FH JOANNEUM
University of Applied Sciences

STUDY AND COMPARISON OF DIFFERENT DEEP LEARNING METHODS REFERING TO SKIN WOUND DETECTION

A Master thesis by

CHAIMAE KASSARA GUENNOUN

advised by

DI Dr. Robert Mischak, MPH

and submitted to the

Institute for eHealth

of the

Graz University of Applied Sciences

in partial fulfillment of the
requirements for the degree of a
Master of eHealth (MSc)

July 2019

Nothing in life is to be feared, it is only to be understood. Now it is the time to understand more, so that we may fear less." (Marie Curie)

Acknowledgements

Coming to Graz for an exchange semester probably is the first risky choice I ever made. The second would be deciding to take as a topic for my master thesis, deep learning, a topic that happened to be almost unknown to me, but always appealing. It was a chance that I wanted to take to get closer to a field that I would like to work in at the future, and I did, even though it was not easy to understand. It was a challenge.

I can try to do my best expressing how grateful I am to all the people that made possible this Master thesis to be in your hands at this moment, and It would not be enough. I can write pages and pages, but still I would probably need more, but if I can summarize it in one page, I would like to express my very great appreciation to my Master thesis advisor, Dr. Mischak for the help provided during the development of this research. I wound not have been able to find an accurate research question without the help provided.

Specially, I would also like to offer my most sincere thanks to Msc Bianca Schnalzer for all the support. She has been an actual guide in this research. The door to her office was always open for any question regarding the research.

I am particularly grateful for the help provided by Msc Robert Rehb regarding LaTex, a completely new software for me to use, still it was an easy task thanks to his help.

Special thank to my friends, who had been great supporters 1979,8 km far from Graz. They encouraged me each time I needed and they believed in me more than I did in myself.

Finally, and absolutely not less important, a very special mention of gratitude to my parents, to my family. I would like to thank them for giving me the wings to fly far while I am still close. Thank them for believing in me, in my capabilities. They have been my energy to keep going. They are great supporters, who do not care if I fail, because all I should do is try it again. They remembered me each time that life is based on risks, and on exploring what we do not know, discover new things, and dive into the unknown so it can become known.

Thank you all, for making possible this amazing ending of my university studies for now,

Chaimae Kassara

Abstract

The present Master Thesis deals with neural-network-based skin wound detection using MatLab. MatLab allow the users to integrate pre-trained deep learning architectures by using the Neural Network Toolbox. For this reason it was chosen to develop a code in order to re-train pre-trained models such as ResNet, VGG and AlexNet. This Toolbox also allows the definition of various machine learning models. In this Thesis, as a first Step, commonly used Deep learning architectures for semantic segmentation were identified and evaluated for skin wound detection, aiming to find the best performer. Previous steps of data Labelling and partitioning were made. The chosen models, according to the state-of-art in deep learning, were trained with different training parameters and it was part of this Master thesis to develop a criterium to evaluate the approaches and compare the different pre-trained architectures. Feasibility of Skin Wound detection on Matlab has been proven, although the available resources clearly were a limiting factor.

Zusammenfassung

Die vorliegende Masterarbeit beschäftigt sich mit der Wunderkennung basierend auf neuronalen Netzwerken in MatLab. MatLab ermöglicht es BenutzerInnen, vortrainierte Deep-Learning-Architekturen mittels der „Neural Network Toolbox“ zu integrieren. Aus diesem Grund wurde MatLab gewählt, um vortrainierte Modelle wie ResNet, VGG und AlexNet erneut zu trainieren. Die Toolbox ermöglicht ebenso die Definition verschiedener Machine-Learning-Modelle. In der vorliegenden Masterarbeit wurden häufig verwendete Deep-Learning-Architekturen identifiziert und hinsichtlich Wunderkennung evaluiert. Ein Ziel war es, das Modell mit der besten Performanz zu finden. Notwendige Vorverarbeitungsschritte inkludierten das Labeln sowie Partitionieren der Daten. Im Anschluss wurden die ausgewählten Modelle mit verschiedenen Trainingsparametern trainiert. Zudem wurden Kriterien zur Evaluierung gefunden, um verschiedene vortrainierte Modelle zu vergleichen. Vorhandene Ressourcen stellten einen limitierenden Faktor dar, dennoch konnte die Wunderkennung in MatLab realisiert werden.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter wound care management, the importance of wound detection and the need of an automated wound detection algorithm are going to be introduced.

1.1 Introduction

Chronic wounds are a signal of non-healthy individuals and can be a symptom of further illness such as diabetes or obesity among others [Sen et al., 2009].
A wound is defined as an injury in the body tissues caused by an accident, surgery or other causes [MedlinePlus, 2019]. Wounds are injuries that break the continuity of the tissue [The Editors of Encyclopaedia Britannica, 2019], specially the skin, involving external and internal breaks [Roddick and Higuera, 2018]. Depending on the healing time, a wound can be an acute wound or chronic wound. An acute wound, the healing process is without complication in a predicted period of time. A chronic wound lasts longer and complications might occur [Center, 2019]. Chronic wounds are those that failed the process of repairing tissue to restore the anatomic integrity of the lesion. This type of wound is epidemically silent and challenge for public health care and economy [Sen et al., 2009]. Chronic wounds may occur as diabetic ulcers, pressure ulcers, arterial insufficiency ulcers or venerous ulcer [Jarbrink et al., 2016]. The increase in health care cost, and an aging population are becoming a burden in the field of treatment. A huge percentage of the money spent in health care are due to wounds care, taking into account the diminishing life quality of the patient and loss of productivity one can make a strong statemente about the importance of an automatic wound detection algorithm to help reduce the costs of wound care in the healthcare system, achieving an accurate diagnosis of wounds [Sen et al., 2009] [J.Veredas et al., 2015].
Another classification of wounds could be related to the penetration. A non-penetrating wound could happen due to a friction with some surfaces and a break of the skin does not

happen, some examples are abrasions, lacerations or contusions. A penetrating wound results in slan damage the skin, for instance, skin cuts, surgical wounds and stab wound [Center, 2019].
Complications of a skin wound vary, starting with regular infections, inflammation, scarring or loss of functions [Center, 2019], and when a wound presents complications, this means a need of observation from a health care center.

1.2 Motivation

With the increasing age of elderly of the European population, the prevalence of chronic wounds is also increasing due to chronic diseases that have more incidence in Aging population. For this reason and in order to achieve a better wound care management, it is necessary to improve the automatic detection of skin wounds and improve measurement which would at the same time help achieving an improved therapy for the patients. Wound surface area changes over time, which makes it difficult to achieve an accurate measurement of wound surface area. Clinicians nowadays determine wound size manually estimating the length and the width, which is not highly accurate [Wang et al., 2015]. Based on Artificial intelligent techniques we aim to achieve an algorithm that supports the management of wound detection and measurement to improve humans attempts. Several applications provide solutions in the photography phase; however, image processing is still a challenge.
The deterioration in quality of life that a patient with chronic wound suffers from is comparable to several severe illnesses such as sclerosis. In addition, an infection arises the probability of hospital admission up to 60% and extends hospital stay. In Spain it was detected that 3.75% of diabetic patients, might suffer from an amputation due to diabetic foot ulcer, and the mortality percentage for these patients is between 45% and 55% within a 5 years period. All this data highlights the importance to review the economic impact of chronic wounds in the health care system and investigate how to improve wound care management. In Europe, according to [Smith and Nephew, 2014], the cost related to diabetic foot ulcers are between 4000 and 6000 Million of Euros, and the cost related to venous ulcers is 6500 Million Euros. Also, it is mentioned in the same article that the economic cost of pressure wounds in Spain may arise to 461 Million of Euros, which is the 5% of the total estimated health care costs. In the case of United States, we can observe in Image 1.1, the prevalence of wound types in the Medicare population in 2014.

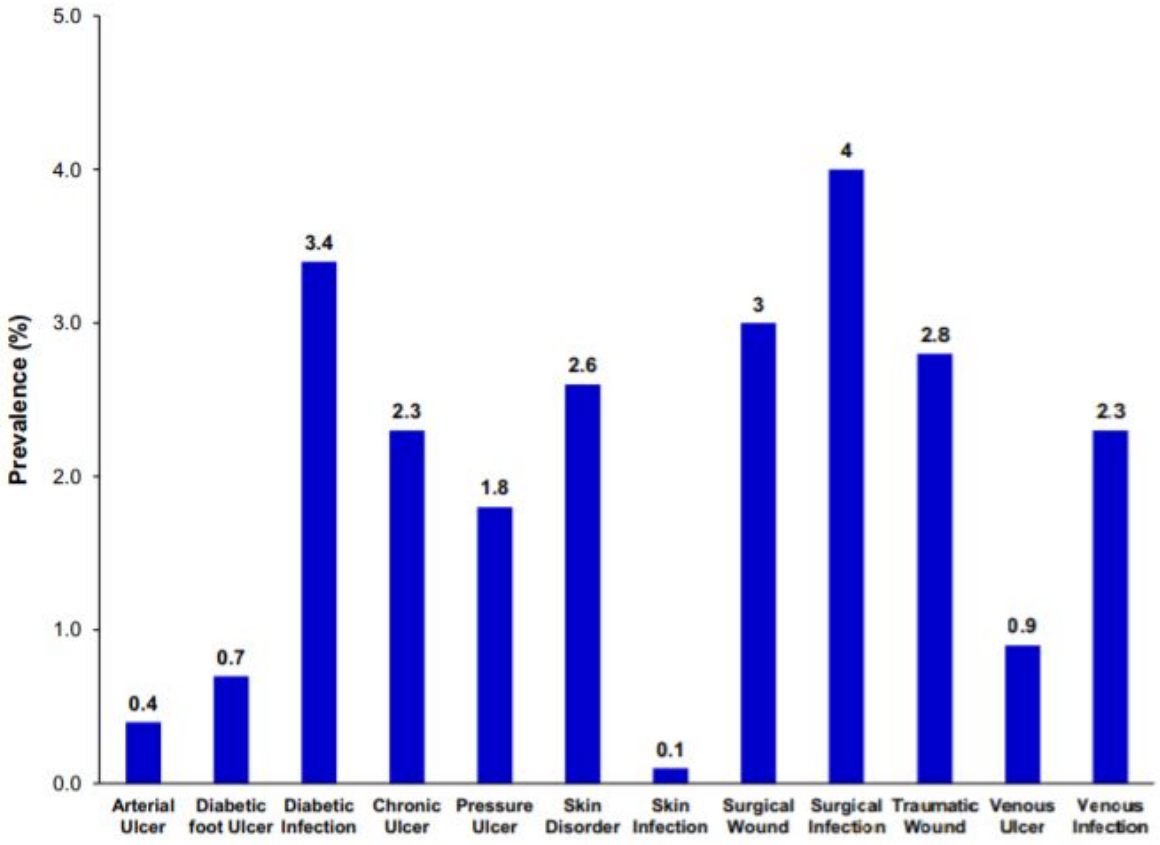

Figure 1.1: The image illustrates the prevalence of wound types in the Medicare population in 2014. (Source: [Nussbaum et al., 2018])

As mentioned previously, it is important to find solutions in wounds care management. In order to achieve an improvement it is important to appeal to new technologies or techniques that arise nowadays. Artificial Intelligence is one of the most important fields of investigation that is growing with non-stopping, especially related to deep learning concerns. In this Master Thesis the aim is to find a suitable Deep learning architecture for wound detection.

1.3 Reaserch Question.

The goal of this thesis is to identify a suitable deep learning architecture for semantic segmentation, aiming to have a suitable model for wound detection by using MatLab. The aim is to compare the different Deep Learning architectures related to semantic image segmentation for wound detection and define indicators to evaluate the performance of

these architectures. Finally, this parameteres are evaluated and compared to achieve some conclusions about the most suitable Deep learning architecture for skin wound detection.

1.4 Methodology

The first step in this Thesis was to conduct a literature review in order to understand the basis of deep learning and find accurate techniques for semantic segmentation related to skin wound detection. Moreover, a literature review was performed to find the best deep learning architectures for the aim of this Thesis, based on semantic segmentation in MatLab for skin wound detection and the next step was to proof if these architectures are suitable or not for the goal proposed. In order to evaluate the performance of this deep learning architectures, evaluation parameters were set according to literature findings. Finally, by evaluating the results of the parameters in each case, these architectures were compared in order to find the most suitable architecture fot the given research question.

1.5 Thesis Structure

The Master Thesis is structured as followed: In Chapter 2 the theorical aspects of this Thesis are provided, such as Neural Networks, Support Vector Machines, semantic segmentation, and machine learning in general. According to stat-of the art findings, an explanation over Deep learning and convolutional neural networks is given. Chapter 3 summarizes an explanation of pre-trained Models and deep learning frameworks. The results of this Master Thesis are presented in Chapter 4, in the examined MatLab approaches studied and tables summarizing the results. Finally a discussion is provided Chapter 5, to provide possible future work according to the present thesis.

Chapter 2

Machine Learning in Medical Imaging

In this Chapter we will introduce the techniques used to develop this master Thesis related to deep learning, in order to find an automated wound detection algorithm.

2.1 Artificial Intelligence

The term Artificial Intelligence was first used in a conference in 1950, in Dartmouth. Its applications started in 1950s with physician searching for processes to improve their diagnosis using computer-aided programs [Yu et al., 2018]. The applications of AI had increased in the recent years, from 2008 so as the research in this field. This increase in literature and research is due to a significantly arise in data availability and the development of the new modern computers [Tran et al., 2019].
The increase in electronic data, thanks to the EHR and the development of new methods to analyse this data supported the evolution of AI and machine learning techniques. The potential of the use of AI in the field of medicine has increased improving the quality and lowering the cost of patient care [Neill, 2013].
Artificial Intelligence consists on the use of computational technology or programs to simulate human intelligence, to perform tasks that are normally performed by humans, such as reasoning, sensory intelligence, interaction, to make decision on a set of data collected [Tran et al., 2019]. AI makes use of several devices and principles from diverse fields in order to solve a model, problem or to replicate intelligence and cognitive processes. It uses datasets to make predictions about a diagnoses, discovers new diseases, or find the best option as a treatment for a particular patient, etc. [Dey et al., 2019].
AI has two branches, the virtual branch and the physical branch. The physical branch consists of the use of robots (medical devices as part of the patient care: such as surgery robots). In this thesis the main goal is to focus on the virtual branch, which consists on the use of machine learning techniques, mathematical algorithm to simulate human

intelligence or behavior without the interventions of humans in order to improve learning through experience [Hamet and Tremblay, 2017].

2.2 Machine Learning

Machine learning is a set of techniques that are used to give the ability to learn from experience and it helps analyzing structured data. Machine learning is useful to identify patterns from a large dataset without prior assumptions. it is highly used in prediction models and decision -making models. It constructs data analytical algorithms to extract features from datasets [Dey et al., 2019].
Machine learning methods are classified as supervised or unsupervised:

- Supervised: works by collecting a large number of training cases, which contains inputs and the desired output labels. By analyzing the patterns all over the labeled input-output pairs, the algorithm learns to produce the correct output for a given input on new cases [Yu et al., 2018]. The algorithm searches for patterns in that data that fit a particular outcome of interest [Dey et al., 2019].

- Unsupervised: infers the underlying patterns in unlabeled data to find sub-clusters of the original data, to identify outliers in the data or to produce low-dimensional representations of the data [Yu et al., 2018]. Pattern recognition happens freely within the data supplied. The program does not try to fit any outputs, it tries to identify consistent patterns in the data [Dey et al., 2019].

In this thesis the focus was related to supervised machine learning techniques. In the following section the state-of-art techniques used in the medical imaging field will be introduced. The main focus in this research included artificial neural networks, however, it is important to mention, for instance, the use of support vector machines in the same field. These techniques will be explained in section2.5.

2.3 State-of-art Image Processing

One of the main fields where the use of Artificial Intelligence techniques became crucial is in the field of medical imaging. A medical data increases, it is easier for health care professionals to make mistakes related to the interpretation, detection or analysis of medical data. AI can be used in order to improve diagnosis and to select the best therapeutic choice for patients in the healthcare system. It is highly supportive for an accurate interpretation of the medical images [Rastgarpour and Shanbehzadeh, 2011]. Various methods

proved to be useful in image analysis. The most used methods are related to digital image processing, machine learning, pattern recognition and fuzzy logic [Rastgarpour and Shanbehzadeh, 2011]. Theimage processing process contain different tasks, Figure 2.1 shows a schema of the main tasks in image processing:

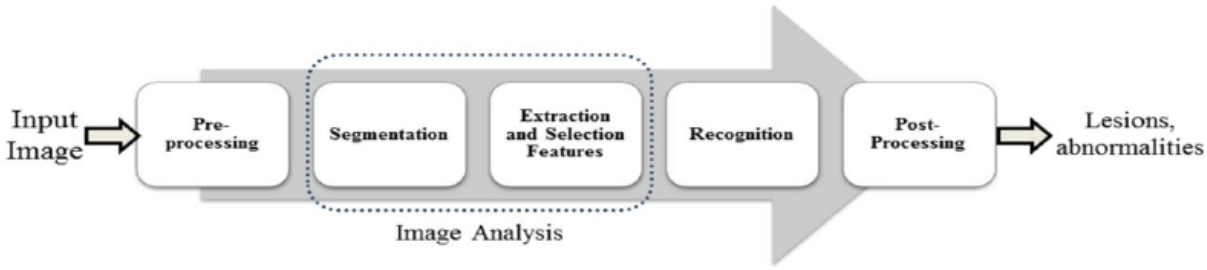

Figure 2.1: Image processing schema containing the main tasks related to image analysis: preprocessing step, segmentation, extractions and Selection Features, Recognition and Post-Processing. (Source: [Rastgarpour and Shanbehzadeh, 2011])

Image segmentation is one of the most challenging task in image processing. It consists of dividing the image in multiple parts with similar features, so called segments [Kaur and Kaur, 2014], in order to identify objects or any relevant information in the medical image, in this case the goal is related to wound skin detection [The MathWorks Inc., 2019b].
In this chapter we will focus on machine learning technique, however, a quick review over some other commonly used techniques is given below:

- Methods based on image processing techniques:
  - Thresholding: the partition depends on the intensity level of the pixels. It is considered the most basic segmentation method [Kaur and Kaur, 2014].
  - Edge detection: depends on changes in the intensity value of the pixels..
  - Region growing: divides the image in regions with similar features or characteristics [Kaur and Kaur, 2014].
  - Watershed segmentation: this method is based on considering the gradient of an image as a topographic surface, where pixels with highest gradient are represented as boundaries which are continuous [Kaur and Kaur, 2014].
- Machine learning
  - Pattern recognition
  - Artificial Neural Networks

  - K-means clustering
  - Support Vector Machines

- Ability of experts system: segmentation using expert knowledge such as rules, models and atlases.

