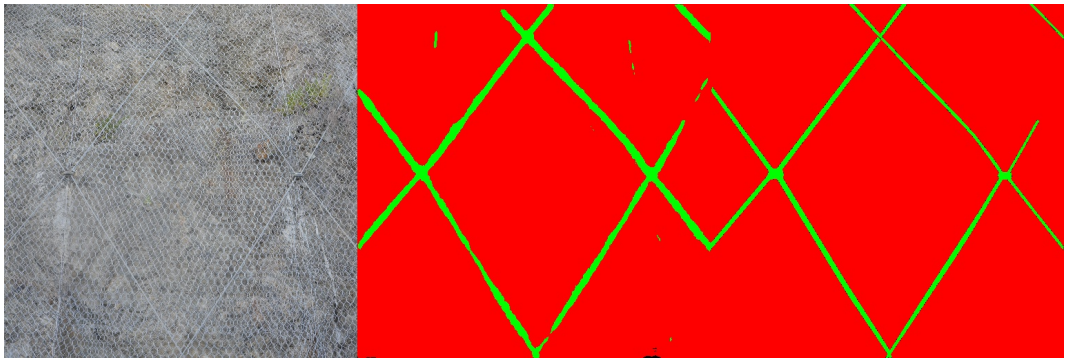


Figure 3.16: A sample of prediction that is trained by U-Net model on input images for multi-class semantic segmentation. Input image in the left, prediction in the center and target mask in the right.

Chapter 4

Evaluation

4.1 Grid Search

This section of the study aims to find the optimal combination of optimization and loss functions for the previously used U-Net model by exploring different options. The investigation involves training U-Net model in terms of binary semantic segmentation and finding the best combination for both mesh and wire separately. Once the best combination is identified, the focus shifts to determining the best filter size and number of convolution blocks for the up-sampling and down-sampling blocks.

Grid Search for wire

The obtained results on test set of U-Net model trained on wire images with Adam optimizer and Binary-focal Loss are reported in Table 4.1

Mean IoU	50.22%
Dice coefficient	66.86%
Pixel accuracy(all image)	96.83%
Pixel accuracy(wire only)	59.41%

Table 4.1: Test results of Adam, Binary Focal Loss

The obtained results on test set of U-Net model trained on wire images with Adam optimizer and Binary Cross Entropy are reported in Table 4.2

Mean IoU	47.58%
Dice coefficient	64.48%
Pixel accuracy(all image)	96.65%
Pixel accuracy(wire only)	56.51%

Table 4.2: Test results of Adam, Binary Cross Entropy

The study tested combinations of the U-Net model with the SGD optimizer, but these combinations did not produce any results. Furthermore, combinations with the Adagrad optimizer also did not work at all.

The obtained results on test set of U-Net model trained on wire images with Adadelta optimizer and Binary Focal Loss are reported in Table 4.3

Mean IoU	5.04%
Dice coefficient	9.60%
Pixel accuracy(all image)	45.33%
Pixel accuracy(wire only)	54.06%

Table 4.3: Test results of Adadelta, Binary Focal Loss

The obtained results on test set of U-Net model trained on wire images with Adadelta optimizer and Binary Cross Entropy are reported in Table 4.4

Mean IoU	4.49%
Dice coefficient	8.60%
Pixel accuracy(all image)	62.57%
Pixel accuracy(wire only)	32.78%

Table 4.4: Test results of Adadelta, Binary Cross Entropy

The obtained results on test set of U-Net model trained on wire images with rmsprop optimizer and Binary Focal Loss are reported in Table 4.5

Mean IoU	46.73%
Dice coefficient	63.70%
Pixel accuracy(all image)	96.70%
Pixel accuracy(wire only)	53.85%

Table 4.5: Test results of rmsprop, Binary Focal Loss

The obtained results on test set of U-Net model trained on wire images with rmsprop optimizer and Binary Cross Entropy are reported in Table 4.6

Mean IoU	46.06%
Dice coefficient	63.07%
Pixel accuracy(all image)	96.57%
Pixel accuracy(wire only)	54.40%

