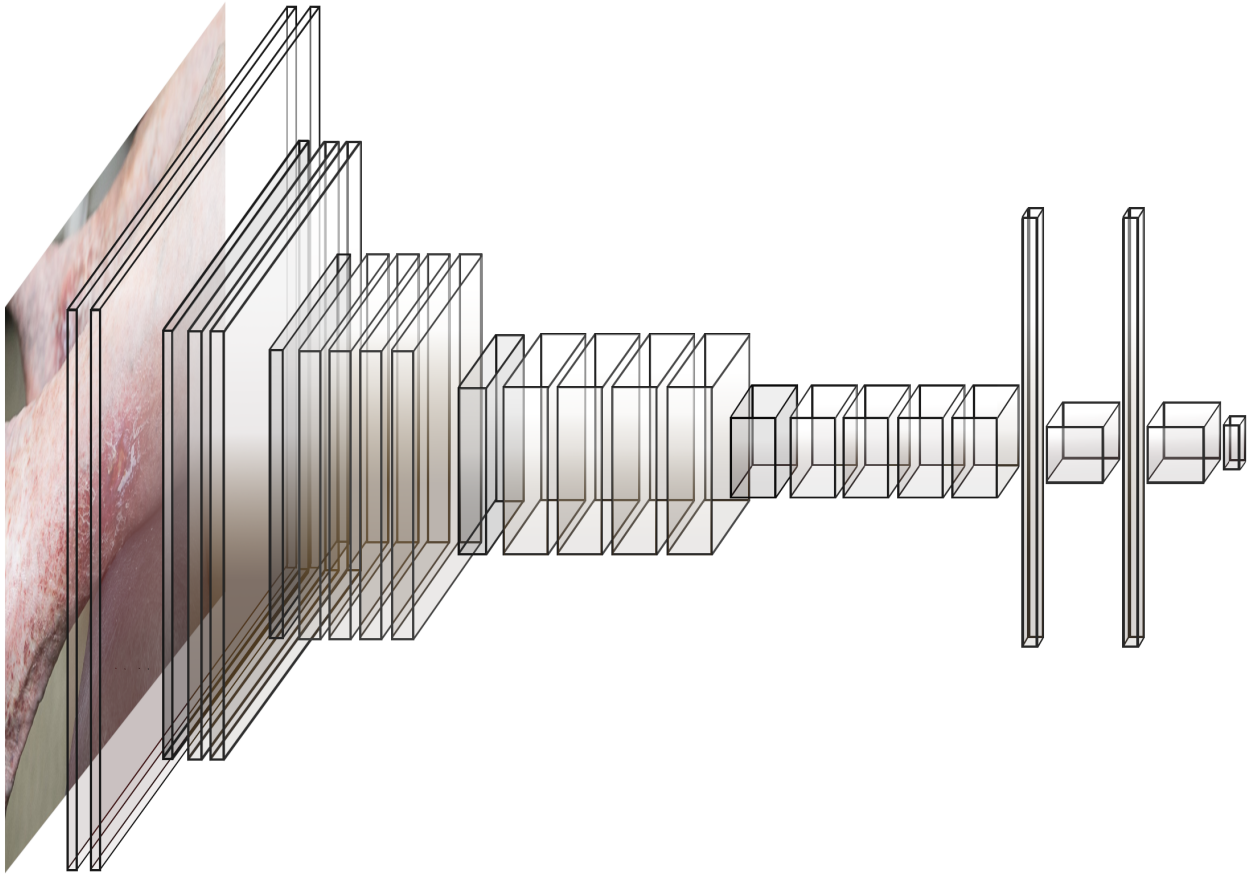
CHALMERS
UNIVERSITY OF TECHNOLOGY



Classification of Ulcer Images Using Convolutional Neural Networks

Master's thesis in Biomedical Engineering

CHARLOTTA AGUIRRE NILSSON
MEDINA VELIC

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis 2018:EX043

Classification of Ulcer Images Using Convolutional Neural Networks

CHARLOTTA AGUIRRE NILSSON
MEDINA VELIC

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Signal processing and Biomedical engineering
Computer vision and medical image analysis
Chalmers University of Technology
Gothenburg, Sweden 2018