- Multispectral and multimodal images: registration

This four sets of methods for segmentation are accurate and complex [Rastgarpour and Shanbehzadeh, 2011]. All these methods have advantages and disadvantages, after a deep literature it was decided to focus on machine learning techniques to achieve the goal of this master thesis.

2.4 Semantic Segmentation

Semantic segmentation consists on assigning each pixel of an image to a label or class. Since it works with pixels, it provides as result several possible arbitrary shapes [Royo, 2016]. There are different problems defined in Computer Vision domain [Lamba, 2019]:

- Image Classification: given an image as an input, the expected output would be a discrete label. It is assumed the existence of only one object in the image.

- Classification with Localization: It is the classification problem, however, in this case it is expected also the localization of the object in the image.

- Object Detection: In this case it is assumed that image in the image , the goal is to localize and classify the different objects in the given image.

- Semantic Segmentation: This is the aim of this Master Thesis and it refers to labelling each pixel of the image with the corresponding class

- Instance Segmentation: In this case, along with classifying each pixel, it is expected a classification of each instance of a class separately.

Figure 2.2 illustrates the difference between object detection, semantic segmentation and Instance segmentation.

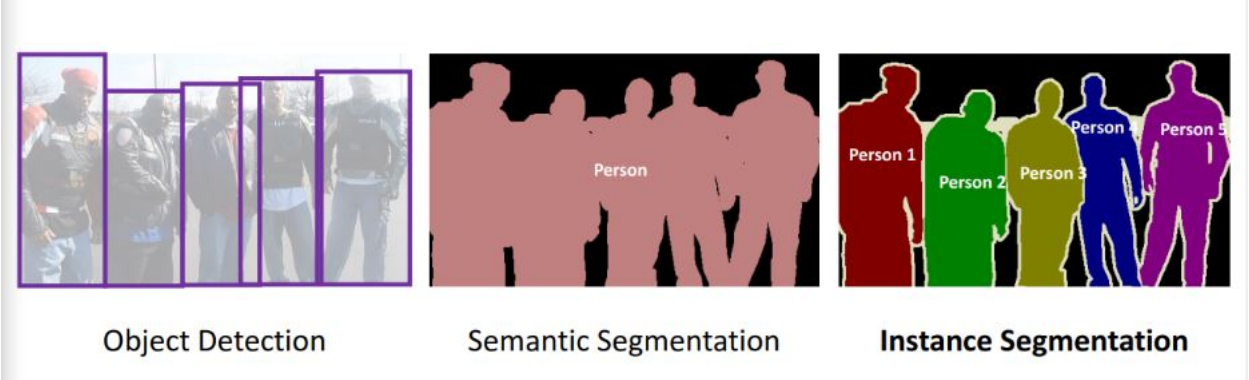

Figure 2.2: The image illustrates a graphic representation of object detection, semantic segmentation and instance segmentation. (Source: [Lamba, 2019])

Semantic segmentation is a common issue in the field of computer vision [Garcia-Garcia et al., 2017]. Historically it has been studied as a clustering problem, however, given the success of deep learning recently, the focus has changed towards supervised variants of the segmentation problem [Xia and Kulis, 2017]. Therefore, nowadays, Semantic image segmentation is mostly supervised: it works with a set of classes, nevertheless, unsupervised algorithm exists and they do not distinguish classes [Thoma, 2016]. Encoder-Decoder is one of the most used methods in unsupervised learning where the encoder is a pre-trained model used to extract features and the decoder projects the input from its lower-dimensional representation into a higher-dimensional space [Xia and Kulis, 2017]. Nowadays, it is possible to train an artificial neural network in a supervised way in order to implement semantic image segmentation. This is possible by training deep learning architectures. Deep learning architectures such as AlexNet, GoogleNet, ResNet and VGG, have been proven to be relevant in this field as mentioned in [Garcia-Garcia et al., 2017].

2.5 Machine learning in Image Segmentation and object detection

Object detection consists on, given an Image, a model or algorithm help identify the present objects in the image and provide the information about their position in the particular image. The model is trained to detect the presence and location of several classes of objects, in the case of this project, the goal is to detect the presence of a wound skin and its position [TensorFlow, 2019]. For this purpose, the use of neural networks is the mostly used machine learning techniques, various architectures can be used. In

the next paragraph a brief introduction to Support vector machines is provided, since it is another machine learning algorithm used in image segmentation and object detection, nevertheless, in all the literature studied in this project, Neural Network architectures are the most used.

2.5.1 Support Vector Machines

Support vector machines is a supervised learning method suggested by Vladimir Vapnik in 1992 as mentioned in [Bentaouza and Benyettou, 2018]. This technique is essentially for classifying objects which could be linear classification or non-linear classification [Bell, 2014]. [Zhanh et al., 2016] suggest that Support vector machines have proven to be excellent in image segmentation, and also it could be combined with different methods to help improve the performance in image segmentation. It is a highly used method for classification in healthcare and other fields of science [Schnalzer, 2018]. The goal of this method is finding a linear classifier that separates the different classes of data points and maximize distance between the classes [Bentaouza and Benyettou, 2018]. The classifier is a decision boundary or hyperplane that separates the data. The main objective is to find the hyperplane in an N-dimensional space, being N the number of features, that maximize the margin between data of different classes as shown in figure 2.3, this would provide some reinforcement helping future data to be classifies with confidence. The hypeplane dimension depend on the number of features, if the number of features is 2, then the hyperplane is a line separating two classes, instead if the number of features is 3, the hyperplane is a 2D plane [Ghandi, 2018]. The closest point that define the orientation and position of the hyperplane are support vectors [Bentaouza and Benyettou, 2018], using this support vector it is possible to maximize the margin of classifier, find the optimal decision boundary, this are the points that help build the support vector machine [Ghandi, 2018].

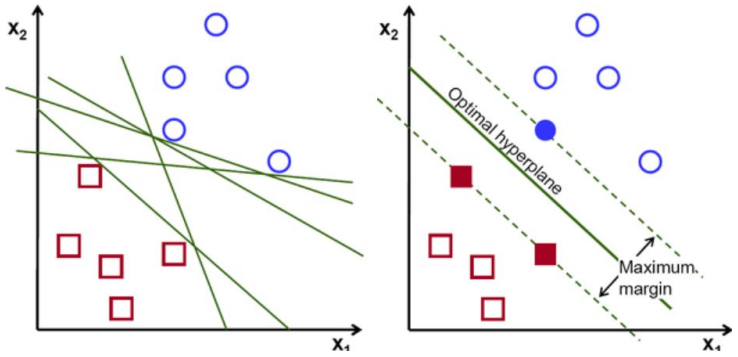

Figure 2.3: The figure illustrates basic idea of Support Vector Machines applied to a binary classification problem. The different possible hyperplanes are represented in the left figure and the optimal hyperplane that maximize the margin between classes is illustrated in the right. It is represented the optimal hyperplane and the support vectors. (Source: [Ghandi, 2018])

In summary, [Vanitha L. AND Venmathi A.R. , 2011] suggest that support vector machine operates on two mathematical steps, first it is presented a nonlinear mapping of an input vector into a high-dimensional feature space hidden from the input and output, in the second step it is carried the construction of the optimal hyperplane to separate the features discovered in the first step. In general, Classification with support vector machine could be useful for binary (two classes) or multiclass (multiple classes) problems [Schnalzer, 2018]. In a classification problem every point of data or object to be classified has a set of features. The confidence of the class points is a probability, an equation that indicates how probable it is that the hyperplane is producing the correct classification and it is based on the distance between the points and the hyperplane. By combining the probability of all data points, it is produced the 'likelihood of the data'. The goal is to search for the set of line parameters with the closest probability value to 1 [Bell, 2014].

As mentioned, support vector machine work in linear and non-linear forms. In linear classification a line with margins created by the hyperplane classifies the objects in one class or another. By adding a new object, the hyperplane and support vector might move. The key is to keep a maximum margin between classes increasing the confidence in prediction [Bell, 2014].

In real life it is difficult to find an easy problem where the objects lie in one side or the other of the hyperplane, it is possible to find objects straying from the hyperplane as illustrated in figure 2.4, [Bell, 2014]. A real-life dataset contains noisy data points, which make it difficult for the data to be linearly separable. The idea is to build a Support vector machine that creates a non-linear decision boundary capable of classifying nonlinearly separable

data. The idea is to use a kernel function ('kernel trick') to transform linear classifiers into higher dimensional feature space achieving a separable data by a non-linear hypersurface [V. Keckman, 2015]. There are several kernel types, such as hyperbolic tangent, Gaussian radial basis function (RBF) and two polynomial functions [Bell, 2014]. An example of nonlinear classification hypersurface is illustrated in figure 2.5.

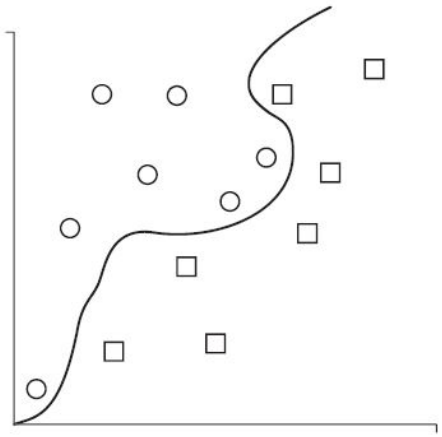

Figure 2.4: The figure illustrates an example of data points of objects that are straying from the hyperplane. (Source: [Bell, 2014, pag 147])

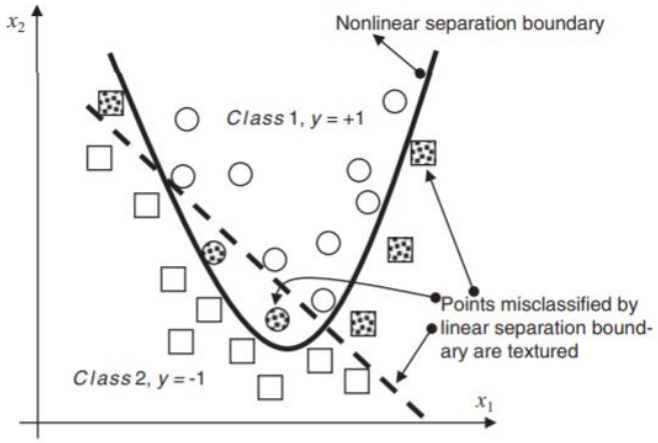

Figure 2.5: The figure illustrates an example of non-linear separation and linear separation. (Source: [V. Keckman, 2015])

2.5.2 Neural Networks

Human information processing mechanisms sometimes might be complex, and the regular machine learning techniques may not be sufficient [Liu et al., 2017]. Research related to Artificial neural networks have emerged in the recent decades [Liu et al., 2017] and became highly accepted within several disciplines in order to solve or model complex real world problems [Basheer and Hajmeer, 2000]. Artificial Neural Networks (ANN) are computational modelling tools [Liu et al., 2017] for information processing. It is based or inspired by the biological nervous system [Kruse et al., 2016, pag 7]. The brain is an organ that acquires knowledge from experience, through learning, and this is the basis of the neural network models [Sumathi and Paneerselvam, 2010]. The ANN are made of several simple units working in parallel, which are the fundamental processing element, the neuron [Kruse et al., 2016, pag 7]. The biological neuron receives information or inputs from other sources, combines the signal of this inputs and performs a nonlinear operation on the results of the combination, the final result would be the output [Sumathi and Paneerselvam, 2010, pag 7,8]. The neurons in ANN are connected to each other and they communicate by sending information in the form of an activation signal [Kruse et al., 2016, pag 7]. Neuron in ANN are considered as threshold logic units or perceptron (the activation of the neuron and the transmition of the signal to other neurons depends on the balance between the excitatory input and inhibitory) [Kruse et al., 2016, pag 7]. The first ideas about ANN date back to 1943 with Warren McCulloch, a model was examined by this neurologist who composed a theme on the working of neurons [Sumathi and Paneerselvam, 2010, pag 29]. In 1958, Rosenblatt introduces the perceptron to solve problems related to character recognition [Basheer and Hajmeer, 2000]. A threshold logic neuron consists on a processing unit that receives n inputs, x_i, ..., x_n, as a stimuli, combines them to find a net input which passes through a linear threshold gate and transmit an output signal y, forward to another neuron depending on the balance between the threshold θ and the net input. [Kruse et al., 2016, pag 15] define the neuron unit as: "A threshold element is a processing unit for real-valued numbers with the input x_i, ..., x_n and the output y. The unit as a whole possesses a threshold θ and every input x_i is associated with corresponding weights w_i. A Threshold element define the following logic unit computes the function":

$$y = \begin{cases} 1 & if \sum_{i=1}^{n} w_i x_i \geq \theta \\ 0 & if \sum_{i=1}^{n} w_i x_i \prec \theta \end{cases} \tag{2.1}$$

Positive values of weights helps exciting the neuron by increasing net input value, meanwhile negative weights reduce the net input value inhibiting the neuron [Basheer and Hajmeer, 2000]. A threshold logic unit is represented in figure 2.6

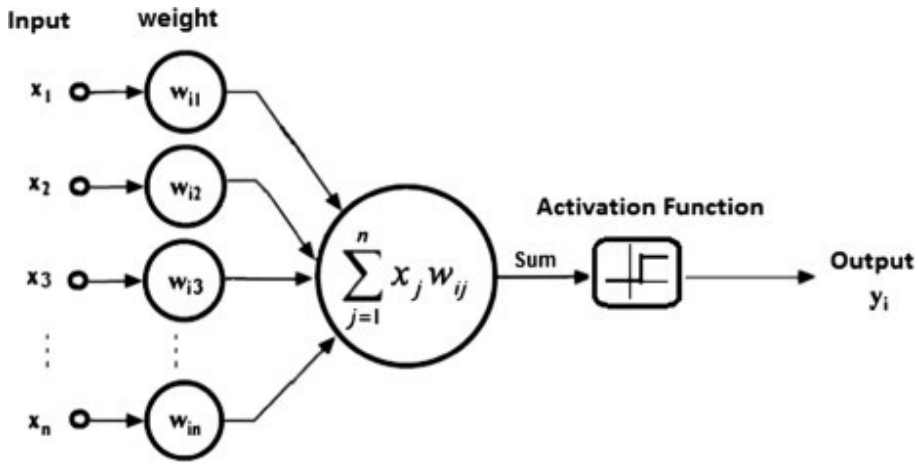

Figure 2.6: a graphic representation of Threshold logic unit of ANN (Source: [Shubh Saxena, 2017])

The process to obtain the net input value is a complex summation process as observed. Each input signal is a weighted combination of several input signals influencing in a different way the output signal. This weighted combination depends on a transformation or a function that each neuron applies to the weighted inputs, which could be linear or nonlinear [Castrounis, 2016]. ANN are capable of processing nonlinear relationships between inputs and outputs in parallel, due to the fact that ANN contains adaptative weights between neurons. Tis adaptative weights can be tuned by a learning algorithm that learn from the existing data aiming to improve the model using a cost function [Castrounis, 2016].

The feedforward neural network is the basic form of an ANN. It consists of an input layer, hidden layers and output layers as shown in figure 2.10. A function, $y=f(x;\theta)$, feds the input x to the output y by the action of several nonlinear transformations: $f(x)=(f_n, \ldots, f_1)(x)$. Each function f_k is a network layer and consists of a linear transformation of the previous output, followed by a nonlinear funcion: $f_k = \omega_k(\theta_k^T f(k-1))$. ω_k is the nonlinear function which is typically a sigmoid function or functions, θ_k are matrices of numbers, the model weights. In the training phase, the model receives input from the training dataset and it is taught to make predictions at the output layer that matches the known labels. Each component of the network produces expedient representation of its input. It is supposed to learn how to achieve the perfect representation of the data, the

best prediction in the output layer. The training phase performs the weights optimizing for the output layer which is done by the use of an optimization algorithm, typically, on a cost function [Lundervold and Lundervold, 2019].