Table 4.6: Test results of rmsprop, Binary Cross Entropy

Grid Search for mesh

The obtained results on test set of U-Net model trained on mesh images with Adam optimizer and Binary Focal Loss are reported in Table 4.7

Mean IoU	84.73%
Dice coefficient	91.73%
Pixel accuracy(all image)	90.03%
Pixel accuracy(mesh only)	92.43%

Table 4.7: Test results of Adam, Binary Focal Loss

The obtained results on test set of U-Net model trained on mesh images with Adam optimizer and Binary Cross Entropy are reported in Table 4.8

Mean IoU	85.69%
Dice coefficient	92.29%
Pixel accuracy(all image)	90.72%
Pixel accuracy(mesh only)	92.87%

Table 4.8: Test results of Adam, Binary Cross Entropy

The obtained results on test set of U-Net model trained on mesh images with Adadelta optimizer and Binary Focal Loss are reported in Table 4.9

Mean IoU	59.78%
Dice coefficient	74.83%
Pixel accuracy(all image)	59.80%
Pixel accuracy(mesh only)	99.90%

Table 4.9: Test results of Adadelta, Binary Focal Loss

The obtained results on test set of U-Net model trained on wire images with Adadelta optimizer and Binary Cross Entropy are reported in Table 4.10

Mean IoU	59.8%
Dice coefficient	74.85%
Pixel accuracy(all image)	59.80%
Pixel accuracy(mesh only)	99.99%

Table 4.10: Test results of Adadelta, Binary Cross Entropy

The study tested combinations of the U-Net model with the SGD optimizer with Binary Focal Loss but this combination did not produce any results. Furthermore, combinations with the Adagrad optimizer also did not work with Binary Focal Loss.

The obtained results on test set of U-Net model trained on mesh images with SGD optimizer and Binary Cross Entropy are reported in Table 4.11

Mean IoU	67%
Dice coefficient	80.24%
Pixel accuracy(all image)	74.36%
Pixel accuracy(mesh only)	87.03%

Table 4.11: Test results of SGD, Binary Cross Entropy

The obtained results on test set of U-Net model trained on mesh images with adagrad optimizer and Binary Cross Entropy are reported in Table 4.12

The obtained results on test set of U-Net model trained on mesh images with rmsprop optimizer and Binary Focal Loss are reported in Table 4.13

The obtained results on test set of U-Net model trained on mesh images with rmsprop optimizer and Binary Cross Entropy are reported in Table 4.14

Mean IoU	59.80%
Dice coefficient	74.85%
Pixel accuracy(all image)	59.80%
Pixel accuracy(mesh only)	100%

Table 4.12: Test results of adagrad, Binary Cross Entropy

Mean IoU	82.86%
Dice coefficient	90.62%
Pixel accuracy(all image)	88.73%
Pixel accuracy(mesh only)	91.02%

Table 4.13: Test results of rmsprop, Binary Focal Loss

Mean IoU	84.74%
Dice coefficient	91.74%
Pixel accuracy(all image)	90%
Pixel accuracy(mesh only)	92.89%

Table 4.14: Test results of rmsprop, Binary Cross Entropy

The optimal model for wire is achieved using Adam optimizer, Binary focal Loss. Similarly, for mesh, the best model is obtained using Adam optimizer and Binary Cross Entropy.

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 were conducted using a starting filter size of 8, we are now examining the impact of filter size on model performance by testing with starting filter sizes of 4 and 16. This will provide the optimal filter size for the model.
Experiment with filter size 4 and 16 is reported below.