Classification of Ulcer Images Using Convolutional Neural Networks
CHARLOTTA AGUIRRE NILSSON
MEDINA VELIC

Supervisors:
Gunnar Snorri Ragnarsson, QRTECH AB
Jennifer Alvén, Department of Electrical Engineering

Examiner:
Olof Enqvist, Department of Electrical Engineering

Master's Thesis 2018:EX043
Department of Electrical Engineering
Division of Signal processing and Biomedical engineering
Computer vision and medical image analysis
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A simplified model of a modified version of the VGG19 architecture used in the project. An image of a venous leg ulcer is received by the network as input, to visualise the path traveled by the image during classification.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2018

Classification of Ulcer Images Using Convolutional Neural Networks
CHARLOTTA AGUIRRE NILSSON
MEDINA VELIC
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The use of artificial intelligence has increased within a variety of different fields the last decade, including the area of health care. Machine learning algorithms have already been successfully used in e.g. skin cancer detection in images, indicating its potential for being applied to wound treatment by automatically classifying wound images. In addition to reducing patient discomfort, a machine learning based solution for wound image classification could lower the costs for treatment of wounds. Today, this cost corresponds to the third biggest health care cost for Västra Götalandsregionen, the health care sector in Västra Götaland.

This master thesis is a collaboration between the electronics and software development company QRTECH and Västra Götalandsregionen with the aim of creating an image-based aid to be used in the diagnosis of venous leg ulcers. A total of 300 manually annotated images depicting various ulcers for training a network have been provided by Alexandra Forssgren, specialist in dermatology and head of the dermatology department at Skaraborg Hospital. A deep learning based algorithm, using a pre-trained version of the VGG19 convolutional neural network, has been implemented to perform automated binary classification in order to separate venous leg ulcers from other ulcers. Since the annotated ulcer data set was limited, transfer learning was used by pre-training on ImageNet as well as on a set of dermoscopic skin images. The network was implemented in Python using the deep learning library Keras with TensorFlow as backend.

To improve performance and reduce overfitting, methods such as dropout, L1/L2 regularisation and data augmentation were applied. Augmentation included rotation, flipping, contrast enhancement as well as added noise and modified brightness. The best set-up was selected based on the performance metrics accuracy, precision and recall. For the test set, the final network reached an accuracy, precision and recall of approximately 85 %, 82 % and 75 %, respectively. Most certainly, the limited size of the data set affected the results negatively and a larger data set would increase the variety of the training examples, possibly leading to better performance. However, the performance metrics are still promising considering the small data set which proves the potential of a machine learning based solution for ulcer image classification.

Keywords: Deep learning, convolutional neural networks (CNNs), image classification, leg ulcers, transfer learning, VGG19, Keras.

Acknowledgements

A special thank you to our supervisors, Gunnar Snorri Ragnarsson at QRTECH and Jennifer Alvén from the Electrical Engineering department at Chalmers University of Technology. We wish to express our deepest gratitude for your guidance during the thesis, with exceptional expertise, regarding everything from the technical solution to the report. We would also like to thank you for being devoted and always eager to help, providing constructive feedback and valuable advice.

Alexandra Forssgren, specialist in dermatology and head of the dermatology department at Skaraborg Hospital, has provided us with images and shared her knowledge of leg ulcers which we are most grateful of. We would like to give a big thank you to everyone involved in the project, both at QRTECH and Innovationsplattformen from Västra Götalandsregionen, for a good collaboration. This collaboration is what made the master thesis possible.

We also want to show our appreciation to our examiner, Olof Enqvist, for agreeing to the role and always providing advice when needed.

Finally, thank you to everyone at QRTECH for a great work environment with all the tools required for completing the master thesis as well as all the support and enthusiasm from co-workers.

Thank you!