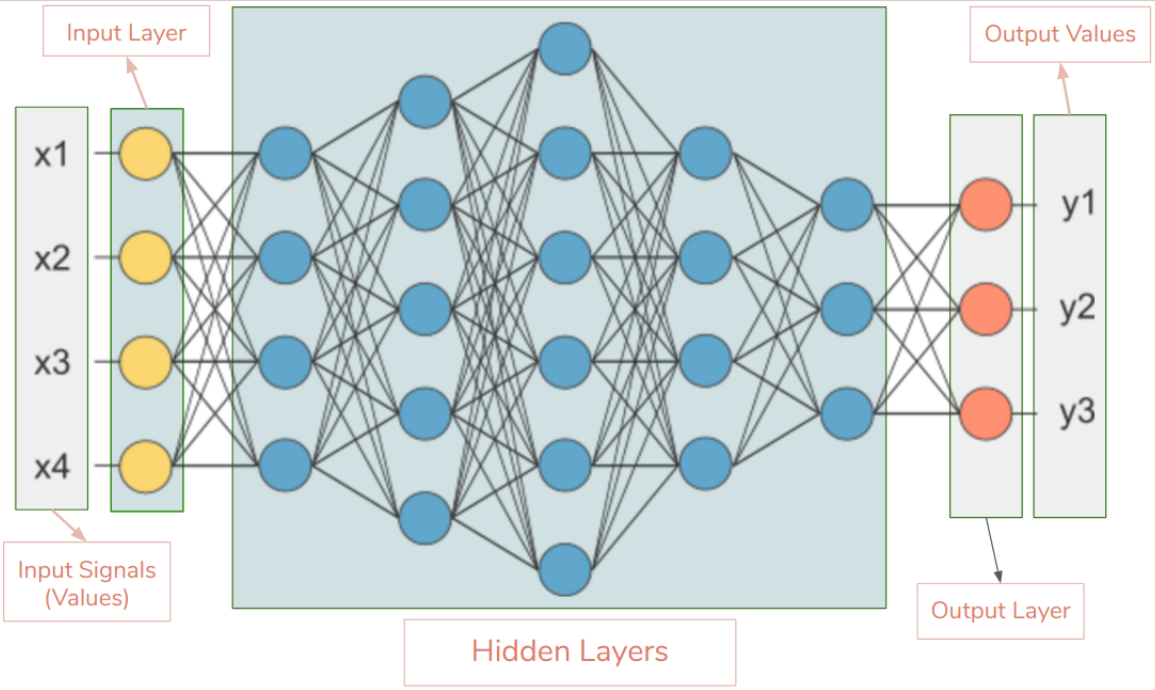

Figure 2.7: A graphic representation of ANN architecture with an input layer, hidden layers and the output layer (Source: [Raven, 2017])

ANN, architecturally examined, refers to a model using layers of artificial neurons, an input layer, an output layer and hidden layers, these layers are connected to each other. The earliest neural network architectures consisted in an input layer directly connected to the output layer, the perceptron, however, this only solved linearly separable patterns, for more complex problems or patterns it was necessary to add one or more hidden layers. Each layer contains one or more neuron [Castrounis, 2016]. How these layers are connected, the grouping of neurons on each layer, the summation and the activation or cost function are crucial to the better work of the neural network [Sumathi and Paneerselvam, 2010]. A simple model contains one hidden layer, however, the more complex the problem is to solve and abstract, more complex models are needed. In this case more hidden layers are used to build the ANN architecture. To achieve more complex models, it is possible to increase the number of hidden layers, number of neurons in each layer or the number of connections. Overfitting might be a problem than can occur due to the increase of model complexity [Castrounis, 2016].

ANN Models could be tuned and characterized by the activation function, several transformations might be used as cost or activation function [Castrounis, 2016]

Training an Artificial Neural Network .

Once the network is structured for a specific application the next step is to train the model beginning with the selection of a random initial weights. The thesis focus relies on Supervised training which provides the network with the desire output. The basic idea is related to the update of the weight values by performing iterations. In each iteration the weights between a selected neuron are updated thanks to a learning function minimizing the neural network error rate to an acceptable level [Sumathi and Paneerselvam, 2010, pag 52]. The artificial neural network is made of nodes or neurons and connections including inputs, outputs and hidden neurons [Kruse et al., 2016]. The weights between the hidden layers can be seen as a matrix [Kruse et al., 2016]:

$$W = \begin{pmatrix} w_{u1,v1} & w_{u1,v2} & \cdots & w_{u1,vm} \\ w_{u2,v1} & w_{u2,v2} & \cdots & w_{u2,vm} \\ \vdots & \vdots & \ddots & \vdots \\ w_{un,v1} & w_{un,v2} & \cdots & w_{un,vm} \end{pmatrix}$$

.

This ability of learning is peculiar for intelligent systems, which is considered as the update of internal variables as a response of against external stimuli performing a specific task. This process includes the modification of the architectures of the neural network, weight modification, creation of connection links, changing the firing rules of individual networks. This learning rules are responsible for the modification of weights values in each iteration or epoch [Castrounis, 2016].

Artificial neural networks can be graphs with many type of structures, acyclic or cyclic, directed or undirected [Mitchell, 1997, pag 83]. If the neural network is acyclic, it is a feed forward neural network (without loops), in the contrary if it is with loops it is recurrent neural networks [Schnalzer, 2018]. The most common approach is the one based on backpropagation algorithm [Mitchell, 1997, pag 89].

To solve the learning problem and achieve the desired output, the delta rule is an algorithm that is mostly used. It uses gradient descendent to find the weights that fits the best the training examples [Mitchell, 1997, pag 89]. Multiple layers of linear units produces only linear functions, however we need representations of highly nonlinear function, therefore we need a differentiable threshold unit, sigmoid unit [Mitchell, 1997, pag 96]. The backpropagation algorithm aims to minimize the squared error between the networks

output values and the target output values [Mitchell, 1997, pag 104]. The backpropagation algorithm is the mostly used learning technique and it is appropriate for problems with the following characteristics [Mitchell, 1997, pag 104]

- Instances are represented by many attribute-value pairs
- The target function output may be discrete-valued, real-valued or a vector of several real- or discrete-valued attributes
- The training examples may contain errors-
- Long training times are acceptable
- Fast evaluation of the learned target function may be required
- The ability of humans to understand the learned target function is not important.

The backpropagation algorithm starts with random weight value and input values. In the forward phase, it is achieved an output vector at the end of the network, the algorithm compares the output vector with the desire output, this error is used to adjust the weight value of the output layer and to the previous layers in a backpropagation phase until the input layers. The process is carried out in each epoch with the training dataset minimizing the error until a threshold [R. Rojas, 1996]. The selection of the finalization criterion is important, few iteration can be insufficient and too many iterations can lead to overfitting [Mitchell, 1997, pag 108]. In order to speed the convergence, sometimes it is added a momentum [Mitchell, 1997, pag 100].
This machine learning technique has proven to be successful in several fields such as handwritten characters recognition, spoken words recognition and faces recognition [Mitchell, 1997, pag 81]. To solve more complex problems and to speed the process, machine learning serves as the basis for deep learning explained in the next section of this chapter.

2.6 Deep Learning Architectures

In "classical" Machine learning techniques, the feature extraction is done manually, which is difficult and time consuming. By applying deep learning, features are learned automatically from the data, the main focus of deep learning architectures is feature learning, automatically learning representations of data [Lundervold and Lundervold, 2019].

Convolutional neural networks .

CNN is a case of ANN where the network maintains spatial information of the data with a few connections between the layers. The input of this CNN is arranged in a grid structure and then fed through layers that preserve these relationships, each layer operating on a small region of the previous layer. The CNN architecture consists of several convolution and activations layers with pooling layer in the between. It is trained by backpropagation and gradient descent method. It has fully connected layers at the end, each compute the final outputs [Lundervold and Lundervold, 2019]. The basic architecture consists on: input image, convolutional layers, pooling layers and a fully-connected layer. The network takes the input images and it applies the convolutional layers. This consists on applying kernels (filters) to extract features (edges, lines, gradient orientation, etc). The result is a set of feature maps. The next step is applying pooling layers to reduce spatial size which leads to a decrease in computational power requirements. This step allows the extraction of dominant features and reduces the number of neurons. The action of the convolutional layer and the pooling layer is a convolution. It is possible to apply several convolution. In each further convolution the feature maps allows the extraction of more complex features. Finally a fully-connected layer connects the output of the convolutions to an output layer, where the number of neurons is equal to the number of classes [Bagnato, 2018].

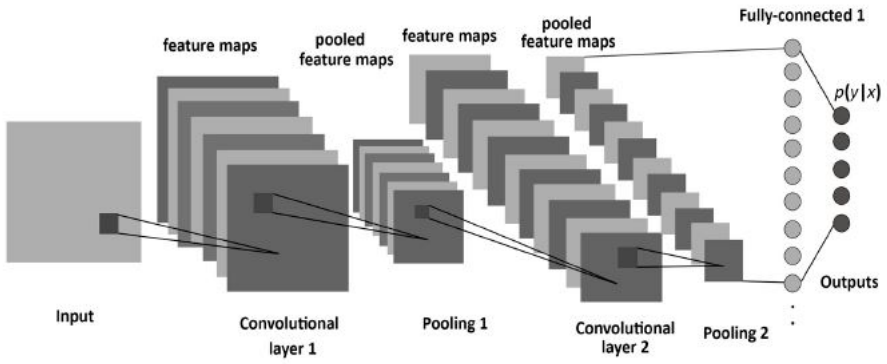

Figure 2.8: A graphic representation of CNN architecture (Source: [Albelwi and Mahmood, 2017])

- Convolutional layers: composed by a set of kernels, filters, to extract features from the input data. The result of the application of this kernels are the feature maps [Albelwi and Mahmood, 2017]. Each filter share the same weight values in the input domain, weight-sharing, so that features that appears in one part os the image, are likely to appear in other parts also [Lundervold and Lundervold, 2019]. Each neuron

is connected to local regions in the feature map of the previous layer through a set of weights [LeCun et al., 2015].

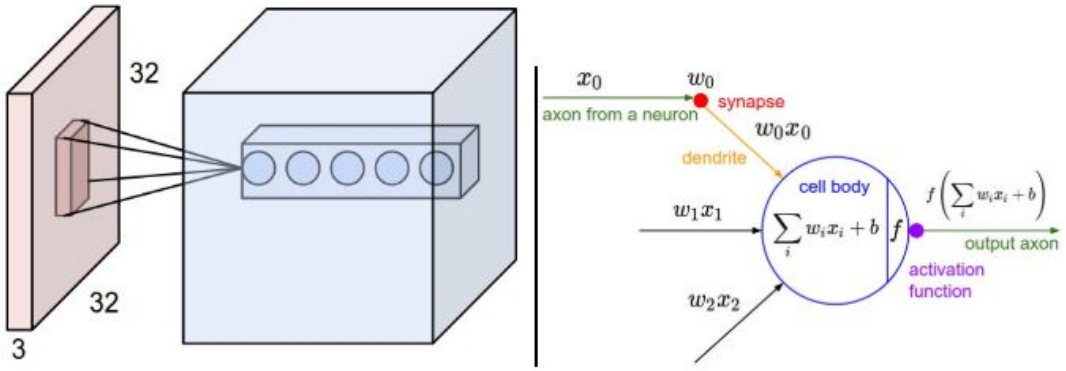

Figure 2.9: A graphic representation of a the action of a convolutional layer (Source: [Stanford, 2019])

- Activation layer: to introduce nonlinearity in the model it is introduced a non-linear activation function, normally ReLUs, producing new tensors.
- Pooling: the main goal is to merge semantically similar features into one [LeCun et al., 2015]. It takes regions as input and produces a single output for each region. This output might be selected by max-pooling or average-pooling. The pooling layers offer the CNN invariance [Lundervold and Lundervold, 2019].

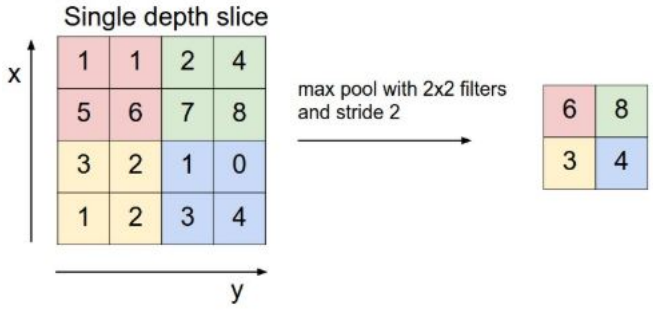

Figure 2.10: A graphic representation of a max-pooling (Source: [Stanford, 2019])

The final layers are fully connected layers used to find the patterns that were not possible to be captures with parameter-sharing convolutional layers [Vaidya and Paunwala, 2019].

Chapter 3

Evaluation of Deep Learning architectures in skin wound detection

In the following chapter the most commonly used deep learning architectures and frameworks are summarized and described.

3.1 Pretrained Models

Usually, building a model from scratch is not an easy task, it is time consuming, requires a huge amount of labeled data and computational restriction might be presented, for this reason using a pretrained model is an option to learn algorithms and use an existing framework [Techlabs, 2018]. The idea is to use a pretrained network that has already extracted important features from natural images as a starting step towards the new task. This pretrained models are normally trained on a subset of the ImageNet database used in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)[The MathWorks Inc., 2019e]. Pretrained models face important benefits such as the time, amount of data and computational requirements and the simplicity of its incorporation [Shao, 2019]. [The MathWorks Inc., 2019e] aprovides various pretrained networks available on ImageNet. In table 3.1 it is represented a list of the different pretrained networks with a brief description and the Layers architecture is represented.

Architecture	Description	Layers
LeNet	Proposed by [LeCun et al., 1998]. The first CNN, developed for hand digits in documents, it is a feed-forward neural network. Gradient-based learning method.	7 Layers (with no inputs):5 alternating layers [convolutional and pooling]. 2 fully-connected layers
ResNet	Deep Residual learning, proposed by [He et al., 2016]. Proposed to train deeper nets, for image recognition.	152 Layers
SegNet **[Badrinarayanan et al., 2017]**	Encoder-Decoder architecture. Developed for Semantic Image Segmentation.	13 convolutional layers in the encoder network with the corresponding decoding layers.
VGG	Proposed by [Simonyan and Zisserman, 2014] in 2014. A convolutional network for image recognition It presents two variantes (VGG16 and VGG19).	16 or 19 weight layers
AlexNet **[Krizhevsky et al., 2012]**	Deep convolutional neural network for ImageNet classi cation.	5 convolutional Layers (max-pooling) ReLus 3 fully connected Layers
GoogleNet **[Szegedy et al., 2015]**	Deep convolutional Neural Network. Developed for image recognitionThe winner of the 2014-ILSVRC competition. Also known as Inception-V1. Goal: high accuracy with small computational cost.	22 Layers No fully-connected layers
U-Net **[Ronneberger et al., 2015]**	CNN trained end.to.end and developed for biomedical image segmentation.	23 convolutional Layers

Table 3.1: The table summarizes the commonly used deep learning architectures according to literature research such as [Garcia-Garcia et al., 2017]. LeNet is the first CNN-based architecture developed [LeCun et al., 1998] in 1998, and it is the the basis for further development of CNN architectures.

In order to have a goos final results, it is important to take into account how similar the data of interest is to data on which the model was trained. In this case, and after an evaluation it was not possible to find a pretrained model appropriate to handle wound images, this is one of the limitations of this project.

3.2 Deep learning network architectures

Each year new competitions give space to new deep learning architectures to be developed in order to solve some specific problems. Several articles and electronic sources made a review on deep learning architectures applied to semantic segmentation, such as [Garcia-Garcia et al., 2017] and [Le, 2019]. The most relevant deep learning techniques are sumarized in table 3.2. AlexNet, VGG-16, GoogleNet and ResNet have made significant contributions to the field of computes vision, they are used as the basis of semantic segmentations systems [Le, 2019]. [GitHub Inc., 2019a] provides a list of deep learning techniques for semantic segmentation, being possible to download pretrained models available for Keras, Caffe, Torch and TensorFlow [Schnalzer, 2018].

3.3 Commonly used Frameworks for deep learning

As mentioned, deep learning is an important area of machine learning, however, we are challenged with techniques that require time to get a reasonable result for the assigned task and sometimes it is not possible to afford having this amount of required time. Therefore, it is necessary to find new possibilities. Deep learning frameworks are sophisticated software packages in deep learning published as open sources [Zacharias et al., 2018]. Those frameworks provide interfaces or libraries that enbale researchers to build fast-deep learning model using pre-built components [Sharma, 2019]. Several toolkits and libraries are available to improve the development of deep learning algorithms applied to object detection in medical images [Erickson et al., 2017]. To evaluate a deep learning framework, it is important to take into account some key features as mentioned in [Erickson et al., 2017] and[Sharma, 2019]:

- optimized for performance
- Simple code language for an easy understanding
- Graphical Processing Units support to increase the learning rate.
- Execution and training speed. Parallelize the processes to reduce computations.
- Automatically compute gradients
- Maturity level
- GitHub commits/contributors

These parameters are set by previous projects, however for each independent project the library choice might change depending on the needs. [Zacharias et al., 2018] summarized the most popular deep Learning frameworks based on GitHub metrics: TensorFlow, Keras, Caffe, MXNet, Theano, Deepleatning4j, CNTK, PyTorch, etc. In this project the focus will be on implementing pretrained models from Keras and Caffe into MatLab environment, since with Matlab tool it is possible to import networks from TensorFlow-Keras, Caffe and the ONNX model format [The MathWorks Inc., 2019e].
On GitHub [GitHub Inc., 2019b] it is possible to find an important amount of pretrained models for Keras or Caffe. In this Master Thesis one approach option was to import model from Caffe and Keras in the MatLab framework, however it was not necessary. In figure 3.1 it is represented some of the available and commonly used architectures in deep learning and also it is represented some of their properties such as the Depth (largest

number of fully connected layers on a path from the input layer to the output layer), size, parameter and image Input Size.

Network	Depth	Size	Parameters (Millions)	Image Input Size
alexnet	8	227 MB	61.0	227-by-227
vgg16	16	515 MB	138	224-by-224
vgg19	19	535 MB	144	224-by-224
squeezenet	18	4.6 MB	1.24	227-by-227
googlenet	22	27 MB	7.0	224-by-224
inceptionv3	48	89 MB	23.9	299-by-299
densenet201	201	77 MB	20.0	224-by-224
mobilenetv2	53	13 MB	3.5	224-by-224
resnet18	18	44 MB	11.7	224-by-224
resnet50	50	96 MB	25.6	224-by-224
resnet101	101	167 MB	44.6	224-by-224
xception	71	85 MB	22.9	299-by-299
inceptionresnetv2	164	209 MB	55.9	299-by-299
shufflenet	50	6.3 MB	1.4	224-by-224
nasnetmobile	*	20 MB	5.3	224-by-224
nasnetlarge	*	360 MB	88.9	331-by-331

Figure 3.1: The table extracted from [The MathWorks Inc., 2019e] illustrates the available pretrained models on ImageNet and some properties (source: [The MathWorks Inc., 2019e])

Chapter 4

Thesis Results

The following chapter presents the thesis outcome related to wound detections using MatLab based on neural networks. Prerequisites and installation steps are listed and usability and limitations are discussed.