Trial of different filter sizes for wire

The obtained results on test set of U-Net model trained on wire images with Adam optimizer and Binary Focal Loss with base filter size 4 are reported in Table 4.15

Mean IoU	41.12%
Dice coefficient	58.28%
Pixel accuracy(all image)	96.37%
Pixel accuracy(wire only)	47.16%

Table 4.15: Test results of unet model trained on wire images with base filter size 4

The obtained results on test set of U-Net model trained on wire images with Adam optimizer and Binary Focal Loss with base filter size : 16 are reported in Table 4.16

Mean IoU	52.15%
Dice coefficient	68.55%
Pixel accuracy(all image)	96.96%
Pixel accuracy(wire only)	61.70%

Table 4.16: Test results of unet model trained on wire images with base filter size 16

Trial of different filter sizes for mesh

The obtained results on test set of U-Net model trained on mesh images with Adam optimizer and Binary Cross Entropy with base filter size 4 are reported in Table 4.17

Mean IoU	82.35%
Dice coefficient	90.32%
Pixel accuracy(all image)	88.33%
Pixel accuracy(wire only)	91.32%

Table 4.17: Test results of unet model trained on mesh images with base filter size 4

The obtained results on test set of U-Net model trained on mesh images with Adam optimizer and Binary Cross Entropy with base filter size 16 are reported in Table 4.18

Mean IoU	86.99%
Dice coefficient	93.04%
Pixel accuracy(all image)	91.64%
Pixel accuracy(wire only)	93.50%

Table 4.18: Test results of unet model trained on mesh images with base filter size 16

It appears that a base filter size of 16 is the most effective for both mesh and wire images. This suggests that larger filters are better suited for our dataset in the unet model.

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

Having identified the best optimizer, loss function, and base filter size, the focus now turns to examining how performance is affected by different numbers of layers in the up-sampling and down-sampling blocks of the U-Net model. Experimenting the performance of a smaller model with fewer parameters and potentially lower accuracy.

The obtained results on test set of U-Net model trained on mesh images with Adam optimizer and Binary Cross Entropy with base filter size 16, 4 layers in encoder and 5 layers in decoder are reported in Table 4.19

Mean IoU	85.96%
Dice coefficient	92.45%
Pixel accuracy(all image)	90.88%
Pixel accuracy(mesh only)	93.34%

Table 4.19: Test results of unet model trained on mesh images with smaller encoder decoder blocks

The obtained results on test set of U-Net model trained on wire images with Adam optimizer and Binary Focal Loss with base filter size 16, 4 layers in encoder and 5 layers in decoder are reported in Table 4.20

Mean IoU	51.16%
Dice coefficient	67.69%
Pixel accuracy(all image)	96.93%
Pixel accuracy(mesh only)	59.86%

Table 4.20: Test results of unet model trained on wire images with smaller encoder decoder blocks

The experiment shows that using smaller encoder-decoder blocks results in less accurate solutions, but the difference is not significant. This means that smaller encoder-decoder blocks can be used to obtain a more lightweight network if needed. On the other hand, using larger blocks can be advantageous. Hence, since this study aims for a higher accuracy percentage, it is better to keep the encoder-decoder blocks as they are. Nonetheless, to prevent the model from becoming more complex and having more parameters, it is advisable to maintain the current 5-layer decoder and 6-layer encoder configuration.

Chapter 5

Conclusion

In conclusion, this thesis concentrated on analyzing wire and mesh images at the pixel level. 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 both binary and multi-class segmentation tasks were performed. 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. The results showed that the best U-net model for wire images was achieved with binary focal loss function and Adam optimizer, while for mesh images, the best model was obtained with binary cross-entropy loss and Adam optimizer. Bigger filter sizes yielded better results on the data-set. Moreover, it was concluded that binary semantic segmentation provided higher accuracy results for the U-net model than multi-class semantic segmentation.

Appendix A

Appendix

Team name	Entry description	Number of object categories won	mAP
NUIST	cascaded region regression + tracking	10	0.808292
NUIST	cascaded region regression + tracking	10	0.803154
CUVideo	4-model ensemble with Multi-Context Suppression and Motion-Guided Propagation	9	0.767981
Trimps-Soushen	Ensemble 2	1	0.709651

Table A.1: Morbi ac varius enim, ac lobortis odio. Nullam venenatis, erat in faucibus vestibulum, magna dolor faucibus sapien, ac iaculis tellus urna ut purus. Pellentesque finibus urna eget maximus cursus. Aenean mollis ante nec dolor iaculis, ac malesuada enim ultrices. Aenean vulputate felis sapien, quis rhoncus odio dictum vitae.

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.