Charlotta Aguirre Nilsson & Medina Velic
Gothenburg, June 2018

Contents

Chapter 1

Introduction

The field of machine learning has developed tremendously the last decades and its area of application has expanded, ranging from social media to self-driving cars [1]. One reason for the field's broad area of use is time efficiency, an inherent property of machine learning algorithms. One possible area of interest for machine learning methods is the health care industry, where machines have proven to process patient data far more rapidly than humans [2].

This report presents the development of a tool, suggested to support medical personnel in wound diagnosis. The technical solution presented is an automated algorithm which can learn to recognise different types of wounds in digital photographs. In this chapter, wound treatment today will be introduced, focusing on potential improvements. The aim, research questions and limitations will be presented as well, introducing the proposed solution. Lastly, a section covering related research is included.

1.1 Background

One of the main advantages with machine learning methods is their ability to accurately recognise objects in images, making them suitable for object detection and classification. Machine learning is divided into different branches, out of which deep learning is one of these. Deep learning methods utilise artificial neural networks, originally inspired by the neural network in the human brain [1, 3]. A network can process several hundred images in seconds, becoming a useful tool for time-consuming tasks. For instance, a network can therefore be used within the field of health care to determine a diagnosis faster than any human [2]. A solution based on deep learning could also be a tool for smaller health care centers that lack clinical specialists [4].

Treatment of wounds is the third biggest cost for *Västra Götalandsregionen* (VGR), who are responsible for the health care and medical treatment in the county Västra

Götaland in Sweden [5]. This cost corresponds to 4 % of the total health care cost and hence, wound treatment is seen as an area of improvement by VGR [6]. Moreover, visits at the hospital for wound dressing two to three times a week may be necessary for some patients which increase costs and discomfort further [7]. As an early and correct diagnosis could lower the costs for wound care, introducing a technical tool for diagnosis could potentially lower the costs but also improve the life quality considerably for patients.

The focus for this project has been to identify venous leg ulcers in images. These ulcers are located below the knee as all leg ulcers, and arise from venous insufficiency which can be caused by widening of the veins. Another cause can be non-functioning valves, which are present in the veins of the legs to prevent blood to flow back towards the feet on its way to the heart. Non-functioning valves are not able to close completely to prevent the blood to flow backwards, causing high pressure in the veins. A wound is categorised as hard-to-heal if the prognosis of healing is more than six weeks. Out of all cases diagnosed with hard-to-heal leg ulcers, 38 - 60 % are caused by venous insufficiency [7]. Research has shown that correct and early treatment of the wound has shortened the healing time [8, 9]. In addition to this, treating venous leg ulcers with compression, such as stockings and bandages, will improve the healing process [10].

1.1.1 Wound Diagnosis

When manually diagnosing wounds, part of the examination is a visual inspection. However, additional parameters need to be taken into consideration as well when determining a diagnosis. Many of the considered parameters regard the medical history and the overall health status of the patient. As a part of the wound evaluation for venous leg ulcers, a so-called Doppler examination is also performed. In a Doppler examination, the pressure in arteries and veins is measured which can indicate the diseases causing the wound. All the parameters taken into consideration when evaluating a leg ulcer are summarised in Table 1.1 [11].

Table 1.1: All parameters taken into consideration when evaluating a leg ulcer in order to determine a diagnosis.

Health status	**Visual inspection**	**Doppler examination**
Profession, heritage, pregnancy	Depth, size	Arterial blood pressure
Other diseases	Redness, infection	Ankle-brachial pressure index
Wound duration	Fibrin, necrosis	
Former thrombosis	Surrounding skin	
Medication		
Social situation, smoking, diet		
Sleep, pain		
Mobility		

1.2 Aim of the Project

The aim of this master thesis is to present an automated algorithm that could aid in determining a diagnosis for patients suffering from hard-to-heal wounds, enabling faster decisions to be made regarding a proper treatment. By utilising image data, manual visual inspection normally performed when determining a diagnosis will be imitated. The image data provided consists of digital photographs, taken using both mobile cameras and digital cameras. Venous leg ulcers have been chosen as the main focus and the goal is to have an automatic algorithm for determining if a wound is a venous leg ulcer or not from image data.