4.1 General information and issues on machine learning tasks

A specific workflow is followed in order to build a machine learning model: problem formulation, data collection, correct analysis of this data, model construction, training the model, model validation and evaluation of potential errors as mentioned in [Wang et al., 2017] and [Foster et al., 2014]. For semantic segmentation the workflow based on machine learning would be similar to the one illustrated in figure 4.1. An important step is to detect potential errors, since each step of the listed-on figure 4.1 can produce biases and errors. Model selection is an important step that needs to be carried out depending on the size and type of dataset, problem category and further criteria [Wang et al., 2017]. In addition, [Mitchell, 1997] suggest as crucial aspects to be aware of while training a machine learning models, generalization, overfitting and stopping criterion. Model generalization is referred to the ability of the trained model to predict or classificate of unknown (or previously not used) dataset and not only the data used in the training step. Overfitting is another important aspect to take into account and occurs when the algorithm in use fits exactly or perfectly a set of data and predicts with errors when it is used on a different dataset. If this occurs, the model is biased to the data. This may occur if the dataset is small or the parameters to train the model are set incorrectly. Overfitting is closely related to the generalization problem, since, if the model overfits, then it may occur with a bad generalization. In order to solve this problem, it is recommended to use a reasonable

size of dataset and if not, to use data augmentation. Data augmentation is a process that enlarges the dataset with similar data that is created from the information in the existing dataset, by the use of several transformations such as rotation, translation, blurring, wrapping, scaling and further modifications to the existing images [Lemley et al., 2017]. The selection of a suitable network is an important step, however suitable data is another key step in machine learning, to achieve correct results [Wang et al., 2017]. Finally, the computational power to use in the process is another important aspect while training a neural network, especially using deep learning architectures.

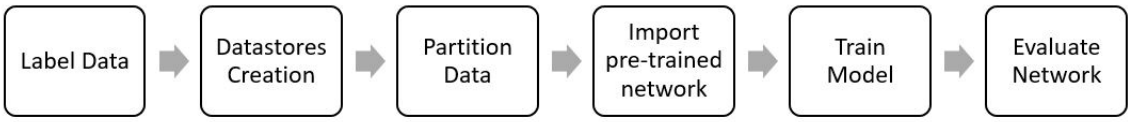

Figure 4.1: .
The image illustrates the steps to be followed for semantic segmentation based on machine learning. Important steps include data labelling, data separation and training/validating the network or model. In order to build this workflow, a previous research was done in [The MathWorks Inc., 2019c].

4.2 Data Preparation

During research for a suitable dataset, several datasets for machine learning can be found available on the different platforms that World Wide Web provides. As mentioned previously, it is crucial to find a correct and appropriate dataset in machine learning. In the case of this thesis, the data set used was provided by the Institute of eHealth, that was collected from the available machine learning platforms in medical image databases. Several datasets and websites were examined in detail [Schnalzer, 2018]:

- ImageNet [Lab, 2018]
- MNIST database [LeCun et al., 2019]]
- Kaggle [Kaggle Inc., 2019]
- UCI Machine Learning Repository [Dua and Graff, 2017]
- CIFAR-10 and CIFAR-100 [Krizhevsky, A, 2019]
- deeplearning4j.org [Skimind, 2019]

- deeplearning.net [deepnearning.net, 2019]
- Google Open Images Dataset [Krasin et al., 2018]
- Caffe datasets within the Model Zoo

The images for wound detection were selected out of several datasets, the next step was to label the images obtained and proceed to use them for training and test process. It was important to partition the data in order to have a dataset for training the selected model, and dataset for the model testing. In this thesis the recommendation in [Schnalzer, 2018] for data distribution were followed, in figure 4.3 it is illustrated the distribution of data to use in this Thesis. 80% of the data was taken for training purpose and the 20% remaining should be reserved for testing. Moreover, out of the 80% for Training data, 16% were reserved for validation. This splitting refers according to recommendation mentioned in [Schnalzer, 2018]. To achieve the goal of this thesis, wound images were taken from different online available sources: [Government, 2018], [Code 3 Emergency Partners, 2016], [Medetec, 2018], [LLC, 2018], [the GAP, 2018], [Trust, 2018], [LTD, 2018], [Team, 2017], [WoundEducators.com, 2018], [U.S. National Library of Medicine, 2018], [Galderma S.A., 2018], [Cadogan, M., 2018], [Medicine, 2013], [Nigeria Galleria, 2018] and [Queensland University of Technology, 2018a]. It is worth mentioning that Medetec [Medetec, 2018] and DermNet NZ [Trust, 2018] were the libraries that mostly provided a collection of free of stock and copyright free images of all wound types with the only restriction of keeping the copyright from the provided images [Schnalzer, 2018]. In the database of images used in this thesis, there are images of ulcers, laceration, burn and scald wounds included. Melanomas were excluded, since they are not categorized as classic wounds. After the analysis of the databases and the selection for the images, 251 fitted the purpose of the thesis and the next step was to split these images into training data, validation data and testing data randomly. Figure 4.2 shows an example of the wound images included in the research.

Figure 4.2: An image of the dataset used for the development of the research

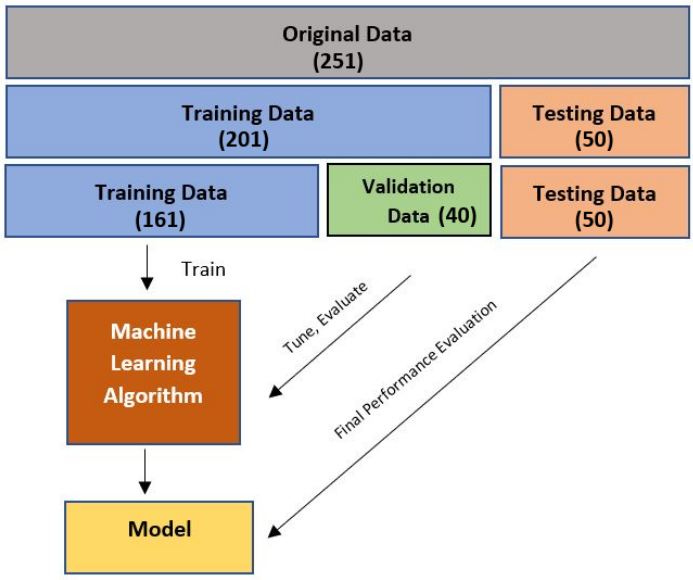

Figure 4.3: The image illustrates a recommendation for data partitioning into training and test sets. Also, training data is splitted into training data and validation data to avoid overfitting and further errors. The image has been done manually based on several recommendations read in the literature. Finally, it was chosen the splitting recommendation in [Schnalzer, 2018].

It was not possible to train the data with the entire dataset due to hardware restrictions. Therefore, a smaller subset was selected, established of 40 images from the training set and 20 images form the validation set leaving the Test subset fot Test purpose.

4.3 Training approach with Matlab

A full training approach needs an amount of data that rises to 1000 images per each class, which is not the case of this study since our dataset consists only of 251 images and also the hardware resources are limited. Therefore, there is a lack of available data to perform a full training, and for this reason solutions need to be found. [Lee et al., 2017] suggests fine tuning as a solution to reduce the necessary amount of data which would increase training speed. [Tajbakhsh et al., 2016] and [Anderson et al., 2016] identified fine tuning machine learning networks as a significant method to train neural networks on small amount of data. In addition, [Gando et al., 2016], in the study developed, suggests a better DCNN in effectiveness while using fine tuning. Fine tuning is based on the idea of using a pretrained neural network that solves a similar problem to the one of interest as an initial step. This pretrained model is modelled to fit our problem and trained with the new data of interest and since the network has already learned universal features, the process will be faster [Queensland University of Technology, 2018b].

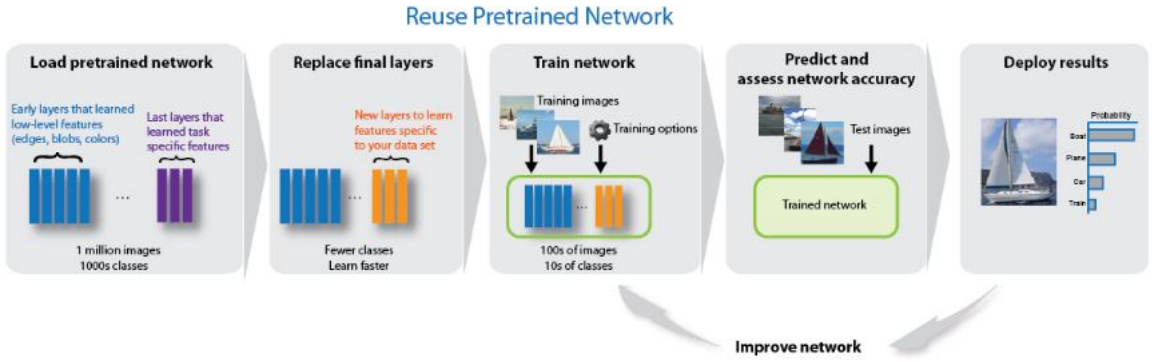

Figure 4.4: The image, originally taken from [The MathWorks Inc., 2019h], represents a scheme about how transfer learning is carried out. The idea behind transfer learning is to replace final layers of a pretrained model with new layers while freezing first layers. The training is performed only with the new layers, which increases the speed of the network training.

In the case of the present thesis, no specific pretrained model was available for wound detection, it was not an expectation to achieve excellent results by applying fine-tuning. Transfer learning is a machine learning method that allows to start training from patterns that have been already learned while solving different problems and avoid starting from scratch [Marcelino, 2018]. Fine-tuning a network with transfer learning is a method to achieve a faster training of the network using a small amount of data. In figure 4.4 it is

represented the basic idea behind transfer learning. The idea is to replace the final layers of a pretrained model with new layers in order to use the network as a starting point for a new task, while the initial layers will be frozen. The training only will be applied to the new layers, leaving the frozen layers untrained [The MathWorks Inc., 2019h]. As mentioned, in the case of this thesis the amount of data is not enough for a full training, for this reason, as recommended in the literature, transfer learning with pre-trained models and semantic labelled data was a reasonable approach [Schnalzer, 2018]. The operating system of the computer in use was windows, which was enough for MatLab training. The goal in this research is to re-train models in MatLab with data of interest, wound images, and observe which pretrained models performs better. For this reason, the first step was to based on literature the best pre-trained model for semantic segmentation and try each of them with a small subset of data. For time limitations and computational limitations, transfer learning was a good initial option to obtain results rapidly and enable to study if the selected pretrained models are valid or not for the goal of this thesis. In MatLab, the *Neural Network Toolbox* allows integration of pretrained models. Furthermore, this toolbox allows the definition of various machine learning models. In this thesis it was used with latest released version of Matlab, R2019a version.

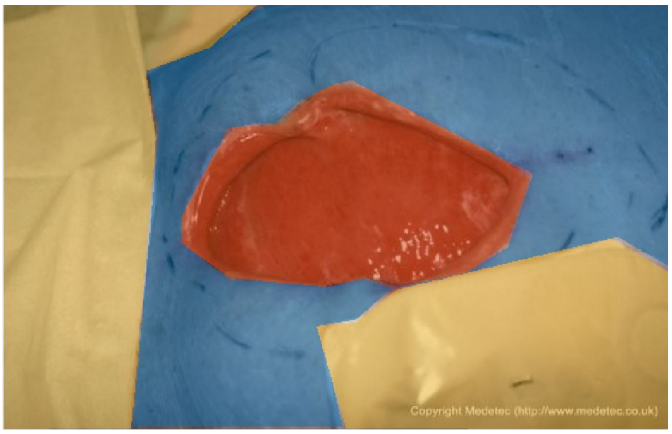

Figure 4.5: Example of a manually semantic labeled image with the image labeler App in MatLab. The original image was extracted from [Code 3 Emergency Partners, 2016].

The first step was related to image labelling manually, since semantic segmentation is a supervised machine learning technique. To carry out the image labelling, the Image Labeler App in MatLab [The MathWorks Inc., 2019a] was used to define ground truth for image collections. Figure 4.5 illustrates an example of hand-crafted image labelling using the Image Labeler App of Matlab, using polygons and brush for better final definition of

labelled areas. The images were labelled in MatLab with three classes: Skin, Wound and Background. MatLab ground Truth object allowed storage of ground truth definitions containing information about data source, labelled pixels and label definitions. MatLab allows the integration of this ground truth information easily to a code by providing the correct format, and this ground truth might be exported as labels and saved to a file system. It is possible to import networks architectures from TensorFlow, Keras, Caffe and the ONNX model format, moreover, it is possible to export trained networks to the ONNX model format. To import Pre-trained networks from Keras MatLab offers the function *importKerasNetwork* and *importKerasLayers*. To import networks architectures from Caffe, the MatLab functions would be *importCaffeNetwork* and *importCaffeLayers*. It is possible to import the networks with or without weights. The ONNX allows the interaction between different deep learning frameworks that support ONNX models, MatLab functions are *exportONNXNetwork* and *importONNXNetwork* [The MathWorks Inc., 2019e]. For Caffe pre-trained Models, it is possible to use the following Matlab Code, based on [Schnalzer, 2018]:

```
% MatLab code to import Caffe pretrained Model with Matlab Add-On
% Import segNet based on VGG-16 from Caffe. Function requires Deep Learning
    Toolbox Importer for Caffe Models
segNetAlexCaffeProtofile= 'C:\Users\chaim\Desktop\TFM\Bildmaterial_MatLab\
    caffe-model-zoo-master\VGG16\deploy.prototxt';
segNetAlexCaffeModel = 'C:\Users\chaim\Desktop\TFM\Bildmaterial_MatLab\
    caffe-model-zoo-master\VGG16\VGG16.caffemodel';
%% Without Weights
CaffeLayers = importCaffeLayers(segNetAlexCaffeProtofile);
%% With weights from pre-trained model
net = importCaffeNetwork(segNetAlexCaffeProtofile,segNetAlexCaffeModel);
% InputSize of Image
inputSize = net.Layers(1).InputSize;
```

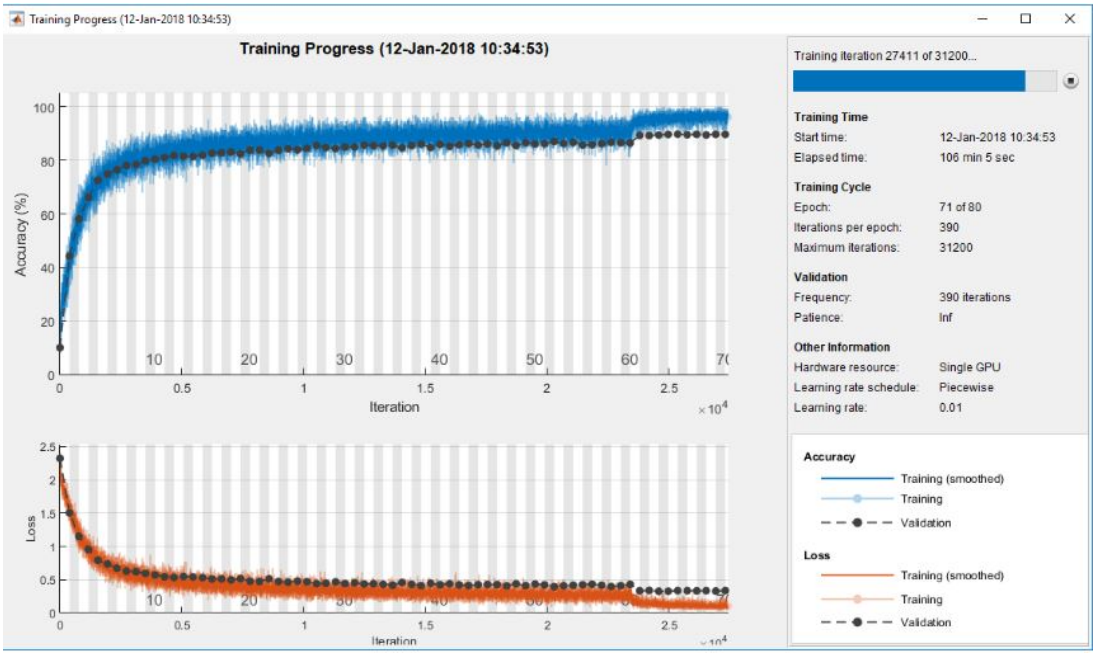

Figure 4.6: The image obtained from [The MathWorks Inc., 2019d] illustrates the display for training information progress while training a network. It plots several metrics during training to see training progress. In this case it is possible to observe accuracy improvement monitored in each epochs during network training.

MatLab offers the possibility to monitor the training progress continuously as shown in figure 4.6. In addition, it is possible to model new architectures and visualize these architectures in order to improve understanding of semantic segmentation task among others. This is possible thanks to several features such as layer representation and object-oriented network customization of deep learning. To achieve the goal of this research a MatLab script was developed in order to use pre-trained Models available in MatLab to perform training and evaluation of the model re-trained with the new dataset of wound images. In order to get a coherent result, images had been resized to 224x224 to fit the first layer of the pre-trained model in use. Also, data was split in training and test data with ImageLabelDatastores and PixelLabelDataStores. Moreover, data-augmentation was possible thanks to imageDataAugmenter object provided by MatLab. Augmentation was performed with reflection and X as well as Y translation [Schnalzer, 2018]. Finally, an important step before training was the setting of training options. Table 4.1 summarizes the most relevant training options mentioned in the MathWorks platform. The following pretrained Models were used: SegNet base on VGG-16 and VGG-19, ResNet and AlexNet. Transfer Learning is needed for time and computational limitations.

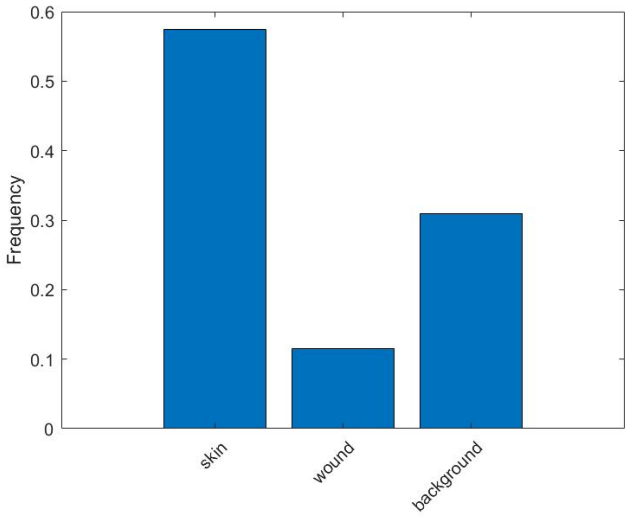

Figure 4.7: Based on [The MathWorks Inc., 2019f] script, the image shows the pixel counts per each class.