1.3 Research Questions

The main question to be answered is whether a deep learning based solution can be used for diagnosis of venous leg ulcers from image data. Further, the method will be evaluated to investigate how well suited the proposed solution is for an implementation in the primary care and hospitals. The final question to answer is whether similar solutions have already been developed, investigating their level of success. In order to gain more in-depth knowledge within the field of deep learning, recent research where deep learning is applied to different medical applications, including wound treatment, will be studied.

1.4 Limitations

The aim is to implement a software limited to venous leg ulcer classification to fit the scope of the master thesis, a decision made unanimously together with VGR and QRTECH. The implemented software should therefore be restricted to perform binary classification only separating venous leg ulcers from other types of wounds. This limitation is motivated by the fact that this kind of ulcer is one of the most common among leg ulcers. Further, merely classification will be investigated, i.e. no segmentation of the wound will be included. In addition to this, only images and no other additional patient data will be used as input for the automatic algorithm as the time span for the project is limited.

1.5 Related Work

Deep learning for image classification is used commercially in everyday tools such as social media and companies' web pages, as well as in different fields of research ranging from the automotive industry to health care. To present a more elabo-

rate picture of how convolutional neural networks (CNNs), see Section 2.1.1, can be utilised for image classification, this section will describe previous research studies. The section will first cover how image classification is employed in different applications, then address suggested solutions specific to health care. Finally, papers related to the topic of classification of skin cancer and wounds will be discussed. Additionally, implemented smartphone and tablet applications for wound treatment will be presented.

1.5.1 Deep Learning for Image Classification

There is an increased general interest in self-driving cars and other autonomous systems, making it important to develop reliable machine learning methods. For instance, deep learning based image classification could be applied to traffic signs [12]. In a paper from 2016, the aim was to detect and classify traffic signs, where classification was performed using a CNN. The results presented an accuracy outperforming average human performance [12].

Deep learning based image classification can be applied to a variety of different images, including earth images taken from above, so-called aerial remote sensing images [13]. In this paper, only a limited data set was available and the authors therefore spent effort on developing a high performing network despite this limitation. The solution was to use two neural networks, whereof one was a CNN constructed by the authors utilised for classification. The authors found that the proposed solution achieved improved performance compared to previous studies using the same data set [13].

1.5.2 Deep Learning for Medical Image Classification

Another area of growing enthusiasm for applying deep learning to image classification is the medical field, including classification of CT images as well as ultrasound images. As described in an article from 2016 [14], pre-trained CNNs may be applied in order to classify whether CT images contain traces of Alzheimer's disease, tumours or normal ageing. Worth noting is that CT images are not considered when diagnosing Alzheimer's disease, but is rather used to eliminate other potential diseases. However, the results indicate the potential of utilising CT images for detection of Alzheimer's disease, achieving an average of 88 % accuracy for the three classes [14].

A similar approach was used in a research article for detecting early stages of thyroid cancer, using pre-trained CNNs for the purpose of classification [15]. The proposed binary classifier produced an output assigning ultrasound images to one of two classes, benign or malignant nodule. The accuracy reached 83 % but according to the paper, a greater set of training samples could possibly improve performance further [15].

Another example of deep learning applied to ultrasound images is a research paper from 2017 [16], which investigated how to classify liver fibrosis in ultrasound images of liver tissue. The paper suggested an image classifier taking three classes of tissue into consideration; normal, early-stage and late-stage fibrosis. The presented solution included a combination of two networks, one being a pre-trained VGGNet. For an in-depth description of one version of the VGGNet, VGG19, see Section 2.1.2. Moreover, the authors included a dropout layer after every fully-connected layer and augmented the training images by using methods such as flipping the images vertically and horizontally. They worked with a limited data set, but by implementing augmentation they could extend the training set. The conclusion was that one can reach better performance with a deeper network and with a greater data set [16].