The script for all the approaches tried in this research was made in the following order: First the images were labeled (the training images and validation images), and image-datastore were created for the images and the labeling data. The images and label data were resized, and the classes of interest were loaded. In order to test if the images and the labeling data were leaded correctly, imshow (MatLab function) was used to plot and overlap of the image and the labeling. At this step a problem with the manes of image and label folders was faced, therefore, the images had to be renamed. The next step was to analyze the data statistics, figure 4.7 shows the pixel counts per class. The next step was to split the data into Training set and validation set and create the network. After network creation, it was necessary to create a pixel classification final layer due to the purpose of this Master Thesis. The training options were set, and the data Augmentation was used to provide more examples to train the network. Finally, before starting the network training we proceeded to freeze some layers in order to decrease the time elapsed. The steps followed were according to [The MathWorks Inc., 2019f].

Training Options	Description
SolverName	Solver for network training
Momentum	Parameter update from one iteration to the next iteration. It is related to the stochastic gradient descent method between iterations.
InitialLearningRate	Network Learning Rate
L2Regularization	Regularization value (weight decay) for the weights added to the loss functions
LearningRateSchedule	Option for dropping learning rate in training
ExcecutionEnvironment	to set a CPU or GPU option
MaxEpochs	Maximum number of training Epochs
MiniBatchSize	Subset of training dataset to use in each training
Verbose	Display progress about training information
VerboseFrequency	Specifies settings for training progress information
\ValidationData	Validation data for training usage
ValidationFrequency	Frequency of network validation in number of iterations
Shuffle	For data shuffling in each iteration
CheckPointPath	Path to save trained network after epochs
Plots	Plot the progress of the training

Table 4.1: Training options for Deep learning architectures in MatLab. The parameters are extracted from [The MathWorks Inc., 2019g].

4.3.1 SegNet Approach

Matlab offers initialization of SegNet layers with a preferred encoder depth or pre-trained weights of VGG16 and VGG19 deep learning architectures [Schnalzer, 2018].

```
%%%% Matlab Code for SegNet initialization %%%%
% Create SegNet Layers
imageSize=[224 224 3];
numClasses=3;
%% set encoder depth manually, Modelds with endocerdepth=5 are specified as
    vgg16 or vgg19
encoderDepth=1;
%% Initialize segnet with customized architecture
lgraph=segnetLayers(imageSize, numClasses, encoderDepth)
%% Initialize segnet with pre-trained architecture
lgraph=segnetLayers(imageSize, numClasses, 'vgg19')
%% DataAugmentation and Create a pixel label image datastore for training
    network
augmenter = imageDataAugmenter('RandXReflection',true,...
    'RandXTranslation', [-10 10], 'RandYTranslation',[-10 10]);
datasource = pixelLabelImageSource(imds,pxds,'DataAugmentation',augmenter);
```

```
%% Set training Options
options = trainingOptions('sgdm', ...
    'MaxEpochs',150, ...
    'MiniBatchSize',9, ...
    'InitialLearnRate',1e-3, ...
    'Verbose',false, ...
    'Plots','training-progress');
%% Train the Network
net = trainNetwork(datasource,lgraph,options)
```

Different architectures with different encoder depths are illustrated in figure 4.8. The model was trained by setting different options, changing epochs and Minibatch size. To evaluate the model, it was calculated the accuracy, Intersection over Union (IoU) and timing execution. The election of this specific Parameter or Metrics to evaluate the Neural network was according to several literature articles such as [Liu et al., 2018], [Saffar et al., 2018] and [Oršić et al., 2019], where it was mentioned that IoU, accuracy and excecution timing are the most common parameters in use.

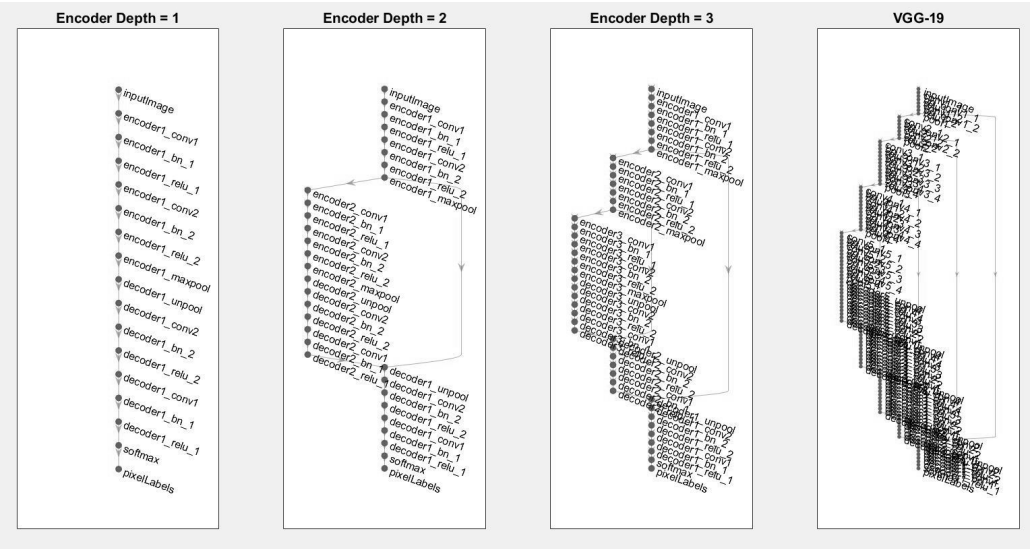

Figure 4.8: The image illustrates four different neural networks with different encoder depth

The accuracy is a metric to evaluate the percentage of correctly identified pixels of each class.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

IoU is used to evaluate performance in machine learning, object detection or semantic segmentation. The maximum value is 1 [Rahman and Wang, 2016]. The intersection over union is a metric that is calculated as the Overlap Area divided by the Union or combined Area [Sandeep, 2019]. It represents the ratio of correctly classified pixels to the total number of ground truth and predicted pixels in that class [The MathWorks Inc., 2019j]. The idea is summarized in the following equation:

$$IoUscore = \frac{TP}{TP + FP + FN}$$

In case the clases are disproportionally sized it is recommended to use weighted-IoU [The MathWorks Inc., 2019j].

MeanBFScore is, along with accuracy and IoU, an indicator that the used function to evaluate the network offers, it indicates how correct is the matching between the predicted boundary and the true boundary of each class. It is used when the need is to correlate better with human qualitative assessment than IoU metric [The MathWorks Inc., 2019j]. Although it was not found any evidence in literature about using MeanBFScore as an evaluating indicator, it was found interesting to include in the final results since the evaluation algorithm used offers the results for the indicator aloGn with accuracy and IoU. An example of validation parameters of a model is illustrated in table reftab:segnet, where it is represented the result of running the developed model in MatLab with different chosen parameters.

The first attempts with segNet to retrain pre-trained models were not consistence in results. In table 4.2 it is summarized the results for 3 attempt with segNet using as Model pretrained models VGG16 and VGG19. It was observable that the timing with few iterations was long and the results of the evaluating parameters were not good enough. For these reasons, the next idea was to run the same model with a small Mini Batch size and to improve Accuracy run the model for more Epochs and freeze less Layers. The results of some attempts for VGG16 and VGG19 are collected in tables 4.3 and 4.4 respectively. The results improved; however, it is important to take into account that the model is being overfitted because of this small amount of dataset. Therefore, it would be interesting, as a future work, to run the model with a normal Minibatch Size for a higher Epoch value. This would take a considerable time; it would be interesting to have more CPU to help the process.

	Mini Batch	Freezed Layers	Epoch	Time	Global Accuracy	Mean Accuracy	Mean IoU	Weighted IoU	Mean BFScore
VGG 16	***30***	***60***	***150***	1:46:18	0.456	0.429	0.263	0.306	0.117
VGG 19	***30***	***60***	***150***	2:31:52	0.467	0.527	0.309	0.308	0.143
	30	***35***	***100***	1:56:53	0.525	0.585	0.367	0.363	0.155

Table 4.2: Table summarizes the results obtained in the first attempts training the different SegNet networks with encoder Depth 5, 'VGG16' and 'VGG19'. The parameteres illustrated are the one selected for evaluation, and also the parameteres that werechanged in each trainig with the correpsonding values are represented.

MiniBatch	Freezed Layers	Epoch	Time Elapsed	Global Accuracy	Mean Accuracy	Mean IoU	Weighted IoU	MeanBFScore
11	***30***	***150 (450It)***	***2:33:16***	0.71426	0.56475	0.41615	0.54991	0.21165
15	***35***	***200 (400It)***	***3:15:33***	0.7082	0.56623	0.41491	0.54726	0.20188
20	***35***	***500 (1000It)***	***9:56:47***	0.74126	0.56214	0.42786	0.57219	0.29905

Table 4.3: Table summarizes the results obtained for different SegNet network trained on 'VGG16'. The parameteres illustrated are the one selectoed for evaluation, and Also it is represented the parameteres that were changed in each trainig with the correpsonding values.

MiniBatch	Freez Layer	Epoch	Time Elapsed	Global Accuracy	Mean Accuracy	Mean IoU	Weighted IoU	MeanBFScore
15	***30***	***200 (400It)***	***4:09:42***	0.64737	0.57912	0.41284	0.44891	0.2299
11	***35***	***150 (450It)***	***05:35:52***	0.6685	0.63349	0.47075	0.4981	0.20829
11	***35***	***250 (1000It)***	***07:38:43***	0.81238	0.70411	0.60542	0.67836	0.42835
20	***35***	***500 (1000It)***	***11:33:17***	0.80448	0.70579	0.60279	0.66794	0.4011

Table 4.4: Table summarizes the results obtained for different SegNet network trained on 'VGG16'. The parameters illustrated are the one selected for evaluation, and also the parameters that were changed in each training with the corresponding values are represented.

As we can observe in the different trainings, it is clear that increasing the number of Epochs improves the Accuracy value. Moreover, the Minibatch Size, has a clear influence in the results, since it affects negatively the time values, however, it improves the Accuracy of the network. For this reason, it is important to choose a suitable Minibatch size

according to the requirements of the training. In this case, the time was a limitation, for this reason, the network was trained on a small Minibatch size to prove the availability of the network as a semantic segmentation model. It is possible to perform further trainings with the same network, a higher Minibatch size and Epochs number. It is recommendable to set more than one CPU to make the process faster.
VGG19 proved to be better in the same conditions than VGG16, this is observable in table 4.2 where it the first attempt of both pre-trained models in same conditions is represented. The accuracy, IoU and MeanBFScore values, are superior for VGG19.
After training the networks, the validation data was used to validate the trained model. The Matlab code applied was the following:

```
% Testing the Network on One image
I = readimage(imdsTest,3);
C = semanticseg(I, net);
%Display results
B = labeloverlay(I,C,'ColorMap',cmap,'Transparency',0.4);
imshow(B)
pixelLabelColorbar(cmap, classes);
% Compare Results with excepted results in stored ground Truth
expectedResult = readimage(pxdsTest,3);
actual = uint8(C);
expected = uint8(expectedResult);
figure(5),
imshowpair(actual, expected)
% Measure the IoU with the jaccard function to measure the overlap per
% class
iou = jaccard(C,expectedResult);
table(classes,iou')
%% Evaluate Trainde Network - measure accuracy and MeanBFScore
pxdsResults = semanticseg(imdsTest,net, ...
    'MiniBatchSize',8, ...
    'WriteLocation',tempdir, ...
    'Verbose',false);
metrics = evaluateSemanticSegmentation(pxdsResults,pxdsTest,'Verbose',false
    );
% See metric in the overall network performance
metrics.DataSetMetrics
% To see the impact in each class separatly
metrics.ClassMetrics
```

4.3.2 ResNet Approach

In this part, the aim was to perform a training with ResNet pretrained Models. The depth of this architecture is 152 Layers and it address the degradation (with depth increase, accuracy might get saturated) problem in deep learning by introducing residual blocks or residual learning framework. The network consists of convolutional layers, it ends with a global average pooling layer and a 1000-way fully connected layer with SoftMax [He et al., 2016]. It is possible to observe the network in Figure 4.9, where the network adapted for semantic segmentation with MatLab plot function is plotted.

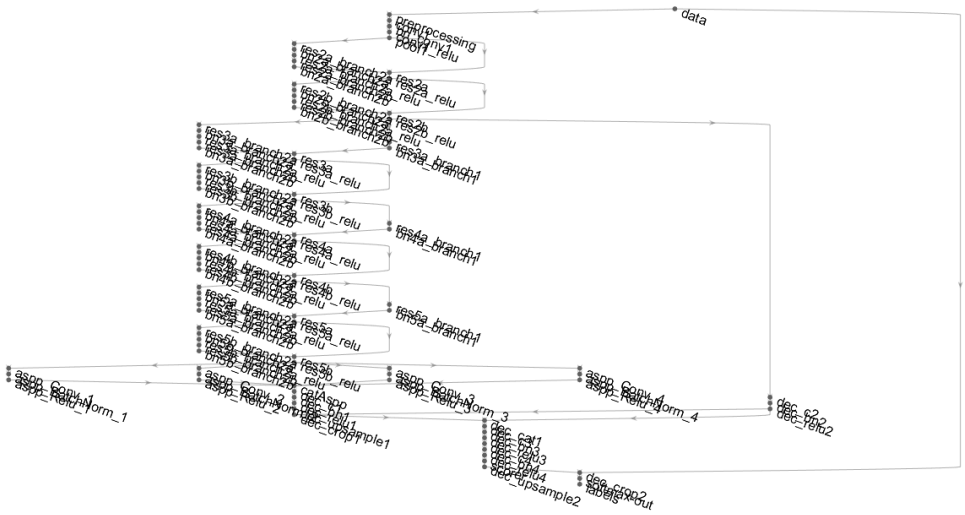

Figure 4.9: The image illustrates a plot of the pretrained model of ResNet neural network offered by MatLab.

The procedure was the same as in the previous subsection. As a first step a datastore was created for the images and the corresponding labels. The data partitioning in training and validation data was carried out and the training options were set. Finally, the network was created by using the ResNet network existing in MatLab thanks to the corresponding MatLab App used (Deep Learning Toolbox Model for ResNet-18 Network). In case of ResNet-18, changing some layers in order to adjust it for the aim of this master thesis, (semantic segmentation) were necessary. In addition, to reduce the time requirements, transfer learning was used, to freeze some layers. Once, the Pretrained network was re-trained on the new dataset of interest, the test data was used to evaluate the performance of this model. The used metrics are the same as in the previous approach, accuracy, IoU and MeanBFScore. The results with different training options and freezing different number of layer it is collected in table 4.5 for ResNet-18, ResNet-50 and ResNet-101.

	Freez Layers	Time	Global Accuracy	Mean Accuracy	Mean IoU	Weighted IoU	Mean BFScore
ResNet 18	***10***	56'	0.8381	0.78122	0.6770	0.7201	0.5358
	30	53'	0.8275	0.77256	0.6635	0.7047	0.5241
	50	51'	0.8274	0.7750	0.6642	0.7047	0.5255
ResNet 50	***30***	53'	0.8275	0.7726	0.6635	0.7047	0.5241
	50	51'	0.8274	0.775	0.6642	0.7047	0.5255
	85	40'	0.8312	0.7936	0.6732	0.7118	0.5210
ResNet 101	***30***	59'	0.8275	0.7736	0.6635	0.7046	0.524
	50	57'	0.8274	0.775	0.6642	0.7047	0.5255
	85	44'	0.831	0.794	0.673	0.712	0.521

Table 4.5: Table summarizes the results obtained for the different trained ResNet networks and the attempts with each network changin the correpsonding parameters. The parameteres illustrated are the one selected for evaluation, and also the parameteres that were changed in each trainig with the correpsonding values are represented.

By running the model several times with different Number of Epochs in some conditions it was realized that taking 200 epoch was a reasonable number since higher number would increase the time of training with no significant increase in the accuracy for example, for that reason the next attempts with different freezing layers were made taking 200 epochs. The same was made with the Minibatch size. The first attempts were training the network with different Minibatch size and it was observable that 30 was a reasonable value, since increasing the value would lead to high training time with no important increase in the evaluating parameters and decreasing the Mini Batch size could lead to overfitting the model. In table 4.6 it is possibe to observe the attempts made with different Minibatch size.

Architecture	**ResNet18 85 Freezed Layers 30 Minibatch**			**ResNet18 85 Freezed Layers 38 Minibatch**		
Class	**Skin**	**Wound**	**Background**	**Skin**	**Wound**	**Background**
Accuracy	0.890	0.741	0.749	0.8965	0.725	0.76
IoU	0.764	0.602	0.653	0.77	0.583	0.67
MeanBFScore	0.562	0.434	0.572	0.567	0.396	0.578

Table 4.6: Table summarizes the results obtained for the different trained ResNet networks and the attempts with each network changing the corresponding parameters. These attempts were made to set a reason behind the selection of the Minibatch Number for further trainings with ResNet networks.

ResNet has been proven to be the most suitable network regarding timings. It was easy to achieve a reasonable accuracy value with less Epoch number than with SegNet, which was an important advantage. The same patterns observed in SegNet were repeated in this case. Figure 4.10 illustrates the evaluation with each epoch of the training with ResNet18. The curve starts to increase until stabilization, for a small Epoch number the accuracy achieved would have been small, however, after a certain number of Epochs, the Accuracy would be stable with small changes and would not vary significantly. For this reason the number of Epoch chosen was 200, in order to ensure being in the stable zone of the curve and to have a suitable execution time, since more epochs would mean more time required to run the Training process until the end. Related to the freezing layers, in this case I can observe that the more freezed layers, the accuracy vary decreasing, however, the time requirements also decrease. It would be interesting to find a suitable balance between a good accuracy and time of execution for a reasonable number of freezed layers and epochs.

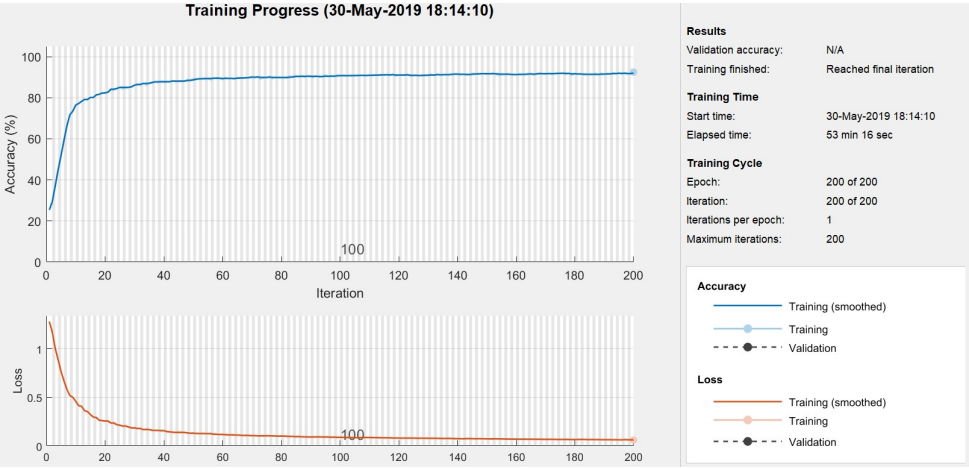

Figure 4.10: The image illustrates a plot of monitor display while training ResNet18 network freezing 30 layers and with a Minibatch size of 30. The curve represents the evolution of the Accuracy value with each Epoch.

4.3.3 FCN-AlexNet Approach

AlexNet is a deep learning architecture that consists on ReLUs as non-linearities (the aim is preventing overfitting), eight weighted layers, the first 5 are convolutional layers and the last three layers are fully connected layers. Finally, the output of the last fully connected layer is fed to a max-pooling one, 1000-way SoftMax. The ReLu applies to the output of the fully connected layers [Krizhevsky et al., 2012]. The figure 4.11 shows the network plotted in MatLab with the analyzer.

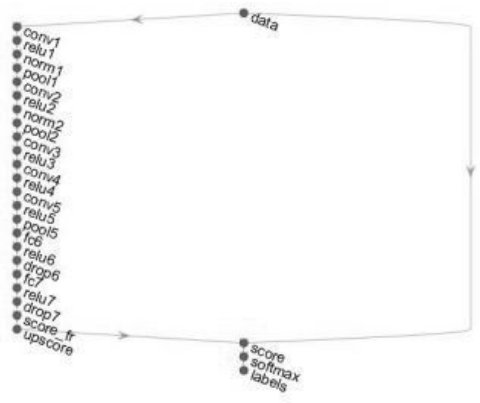

Figure 4.11: The image illustrates a plot of the pretrained model of AlexNet neural network offered by MatLab.

AlexNet								
Epoch	**Time**	**Mini Batch**	**Freezed Layers**	**Global Accuracy**	**Mean Accuracy**	**Mean IoU**	**Weighted IoU**	**Mean BFScore**
250	***0:48:50***	***40***	***15***	0.428	0.347	0.221	0.291	0.107
500 (2500It)	***3:08:21***	***8***	***10***	0.838	0.812	0.689	0.722	0.470
500 (1000It)	***2:32:08***	***20***	***10***	0.478	0.657	0.328	0.283	0.171
1000	***4:22:48***	***25***	***10***	0.481	0.657	0.329	0.287	0.170
600	***1:24:35***	***30***	***14***	0.437	0.383	0.237	0.297	0.114
1000	***2:33:43***	***30***	***12***	0.451	0.632	0.306	0.257	0.161
2000	***6:31:24***	***30***	***12***	0.823	0.814	0.670	0.702	0.417
2000	***5:11:57***	***30***	***15***	0.793	0.785	0.630	0.659	0.389

Table 4.7: Table summarizes the results obtained for different AlexNet network Training. The parameters illustrated are the one selected for evaluation, and also it is represented the parameters that were changed in each training with the corresponding values.

The procedure for evaluating this Pre-trained Model performance is the same as the previous ones. In table 4.7, it is shown the results for different training options and freezing layers. Same as with SegNet approach, in this case the first attempts training the network with a large Minibatch and small Epochs number did not lead to a good result. The next step was to decrease the Minibatch Size and increase the number of Epochs to see how the network would perform. Decreasing the Minibatch size, would help run the model with a high number of Epoch in a reasonable time. Once the results were set the idea was to try and perform a training with a Minibatch size reasonable

without leading to overfitting and a high number of Epochs once it was clear that the training improved with increased Epoch value.
With AlexNet again we face the same problem as in SegNet, the time was crucial in the performing of the trainings. Increasing the number of Epoch was the key to achieve a reasonable Accuracy value, however this would mean a huge increase in the Time of training as observed in table 4.7. The same happens with the freezed layers: increasing the number of freezed layers would improve the time of training, but this would also decrease the accuracy value and therefore the wound detection might not be as precise as required.

4.3.4 U-Net

U-Net it is a Fully convolutional deep learning network for semantic segmentation, specified for biomedical image segmentation. The architecture consist of encoder path and the decoder path. The encoder path contains convolutional and max-pooling layers, meanwhile the decoder path uses transported convolutions for a correct detection [Lamba, 2019]. MatLab offers the function *unetLayers* to create a U-Net network for semantic segmentation and for further training [The MathWorks Inc., 2019i]. The procedure to Train the network was the same as describes in the previous subsections related ResNet or SegNet. Nevertheless, in this case the encoder/decoder Depth was a changing parameter. The first attempt was with and encoder/decoder Depth of 3, further attempts were with 4 and 5 encoder/decoder Depth. Regarding the freezed Layers, it was selected according to the number of layers of each attempt.

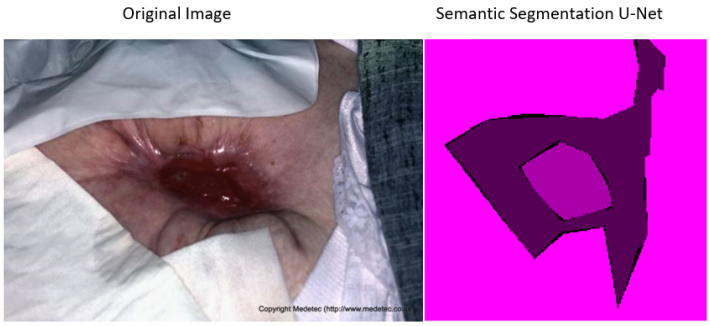

Figure 4.12: The image illustrates a plot of the results of evaluating a trained U-Net network.

The result of each training was not satisfactory. The higher the value, longer would take for the training to end, but with no improvement in the Accuracy. In Figure 4.13 it is

illustrated the display for training progress in the three cases. The number of Epochs was set differently in each case due to time limitations. For the first attempts, the network was not as complex as in the case of an encoder depth of 5, for this reason, it was possible to try a higher number of epochs to see the performance. Also, since the curve gets to stability soon, it was not necessary to set a higher number of Epochs as a first overview in the matter.

Encoder Depth	Freez Layers	Minibatch Size	Time	Global Accuracy	Mean Accuracy	Mean IoU	Weighted IoU	MeanBFScore
3	**20**	**20**	2:00:46	0.59028	0.33333	0.19676	0.34843	0.73744
4	**30**	**20**	2:11:15	0.59028	0.33333	0.19676	0.34843	0.73744
5	**40**	**20**	1:22:10	0.59028	0.33333	0.19676	0.34843	0.73744

Table 4.8: The table summarizes the results for U-Net approach regarding the Accuracy, IoU, MeanBFScore and Time.

In the table 4.8, it is possible to observe the time results, IoU, MeanBFScore and Accuracy in each case. Regarding the results per class, there was no coherence, since it detected all as the same class, Skin, as observed in figure 4.13.

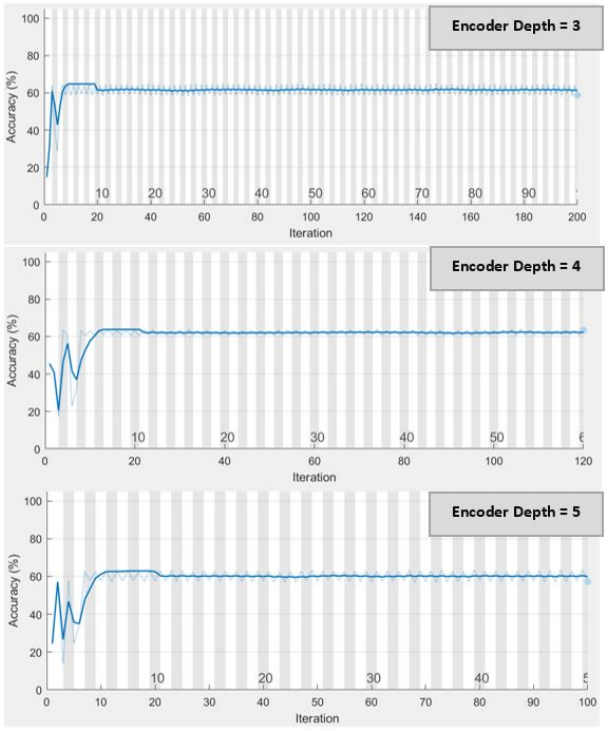

Figure 4.13: The image illustrates a plot of the monitoring progress in MatLab related to the three attempts made with U-Net.

4.4 Test Results

In this section the trained network will be tested with the test data separated in the data preparation section. To test the model, they were chosen the best trained options of each network due to time limitations. Each trained network was tested on new data (test data) to evaluate the wound detection performed by each model over the data in use. The MatLab code used in this section was similar to the Validation of the Network, used in the previous section. In this case, it was easier, since we did not had to label the data previously, due to the need of performing semantic segmentation over a new subset of completely new images, it was not necessary to label them previously. The idea was to compare the original image with the performance of the trained network for semantic segmentation, if it detects correctly the observed wound in the original image.

```
% Load the net of interest
load net;
% Load the images
idx=22;
Im= 'C:\Users\chaim\Desktop\TFM\Bildmaterial_MatLab';
imgDir1=fullfile(Im, 'OrgDataTesting', '');
imdsTest = imageDatastore({imgDir1});
imageFolder = fullfile(outputFolderResized,'imagesResizedtest',filesep);
imdsTest = resizeWoundImages(imdsTest,imageFolder);
% read the image of interest
I = readimage(imdsTest,idx);
% smenaticseg to un the network of interest over the new image of interest
C = semanticseg(I, net);
%Display results
B = labeloverlay(I,C,'Colormap',cmap,'Transparency',0.4);
subplot(1,2,1), imshow(I);
subplot(1,2,2), imshow(B);
pixelLabelColorbar(cmap, classes);
%% It is also possible to run the semanticseg over all the data set of interest
pxdsResults = semanticseg(imdsTest,net, ...
    'MiniBatchSize',5, ...
    'WriteLocation',tempdir, ...
    'Verbose',false);
% And evaluate the performance with the correpsonding metrics
metrics = evaluateSemanticSegmentation(pxdsResults,pxdsTest,'Verbose',false);
% See metric in the overall network performance
metrics.DataSetMetrics
% To see the impact in each class separatly
metrics.ClassMetrics
```

SegNet Test: The first approach was with SegNet Networks, starting with SegNet based on VGG16. Two training attempts were selected. The first was the option trained with 200 Epochs, 35 freezed layers and a Minibatch size of 10. The second network was trained with 500 Epochs, 35 freezed layers and a Minibatch size of 20.

Regarding SegNet based on VGG19, two of the training attemps were selected with the same conditions as VGG16. The first was the option trained with 200 Epochs, 35 freezed layers and a Minibatch size of 10. The second network was trained with 500 Epochs, 35 freezed layers and aMinibatch size of 20.

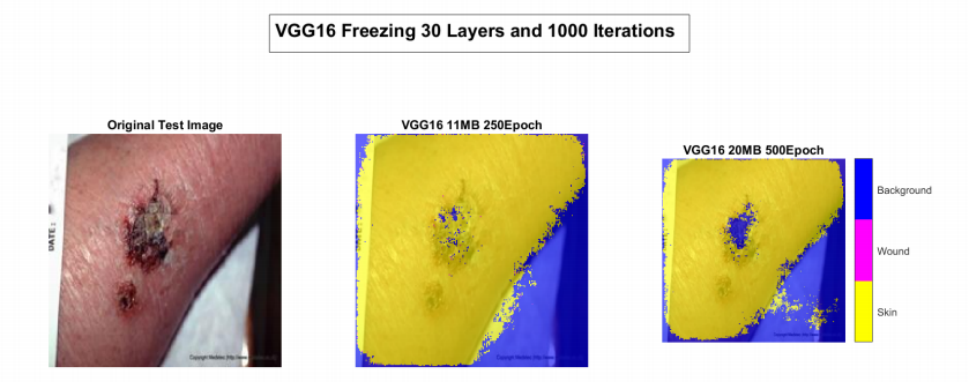

Figure 4.14: The image illustrates the result of using the trained network VGG16 on new data, which is the test data from the splitting in section data preparation. In the right we have the plotted the original image, meanwhile in the center and the right the results of the test detecting the wound for a Minibatch size of 11 and 20 respectively are plotted.

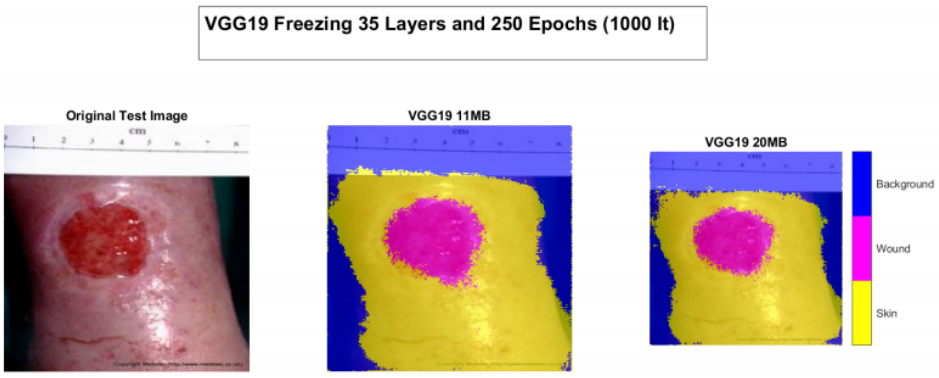

Figure 4.15: The image illustrates the result of using the trained network VGG19 on new data, which is the test data from the splitting in section data preparation. In the right we have the plotted the original Image, meanwhile in the center and the right the results of the test detecting the wound for a Minibatch size of 11 and 20 respectively are plotted.

After training the networks the trained net was saved as a '.mat' file and used for the testing purpose with the test subset. The images plotted contrasted the original image

and the semantic segmentation developed by the trained network over this new data. The results for segnet based on VGG16 and VGG19 are shown in figure 4.14 and 4.15 respectively. As shown in the image, the performance is similar, however, in both cases it is not perfect the precision, since some pixels related to wound class are misplaced in the image, and the result is not fully consistent.

ResNet Test: For testing the performance of trained ResNet network, the procedure was the same as the previous with SegNet. The trained network was saved as a '.mat' file and used to observe the performance on the new data (Test data). The results of the three approaches were observed. Nevertheless, analyzing the numerical data in the tables and the visual results after testing the networks, it was observed that the results were similar, sometimes exactly the same values were seen in ResNet18 and ResNet50 for the same number of freezed layers. After a closer look on the matlab code based on [The MathWorks Inc., 2019f], it was clear that due to the function *helperDeeplabv3PlusResnet18*. The network being trained was constantly ResNet18 with different training option, but never ResNet50 or ResNet101 itself. This is due to the fact that the function mentioned, by default it loads ResNet18 to perform the changes for a semantic segmentation network. An attempt to arrange the function and make it suitable for ResNet50 and ResNet101 was made, however, it was not possible to reach a result, since a lot of architecture re-designing has to be done to achieve the goal.

Architecture	**ResNet18 30 Freezed Layers 200 Epochs**			**ResNet18 30 Freezed Layers 300 Epochs**			**ResNet18 30 Freezed Layers 400 Epochs**		
Class	**Skin**	**Wound**	**Background**	**Skin**	**Wound**	**Background**	Skin	Wound	Background
Accuracy	0.905	**0.762**	0.593	0.908	0.663	0.743	0.918	0.646	0.734
IoU	0.679	**0.590**	0.406	0.765	0.588	0.646	0.766	0.584	0.7644
MeanBFScore	0.734	**0.639**	0.579	0.599	0.410	0.584	0.614	0.406	0.587
Time	53'			82'			106'		

Table 4.9: The table summarizes the evaluation parameter values for three attempts with ResNet18 in the same conditions changing the number of Epochs

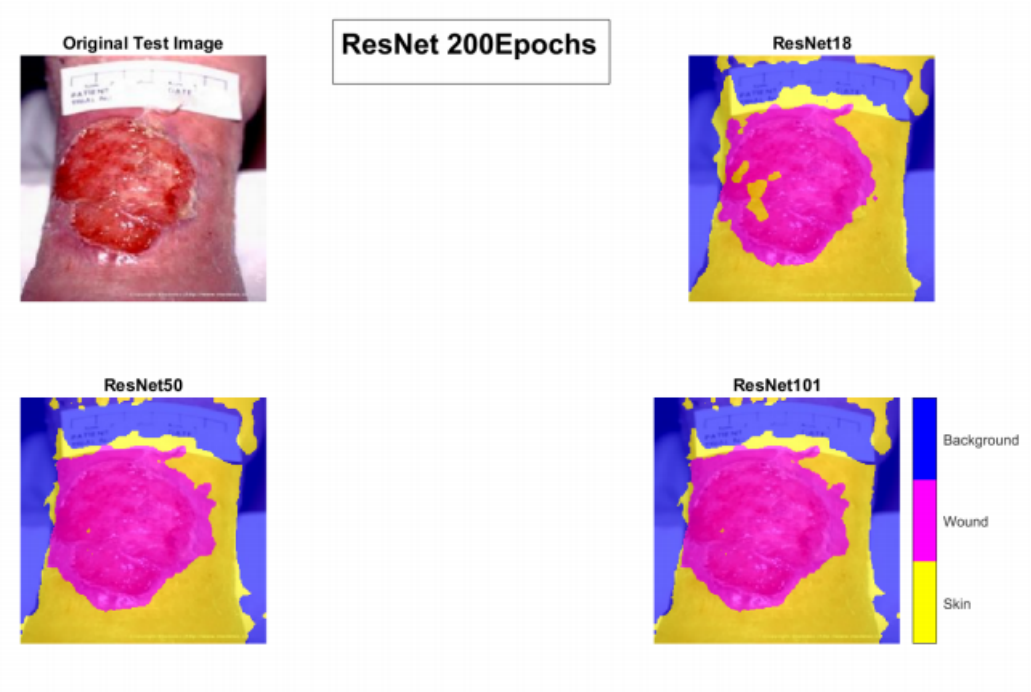

Figure 4.16: The image illustrates the result of using the trained network ResNet on new data, which is the test data from the splitting in section data preparation. In the right above we have the plotted the original Image, meanwhile in the left above, right below and left below are plotted the result of the test detecting the wound for ResNet18, ResNet50 and ResNet101 respectively.