1.5.3 Deep Learning for Skin Image Classification

A medical application for classification of images using CNNs is the detection of skin cancer. In a study from 2017 [17], the performance of a computer-aided classifier for skin cancer severity was compared to a set of manual classifications done by dermatologists. For a set of images, both photographic and dermoscopic, the network outperformed the average of all dermatologists in the task of deciding whether the lesion should be treated or not. The network used was a pre-trained version of Google's Inception V3, which was first introduced in [18], where all layers were retrained on nearly 130 000 images of skin lesions [17].

Classification using CNNs has also proved to be useful for identifying skin damage caused by pressure injuries [19]. In this paper, the studied wounds contained different types of tissue, demonstrating both infection and different stages of healing. Images were cropped into patches, containing only one tissue type each. Using a CNN, classification was performed in order to assign each patch a tissue type. By classifying each patch, regions of a certain tissue could be recognised in the original images and hence, the entire image could be segmented [19].

1.5.4 Mobile and Tablet Applications for Wound Care

Despite their promising results, the mentioned advances in research regarding deep learning based skin image classification are yet to be implemented in health care. However, a number of mobile and tablet applications for wound care have been developed over the last few years [20–22]. Even though these applications do not use fully automated algorithms, they offer personal assessments for patients without the need for patients leaving their homes as well as tools for health care professionals to ease their workload.

WoundDoc is an application offering consultations regarding different wounds, proposing a treatment plan that can be sent to the patient's medical practitioner [20]. In contrast to WoundDoc, *WoundDesk* is a tool to be used solely by health care

professionals. The application can help in keeping track of all patients and their chronic wounds, by offering a place for documentation but also tools such as semi-automated wound surface measurements [21]. Another application for health care professionals is the Swedish *Dermicus Wound*, facilitating diagnosis and decisions regarding the most appropriate treatment for hard-to-heal ulcers. These decisions are made by skin specialists that can be notified by the primary care in case of a new patient [22].

1.5.5 Concluding Remarks

A common factor among the aforementioned papers is the use of pre-trained networks to improve performance, all pre-trained on the database ImageNet [23]. Furthermore, an issue often encountered in the medical papers was the challenge to get access to a big enough data set due to patient confidentiality, indicating a common problem when using medical data in machine learning. As indicated by many of the authors, the hope for the future is to be able to train the networks with a more extensive database and hence, achieve better performance. Moreover, the mobile and tablet applications mentioned demonstrate the potential of eHealth based solutions regarding wound treatment. The company GNOSCO, responsible for the Dermicus Wound application, recently announced the start of a project to develop an automated algorithm for early detection of skin cancer from images using artificial intelligence [24]. This affirms the possibility of developing completely automatic, mobile solutions arising with advances in machine learning.

1.6 Thesis Outline

To cover the reminder of this thesis, the report is divided into the following chapters; Theory, Data and Methods, Results and Concluding Discussion.

The theoretical background to machine learning is introduced in Theory, Chapter 2. In this chapter, the building blocks for the chosen algorithm are presented. All details regarding the implementation are covered in Data and Methods, Chapter 3. The results from implementing the chosen machine learning algorithm to the task of classifying wound images are presented in Results, Chapter 4. The results are discussed in Concluding Discussion, Chapter 5, including factors affecting the performance of the algorithm. Furthermore, future work will be mentioned in Chapter 5 as well, including additional improvements which could be taken into consideration for future developments. A more extensive documentation of all the results during the master thesis will be found in Appendix A, where additional graphs and tables are found.

Chapter 2

Theory

Artificial intelligence can undoubtedly contribute to solving many problems today, saving both time and resources. For this to be achieved, machines need to be able to solve simpler tasks which include tasks among the most trivial to humans. Image classification is a typical example of this kind of task, where classifying the contents of an image may seem simple for a human but is more demanding of a task for a computer. Deep learning, a field within machine learning, embodies a number of strategies for teaching machines how to solve these challenges [25].

The following components can be considered the building blocks of a machine learning algorithm; a specification of a task, a cost function, a learning method and a model [25]. The building blocks listed will be introduced in this chapter, establishing the theoretical foundation for creating a binary classifier with the aim to classify wounds. Furthermore, additional subjects regarding implementation will be covered, including implementation frameworks, common pre-trained networks and fine-tuning.