Finally, with this network, it was performed a trainig with only ResNet18 changing the number of Epochs and Freezed Layers to observe whether the number of Epochs was crucial when observing the metrics for each class, and specially in the network testing, the visual representation of wound detection using ResNet. In figure 4.17 and table 4.9 the results are represented, and once again, it confirmed the previous results, increasing the number of epoch did affect the accuracy, IoU and MeanBFScore, but it was not crucial, however, the time of training arises significantly.

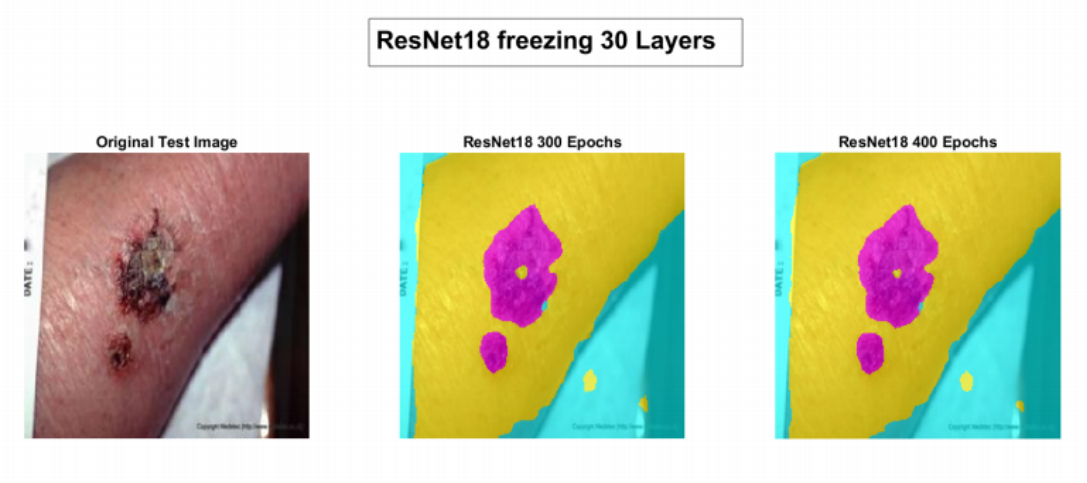

Figure 4.17: The image illustrates the result of using the trained network ResNet, with 30 freezed layers, on new data, which is the test data from the splitting in section data preparation.

A final comparison between the 4 different freezing optiones tried with ResNet to see the difference is plotted. Figure 4.18 it illustrates of the results.

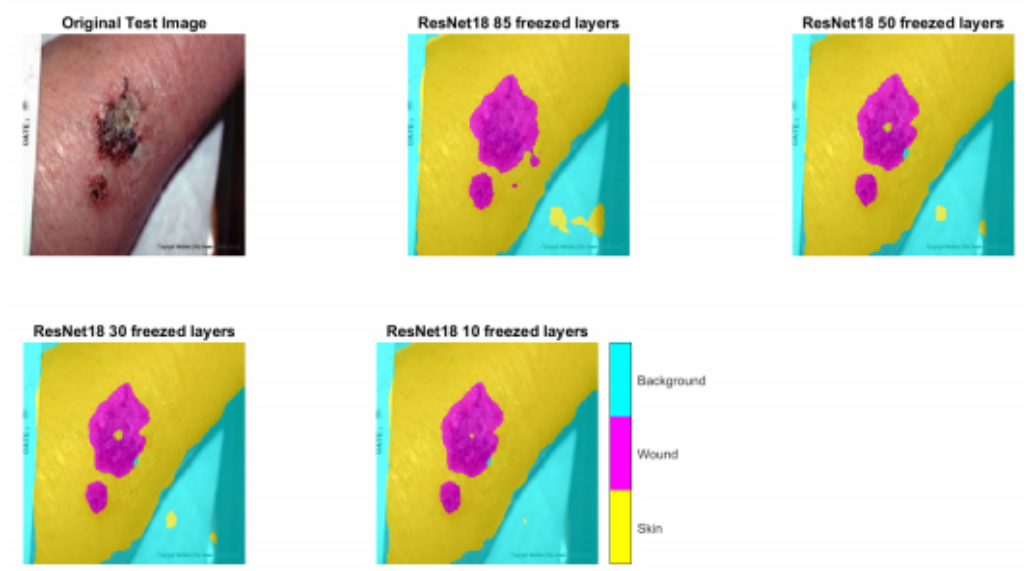

Figure 4.18: The image illustrates the result of using the trained network ResNet, with 300 Epochs, on new data, which is the test data from the splitting in section data preparation.

AlexNet Test: In the case of AlexNet testing, the results were similar to SegNet. It required a high number of Epochs, and therefore, a large time to achieve a suitable

accuracy value. The tested network was trained with a 1000 Epochs on a Minibatch size of 30 and the result of testing on new data is the illustrated on figure 4.19. The precision of the detection is not quite high, however it is better than the SegNet approach for the given conditions.

Figure 4.19: The image illustrates the result of using the trained network VGG16 on new data, which is the test data from the splitting in section Data Preparation. In the left we have the original Image, meanwhile in the right the result of the test detecting the wound is plotted.

U-Net Test: U-Net has shown incoherent results regarding wound detection, since it did not recognize wound and background classes. This is easly observable in figure 4.20, where the results of testing and U-Net netwrok with encoder depth of 3 and 4 are plotted.

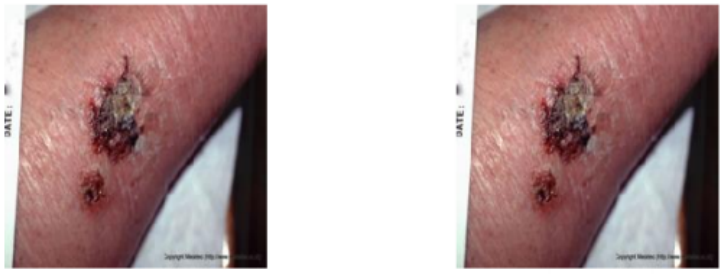

Figure 4.20: The image illustrates the result of using the trained network U-Net on new data, which is the test data from the splitting in section data preparation. In the right we have the plotted the original Image, meanwhile in the left and in the right below are plotted the results of the test detecting the wound for Unet with encoder depth of 3 and 4 respectively.]

4.5 Thesis Limitations

The research developed in this Master Thesis faced several technical limitations or restrictions, who influenced the outcome.
In general, the first aspect to take into account was the fact that the architectures used, originally were trained on ImageNet dataset, where most of the dataset are of animals, cars, etc. This was important to evaluate the performance of the networks after freezing some layers. The freezed Layers are the initial layers, more related to general feature extraction. The main limitation was due to a lack of hardware resources. In this Thesis I have worked with only one CPU and during network training on images the system could not achieve to work fast or at the same efficiency with all the networks. Therefore, it is believed, that CPU resources could have been a key factor for improving the performance of the network training, so as setting a GPU on the computer system in use, since GPUs are more efficient making the training of the neural network faster [Giuliano Giacaglia, 2019]. Due to the parallel architecture that allows to solve several tasks at the same time [NVIDIA Inc., 2019]. The availability of a suitable computational resource would have been an advantage for further trainings with more images and shorter time requirements. The data available was also a limitation in this thesis, since normally, indeep learning an amount of 1000 images is recommended. Finally, the image labelling was another restriction, since it was done manually with no clinical advice. It is important for future work to perform a labelling advised by profession for a medically valuable result. The images were collected from different sources, which meant a different resolution and quality for each image, this requires a standardization step during the development of this Thesis [Schnalzer, 2018].

Chapter 5

Discussion

In this chapter the obtained results will be discussed and some conclusions will be presented.

5.1 Summary

The present thesis evaluated the performance of 4 pre-trained models and its corresonding possibilities, in semantic segmentation for skin wound detection. The models trained were ResNet, AlexNet, SegNet (based on VGG16 and VGG19), and U-Net, which were found to be the commonly used for semantic segmentation. It was not possible to perform a full training due to hardware and available data limitations. Therefore, fine tuning the model was the selected option. These deep learning architectures were trained and segmentation performance was evaluated based on the commmonly used metrics for evaluation. The present source code proved that ResNet might be the reasonable option regarding the available time and the accuracy achieved. However, further training with the remaining networks counting on better hardware availability might prove a different statment affecting the results. Nevertheless, it is interesting to take the performance of **ResNet**, taking into acount the required time for a good result in comparison with the different networks in use during the development of this thesis.

5.2 Thesis Discussion

After Training the networks, evaluating the Network performance with the validation data and testing the models with test Data (using the parameteres indicated in table 5.1), it was observed that **ResNet** would be the **best approach for semantic segmentation** in Matlab regarding skin wound detection, in the present conditions of this thesis. This is reasonable due to three factors: First of all, time restriction has been a main factor in

Network	Network Layers	Freezed Layers	Epochs	Minibatch Size
ResNet 18	**101**	**10**	**200**	**30**
AlexNet	27	15	2000	30
VGG 19	109	35	500 (1000It)	20
VGG 16	91	20	500 (1000It)	20
UNET-3	46	20	100 (200It)	20
UNET-4	58	30	60 (120It)	20
UNET-5	70	40	60 (120It)	20

Table 5.1: The table summarizes the used parameters to train each network. The table includes only the best approaches made

Network	Time	Accuracy	Global Accuracy	Mean Accuracy	MeanIoU	Weighted IoU	Mean BFScore	Computational Storage
ResNet 18	**0:53:16**	**92.47%**	**0.8381**	**0.781**	**0.677**	**0.720**	**0.536**	**260 KB**
AlexNet	5:11:57	80.37%	0.793	0.785	0.630	0.659	0.390	682 KB
VGG19	11:58:06	87.40%	0.804	0.706	0.603	0.668	0.401	930 KB
VGG 16	9:56:47	74.85%	0.741	0.562	0.428	0.572	0.299	590 KB
UNET-3	2:00:46	61.15%	0.590	0.333	0.197	0.348	0.737	715 KB
UNET-4	2:11:15	63.34%	0.590	0.333	0.197	0.348	0.737	543 KB
UNET-5	1:43:27	57.56%	0.590	0.333	0.197	0.348	0.737	227 KB

Table 5.2: The table summarizes the results of the indicators used for network performance evaluation. The results correspondents to the conditions exposed in table 5.1.

this case. With ResNet network it is possible to achieve a good accuracy in a reasonable time with a small number of Epochs, working with only one CPU. This was the main advantage of this network in use. Secondly, the Network, as observed in table 5.2, does not require a high computational storage, which is an advantage for futuro use in mobil devices. Additionally, another important advantage was related to accuracy values, even with less Epochs, none of the other trained networks has achieved the accuracy values of ResNet.

AlexNet was another possible network if timing execution was no longer a limitation. In this case, the Network always started to improve the Accuracy values gradually with each Epoch. Also, the computational storage is not high, and even ending the training in the Epoch 2500 (was the maximum number of Epoch tried with a Mini-Batch size of 20),

there were possibilities of improvement running the model for more Epochs. It would be a possible future work to train the network for more epochs and higher Minibatch size and look for the scope of improvement of this Epoch in order to evaluate if it is worthy regarding the time-accuracy equation.

SegNet was another approach in this research that acted same as AlexNet, however, the computational storage was higher and so were the time requirements. As mentioned in the literature [He et al., 2016], VGG is one of the complex architectures in deep learning. Therefore, it was expected to be in front of a network that requires a large number of Epochs in order to achieve a good accuracy value, which was possible, for instance, in the case of SegNet based on **VGG19** with a Minibatch size of 20 and 500 Epochs (1000 Iterations). Nevertheless, the timing was almost 12 hours. The advantage is, that even when the training ended, the curve has not achieved stability. There was a possibility for improvement in accuracy values with more Epochs, which could be an interesting future work to see the scope of this architecture.

U-Net in the other hand, was the last approach. Since it is a Network created based on Biomedical Images, the expectative were high. However, the results were not the expected. The network gave a result where only one class was detected. The specific results for each class confirmed the fact that for Wound and Bakcground the MeanBFScores gave a NaN result.

Focusing on each network, it was observed that, on the one hand, increasing the freezed layers showed a decrease in the accuracy, IoI and MeanBFScore values, which means a decrease on the quality of wound detection by the model. This is logical since these freezed layers were not being trained on the new data, and kept the weights loaded with the pretrained network, which was trained on different images. Nevertheless, this helped decreasing the trainig times significantly in some cases. On the other hand, the Minibatch size has proven to be a key factor in the performance of the networks. The higher the Minibatch size, the longer is the training required to achieve a suitable parameter results. Therefore, again we face the quality-Timing issue. Also, the number of Epoch, at some point, when the performance is stable and there is no improvement in the Accuracy, therefore, the network has achieved its scope, only affecting negatively in the time values. For instance, in the case of ResNet network, the stability is achieved with a small Epochs number, at a point, running the training for a higher number of Epoch would just mean longer training with the same Accuracy. In other cases, such as for SegNet or AlexNet, we did not get to find the scope of the network, in this caso it would have been interesting to perform training with higher number of Epoch until reaching the scope of the network.

Comparing the numerical results and the visual results in some cases, for example regarding U-Net or VGG16, the accuracy had not a bad result, however if when the metrics specifics for each class were observed, the accuracy was not similar for each class. In table 5.3, the results regarding Skin, Wound and Background for the best performing option regarding each network are summarized.

Architecture	**Class**	**Accuracy**	**IoU**	**MeanBFScore**
VGG 16	Skin	0.882	0.696	0.383
	Wound	8.3 E-06	8.3 E-06	0
	Background	0.805	0.587	0.317
VGG 19	Skin	0.915	0.735	0.500
	Wound	0.470	0.435	0.500
	Background	0.733	0.638	0.471
RESNET 18	Skin	**0.915**	**0.775**	**0.609**
	Wound	**0.677**	**0.509**	**0.404**
	Background	**0.751**	**0.666**	**0.600**
ALEXNET	Skin	0.823	0.704	0.419
	Wound	0.817	0.589	0.361
	Background	0.716	0.598	0.384
U-NET	Skin	1	0590	0.737
	Wound	0	0	NaN
	Background	0	0	NaN

Table 5.3: The table summarizes the impact on each class. The metrics value for each class are represented for the different trained Networks.

For instance, it was observed in case of U-Net, since is only recognize one class, Skin. This class is clear to be Skin, since the entire image was taken as Skin, then every actual pixel of the Skin was correctly recognized, nevertheless, in case of Class wound, zero pixels were recognized as wound. In addition, in VGG16, it is possible to conclude from the results that the wound class detection was not the desired, since the accuracy, Iou and MeanBFScore has a very low value. It is possible to perform the trainig with further attemps for a higher number of Epochs and a larger dataset and if possible a full training to get the scope of this architecture regarding semantic segmentation in skin wound medical images. In the case of this research, the time and hardware available were a clear limitation to perform more complex trainings.

5.3 Conlusions

In conslusion, counting on the limitations faced in the development of this thesis, **ResNet was found to be the best network** taking into account the attempts made with the other approaches. The network reached a reasonable and good accuracy with a small period of time, meanwhile other networks in similar conditions did not achieve good results as ResNet. As a **second recommendable approach** it is to be mentioned **AlexNet**, since it presented a good evaluating results, visually it is possible to detect that along with ResNet it is the best network to detect correctly the skin wound on the test images, however, the timing execution is significantly higher than ResNet. As a **third option**, **VGG19** showed good results, nevertheless, the timing is a limitation in the conditions of this project that affected the results. Finally, **VGG16**, proved good skin and background accuracy, IoU and MeanBFScore results, meanwhile a poor accuracy, IoU and MeanBFScore values for wound detection. Refering **U-Net** the results were not the expected. It would be interesting further research in the field to understand the reasons behind the presented results. Figure 5.1 shows a personal recommendable ranking of the perfomance of the trained networks. The classification is based on the results of this master thesis taking into account the differente faced limitations.

Figure 5.1: The image illustrates a personal recommendable ranking for the trained networks performance being ResNet the best and U-Net the last recommendable one. The classification is according to the obtained results in this research, taking into account the restrictions faced. (Source: [Nussbaum et al., 2018])

Although ResNet in the case of this research has proven to be the best approach, it is interesting to take into account that with the attempts made the scope of this network was reached. However, with AlexNet or VGG, this scope was not reached. For this reason it is coherent to think of possible improvements and changes in the conclusion if more hardware resources were available to perform more complex trainings according to the requirements of each networks.

Compared to literature, it was possible to find some articles where **ResNet** was metioned as the best, or one of the best, performer network, [Saikia et al., 2019] and [Hershey et al., 2017]. Also a comparation regarding accuracy was developed by [Koustubh, 2018]. Despite the fact that the three sources were using deep learning architecture for different tasks, among the architectures uses, ResNet reached the best or one of the best Accuracy values.

Bibliography

Albelwi, S. and Mahmood, A. (2017). A framework for designing the architectures of deep convolutional neural networks. *Entropy*, 19(6):242.

Anderson, A., Shaffer, K., Yankov, A., Corley, C. D., and Hodas, N. O. (2016). Beyond fine tuning: A modular approach to learning on small data. *arXiv preprint arXiv:1611.01714*.

Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495.

Basheer, I. A. and Hajmeer, M. (2000). Artificial neural networks: fundamentals, computing, design, and application. *Journal of microbiological methods*, 43(1):3–31.

Bell, J. (2014). *Machine learning: hands-on for developers and technical professionals*. John Wiley & Sons.

Bentaouza, C. M. and Benyettou, M. (2018). Support vector machine applied to compress medical image. *JCP*, 13(5):580–587.

Dey, D., Slomka, P. J., Leeson, P., Comaniciu, D., Shrestha, S., Sengupta, P. P., and Marwick, T. H. (2019). Artificial intelligence in cardiovascular imaging: Jacc state-of-the-art review. *Journal of the American College of Cardiology*, 73(11):1317–1335.

Erickson, B. J., Korfiatis, P., Akkus, Z., Kline, T., and Philbrick, K. (2017). Toolkits and libraries for deep learning. *Journal of digital imaging*, 30(4):400–405.

Foster, K. R., Koprowski, R., and Skufca, J. D. (2014). Machine learning, medical diagnosis, and biomedical engineering research-commentary. *Biomedical engineering online*, 13(1):94.

Gando, G., Yamada, T., Sato, H., Oyama, S., and Kurihara, M. (2016). Fine tuning deep convolutional neural networks for distinguishing illustrations from photographs. *Expert Systems with Applications*, 66:295–301.

Garcia-Garcia, A., Orts-Escolano, S., Oprea, S., Villena-Martinez, V., and Garcia-Rodriguez, J. (2017). A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*.

Hamet, P. and Tremblay, J. (2017). Artificial intelligence in medicine. *Metabolism*, 69:S36–S40.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. pages 770–778.

Hershey, S., Chaudhuri, S., Ellis, D. P., Gemmeke, J. F., Jansen, A., Moore, R. C., Plakal, M., Platt, D., Saurous, R. A., Seybold, B., et al. (2017). Cnn architectures for large-scale audio classification. pages 131–135.

Jarbrink, K., Ni, G., Sonnergren, H., Schmidtchen, A., Pang, C., Bajpai, R., and Car, J. (2016). Prevalence and incidence of chronic wounds and related complications: a protocol for a systematic review. *Systematic reviews*, page 152.

J.Veredas, F., Luque-Baena, R. M., Martin-Santos, F. J., Morilla-Herrera, J. C., and Morente, L. (2015). Wound image evaluation with machine learning. *ELSEIVER*.

Kaur, D. and Kaur, Y. (2014). Various image segmentation techniques: a review. *International Journal of Computer Science and Mobile Computing*, 3(5):809–814.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. pages 1097–1105.

Kruse, R., Borgelt, C., Braune, C., Mostaghim, S., and Steinbrecher, M. (2016). *Computational intelligence: a methodological introduction*. Springer.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lee, J.-G., Jun, S., Cho, Y.-W., Lee, H., Kim, G. B., Seo, J. B., and Kim, N. (2017). Deep learning in medical imaging: general overview. *Korean journal of radiology*, 18(4):570–584.

Lemley, J., Bazrafkan, S., and Corcoran, P. (2017). Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, 5:5858–5869.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., and Alsaadi, F. E. (2017). A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26.

Liu, X., Deng, Z., and Yang, Y. (2018). Recent progress in semantic image segmentation. *Artificial Intelligence Review*.

Lundervold, A. S. and Lundervold, A. (2019). An overview of deep learning in medical imaging focusing on mri. *Zeitschrift für Medizinische Physik*, 29(2):102–127.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill Science/Engineering/Math.

Neill, D. B. (2013). Using artificial intelligence to improve hospital inpatient care. *IEEE Intelligent Systems*, 28(2):92–95.

Oršić, M., Krešo, I., Bevandić, P., and Šegvić, S. (2019). In defense of pre-trained imagenet architectures for real-time semantic segmentation of road-driving images. *arXiv preprint arXiv:1903.08469*.

R. Rojas (1996). The backpropagation algorithm. *Springer-Verlag*.

Rahman, M. A. and Wang, Y. (2016). Optimizing intersection over union in deep neural networks for image segmentation. pages 234–244.

Rastgarpour, M. and Shanbehzadeh, J. (2011). Application of ai techniques in medical image segmentation and novel categorization of available methods and.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. pages 234–241.

Saffar, M. H., Fayyaz, M., Sabokrou, M., and Fathy, M. (2018). Semantic video segmentation: A review on recent approaches. *arXiv preprint arXiv:1806.06172*.

Saikia, A. R., Bora, K., Mahanta, L. B., and Das, A. K. (2019). Comparative assessment of cnn architectures for classification of breast fnac images. *Tissue and Cell*, 57:8–14.

Schnalzer, B. K. (2018). Wound detection with neural networks on ios mobile devices. *Institute for eHealth of the Graz University of Applied Science*.

Sen, C. K., Gordillo, G. M., Roy, S., Kisner, R., Lambert, L., Hunt, T. K., Gottrup, F., Gurtner, G. C., and Longaker, M. T. (2009). Human skin wounds: A major and snowballing threat to public health and the economy. *The wound healing society*.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Smith and Nephew (2014). El impacto humano y economico de las heridas. *Smith and Nephew*.

Sumathi, S. and Paneerselvam, S. (2010). *Computational intelligence paradigms: theory & applications using MATLAB*. CRC Press.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. pages 1–9.

Tajbakhsh, N., Shin, J. Y., Gurudu, S. R., Hurst, R. T., Kendall, C. B., Gotway, M. B., and Liang, J. (2016). Convolutional neural networks for medical image analysis: Full training or fine tuning? *IEEE transactions on medical imaging*, 35(5):1299–1312.

Thoma, M. (2016). A survey of semantic segmentation. *arXiv preprint arXiv:1602.06541*.

Tran, B. X., Vu, G. T., Ha, G. H., Vuong, Q.-H., Ho, M.-T., Vuong, T.-T., La, V.-P., Ho, M.-T., Nghiem, K.-C. P., Nguyen, H. L. T., et al. (2019). Global evolution of research in artificial intelligence in health and medicine: A bibliometric study. *Journal of clinical medicine*, 8(3):360.

V. Keckman (2015). Support vector machines – an introduction. *Springer*.

Vaidya, B. and Paunwala, C. (2019). Deep learning architectures for object detection and classification. pages 53–79.

Vanitha L. AND Venmathi A.R. (2011). *Classification of Medical Images Using Support Vector Machine*. 2011 International Conference on Information and Network Technology.

Wang, C., Yan, X., Smith, M., Kochlar, K., Rubin, M., Warren, S. M., and Lee, H. (2015). Unifed framework for automatic wound segmentation and analysis with deep convolutiona neural networks. *IEEE*.

Wang, M., Cui, Y., Wang, X., Xiao, S., and Jiang, J. (2017). Machine learning for networking: Workflow, advances and opportunities. *IEEE Network*, 32(2):92–99.

Xia, X. and Kulis, B. (2017). W-net: A deep model for fully unsupervised image segmentation. *arXiv preprint arXiv:1711.08506*.

Yu, K.-H., Beam, A. L., and Kohane, I. S. (2018). Artificial intelligence in healthcare. *Nature biomedical engineering*, 2(10):719.

Zacharias, J., Barz, M., and Sonntag, D. (2018). A survey on deep learning toolkits and libraries for intelligent user interfaces. *arXiv preprint arXiv:1803.04818*.

Zhanh, T. C., Yang, J., Zhang, J. P., and Zhang, J. (2016). *SVM Methods in Image Segmentation*. The Sixth International Conference on Advanced Collaborative Networks, Systems and Applications.

Internet sources

Bagnato, J. I. (2018). ¿cómo funcionan las convolutional neural networks?. visión por ordenador. https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/. Accessed: 30-06-2019.

Cadogan, M. (2018). Clinical image database. https://lifeinthefastlane.com/table/clinical-image-database/. Accessed: 22-05-2019.

Castrounis, A. (2016). Artificial intelligence, deep learning, and neural networks, explained.

Center, W. C. (2019). Different types of wounds. https://www.woundcarecenters.org/article/wound-basics/different-types-of-wounds. Accessed: 22-03-2019.

Code 3 Emergency Partners (2016). Lacerations. https://code3er.com/lacerations/#. Accessed: 22-05-2019.

deepnearning.net (2019). Datasets. http://deeplearning.net/datasets/. Accessed: 22-05-2019.

Dua, D. and Graff, C. (2017). UCI machine learning repository. http://archive.ics.uci.edu/ml". Accessed: 22-05-2019.

Galderma S.A. (2018). Dermquest.com. https://www.dermquest.com/image-library/image/5044bfd1c97267166cd6614d?_id=5044bfd1c97267166cd6614d. Accessed: 22-05-2019.

Ghandi, R. (2018). Support vector machine - an introduction to machine learning algorithms. https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. Accessed: 15-05-2019.

GitHub Inc. (2019a). Awesome semantic segmentation. https://github.com/mrgloom/awesome-semantic-segmentation. Accessed: 28-04-2019.

GitHub Inc. (2019b). github. https://github.com/. Accessed: 06-06-2019.

Giuliano Giacaglia (2019). Deep learning processors. https://medium.com/@giacaglia/deep-learning-processors-4dbda91a9845. Accessed: 10-06-2019.

Government, A. (2018). Skin tears. https://www.dva.gov.au/providers/provider-programs/wound-care/skin-tears. Accessed: 22-05-2019.

Kaggle Inc. (2019). Kaggle. https://www.kaggle.com/. Accessed: 22-05-2019.

Koustubh (2018). Resnet, alexnet, vggnet, inception: Understanding various architectures of convolutional networks. https://cv-tricks.com/cnn/understand-resnet-alexnet-vgg-inception/. Accessed: 13-06-2019.

Krasin, I., Duerig, T., Alldrin, N., Ferrari, V., Abu-El-Haija, S., Kuznetsova, A., Rom, H., Uijlings, J., Popov, S., Veit, A., et al. (2018). Openimages: A public dataset for large-scale multi-label and multi-class image classification, 2017. dataset available at https://storage.googleapis.com/openimages/web/download.html.

Krizhevsky, A (2019). Cifar-10 and cifar-100 dataset. http://archive.ics.uci.edu/ml. Accessed: 22-05-2019.

Lab, S. V. (2018). Imagenet. http://image-net.org. Accessed: 22-05-2019.

Lamba, H. (2019). Understanding semantic segmentation with unet. https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47. Accessed: 15-05-2019.

Le, J. (2019). How to do semantic segmentation using deep learning. https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef. Accessed: 28-04-2019.

LeCun, Y., Cortes, C., and Burgues, C. J. (2019). The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/. Accessed: 22-05-2019.

LLC, H. L. C. (2018). Clinical hypnosis to enhance the scar & wound healing process. https://www.iwanttoquitsmoking.com/clinical-hypnosis-to-enhance-the-scar-wound-healing-process/. Accessed: 22-05-2019.

LTD, E. T. M. P. (2018). Injury descriptionin the ed. https://etmcourse.com/injury-description-in-the-ed/. Accessed: 22-05-2019.

Marcelino, P. (2018). Transfer learning from pre-trained models. https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751. Accessed: 25-05-2019.

Medetec (2018). Medetec wound database. http://medetec.co.uk/files/medetec-image-databases.html. Accessed: 22-05-2019.

Medicine, J. (2013). Abscess incision and drainage, a photographic tutorial. //www.jailmedicine.com/tag/mrsa/. Accessed: 22-05-2019.

MedlinePlus (2019). Heridas y lesiones. https://medlineplus.gov/spanish/woundsandinjuries.html. Accessed: 22-03-2019.

Nigeria Galleria (2018). Open wound. https://www.nigeriagalleria.com/Community-Health/Open-Wound.html. Accessed: 22-05-2019.

Nussbaum, S. R., Carter, M. J., Fife, C. E., DaVanzo, J., Haught, R., Nusgart, M., and Cartwright, D. (2018). An economic evaluation of the impact, cost, and medicare policy implications of chronic nonhealing wounds. *Value in Health*, 21(1):27–32.

NVIDIA Inc. (2019). ¿qué es la computación acelerada por gpu? https://la.nvidia.com/object/what-is-gpu-computing-la.html. Accessed: 10-06-2019.

Queensland University of Technology (2018a). 7a: Wound healing. http://promoting-healthy-skin.qut.edu.au/m7_healing.html. Accessed: 22-05-2019.

Queensland University of Technology (2018b). Fine tuning en reconocimiento de imagenes mediante deep learning. https://medium.com/neuron4/fine-tuning-en-reconocimiento-de-im%C3%A1genes-mediante-deep-learning-c656ae728d73. Accessed: 25-05-2019.

Raven (2017). Everything you need to know about neural networks. https://medium.com/ravenprotocol/everything-you-need-to-know-about-neural-networks-6fcc7a15cb4. Accessed: 28-04-2019.

Roddick, J. and Higuera, V. (2018). Open wound. https://www.healthline.com/health/open-wound. Accessed: 22-03-2019.

Royo, C. V. (2016). La segmentacion semantica y sus benchmarks. http://informatica.blogs.uoc.edu/2016/05/26/la-segmentacion-semantica-y-sus-benchmarks/. Accessed: 15-04-2019.

Sandeep, A. (2019). Object detection iou intersection over union. https://medium.com/@nagsan16/object-detection-iou-intersection-over-union-73070cb11f6e. Accessed: 30-05-2019.

Shao, C. (2019). Approach pre-trained deep learning models with caution. https://medium.com/comet-ml/approach-pre-trained-deep-learning-models-with-caution-9f0ff739010c. Accessed: 28-04-2019.

Sharma, P. (2019). 5 amazing deep learning frameworks every data scientist must know! (with illustrated infographic). https://www.analyticsvidhya.com/blog/2019/03/deep-learning-frameworks-comparison/. Accessed: 29-04-2019.

Shubh Saxena (2017). Artificial neuron networks (basics) introduction to neural networks. https://becominghuman.ai/artificial-neuron-networks-basics-introduction-to-neural-networks-3082f1dcca8c. Accessed: 30-06-2019.

Skimind (2019). Open data for deep learning & machine learning. https://deeplearning4j.org/opendat. Accessed: 22-05-2019.

Stanford (2019). Convolutional neural networks for visual recognition. hhttp://cs231n.github.io/convolutional-networks/. Accessed: 10-04-2019.

Team, S. (2017). deep laceration. http://canacopegdl.com/keyword/deep-laceration.html. Accessed: 22-05-2019.

Techlabs, M. (2018). 8 best deep learning frameworks for data science enthusiasts. https://medium.com/the-mission/8-best-deep-learning-frameworks-for-data-science-enthusiasts-d72714157761. Accessed: 29-04-2019.

TensorFlow (2019). Object detection. https://www.tensorflow.org/lite/models/object_detection/overview. Accessed: 10-04-2019.

The Editors of Encyclopaedia Britannica (2019). Wound medicine. https://www.britannica.com/science/wound. Accessed: 22-03-2019.

the GAP, C. (2018). Absorbable sutures. https://lacerationrepair.com/wound-blog/absorbable-sutures/. Accessed: 22-05-2019.

The MathWorks Inc. (2019a). Image labeler. https://es.mathworks.com/help/vision/ref/imagelabeler-app.html. Accessed: 25-05-2019.

The MathWorks Inc. (2019b). Image segmentation. https://www.mathworks.com/discovery/image-segmentation.html. Accessed: 07-04-2019.

The MathWorks Inc. (2019c). Image segmentation. https://es.mathworks.com/discovery/image-segmentation.html. Accessed: 15-05-2019.

The MathWorks Inc. (2019d). Monitor deep learning training progress. https://es.mathworks.com/help/deeplearning/examples/monitor-deep-learning-training-progress.html. Accessed: 15-05-2019.

The MathWorks Inc. (2019e). Pretrained deep neural networks. https://es.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html#mw_6dc28e13-2f10-44a4-9632-9b8d43b376fe. Accessed: 25-05-2019.

The MathWorks Inc. (2019f). Semantic segmentation using deep learning. https://es.mathworks.com/help/vision/examples/semantic-segmentation-using-deep-learning.html. Accessed: 15-05-2019.

The MathWorks Inc. (2019g). trainingoptions. https://es.mathworks.com/help/deeplearning/ref/trainingoptions.html. Accessed: 25-05-2019.

The MathWorks Inc. (2019h). Transfer learning using alexnet. https://es.mathworks.com/help/deeplearning/examples/transfer-learning-using-alexnet.html. Accessed: 25-05-2019.

The MathWorks Inc. (2019i). unetlayers. https://es.mathworks.com/help/vision/ref/unetlayers.html. Accessed: 25-05-2019.

The MathWorks Inc. (2019j). valuatesemanticsegmentation. https://es.mathworks.com/help/vision/ref/evaluatesemanticsegmentation.html. Accessed: 30-05-2019.

Trust, D. N. Z. (2018). Dermatology image library. https://www.dermnetnz.org/image-library/. Accessed: 22-05-2019.

U.S. National Library of Medicine, L. H. N. C. f. B. C. (2018). Medpix. https://medpix.nlm.nih.gov/search?allen=true&allt=true&alli=true&query=wound. Accessed: 22-05-2019.

WoundEducators.com (2018). Wound assessment and documentation. https://woundeducators.com/wound-assessment-and-documentation/. Accessed: 22-05-2019.

Obligatory Signed Declaration

concerning this thesis titled

STUDY AND COMPARISON OF DIFFERENT DEEP LEARNING METHODS REFERING TO SKIN WOUND DETECTION

"I hereby declare that the present thesis was composed by myself and that the work contained herein is my own. I also confirm that I have only used the specified resources. All formulations and concepts taken verbatim or in substance from printed or unprinted material or from the Internet have been cited according to the rules of good scientific practice and indicated by footnotes or other exact references to the original source.

The present thesis has not been submitted to another university for the award of an academic degree in this form. This thesis has been submitted in printed and electronic form. I hereby confirm that the content of the digital version is the same as in the printed version.

I understand that the provision of incorrect information may have legal consequences."

Graz, July 2, 2019

(CHAIMAE KASSARA GUENNOUN)