2.1 Artificial Neural Networks

Artificial neural networks are inspired by the human neural network, composed of neurons responsible for transferring information to be processed in various parts of the human brain. By interacting with other neurons, each neuron receives multiple signals which can be weighted according to their importance. If the sum of all inputs exceeds a certain threshold, the neuron will fire and send a signal forward. This behaviour can be described by a simplified mathematical model, according to the following equation,

$$y = a\Big(\sum_{i=1}^{n} w_i x_i + w_0\Big). \tag{2.1}$$

In the equation, the inputs to the neuron are denoted x_i, index i representing a specific input. As seen in the equation, a total of n inputs are received by the

neuron. The weight associated with each input is denoted by w_i while w_0 represents a bias term. The function a is the activation function, mimicking the activation of a neuron sending a signal forward. The signal is represented by the right hand side of the equation, which will be received as input by another neuron. An artificial neural network is a collection of these artificial neurons, stacked into layers similar to its biological counterpart [25, 26]. In the layers of an artificial network, these neurons are seen as individual units, working in parallel. In the case of a larger number of stacked layers, the network is classified as a deep neural network [25].

2.1.1 Convolutional Neural Networks

A feedforward network, which is an artificial neural network according to the previous definition, is one important model within the field of deep learning. For a network to be classified as feedforward, the information passed through the network needs to be passed forward through the network without feedback loops. An example of a feedforward network is the convolutional neural network (CNN). These networks are used extensively in computer vision and as the name might suggest, CNNs utilise the convolution operation [25].

Convolution is utilised in the convolutional layers of a CNN, by the use of kernels. 2D convolution is applied similarly to applying a filter to an image in image processing, enabling the kernels to detect different features. Each kernel of the layer, typically much smaller than the input image, will give rise to a feature map. The size of the feature map will be equal to the image if zero-padding is used and slightly smaller if zero-padding is neglected, excluding the edges. The found features represent different structures in images and are thus dependent on the elements of the kernel, the so-called weights. The weights are the parameters of the model, optimised during training of a network. In the early layers of the network, the kernels recognise basic features and simple shapes such as blobs and lines while more specific features are recognised further into the network [3].

An advantage of using artificial neural networks as deep learning models is the level of complexity that can be described when compared to linear functions. In order to ensure non-linearity, a non-linear activation function is used for mapping the filter output, typically after each layer. The most common is the rectified linear unit (ReLU). This is a piecewise linear function, enabling it to possess some of the advantageous properties of linear functions. For example, it preserves the ability of a linear model to generalise well and to be easily optimised using gradient-based methods [25].

Another non-linear operation typically included in a CNN is pooling. A pooling function is applied to a smaller region of the feature maps, and computes a single, representative value for each region. One example is max pooling where the maximum value from a smaller region of the input is determined and processed as the new output. By introducing a pooling layer, an invariance will be introduced which enables small translations to be present in the input without affecting the output

tremendously. This property is valuable when a feature can be detected without its exact location being of high importance. Pooling increases computational efficiency since it can be viewed as a down sampling, reducing the resolution of the feature maps. Furthermore, pooling will reduce model complexity which in turn reduces the risk of overfitting [25].

For image classification tasks, one or more fully-connected (FC) layers can be added in the end of the network, performing a linear combination of all outputs from the previous layer [3]. Finally, the output layer of the network needs to be adjusted in order to comply with the task of the network. When performing classification, the output could for example represent the probabilities of the image belonging to each class. In the case of a binary output, a sigmoid function can convert the network output to a probability. In the case of multiple values, the softmax function will output a probability distribution over all possible outcomes. Hence, the sigmoid or softmax function will constitute the final layer of a network [25].

2.1.2 VGG19

A CNN that has proved to be applicable for image classification is VGG19. This network is derived from the network architecture VGGNet, where the number 19 represents the number of weight layers present in the structure. The VGGNet was designed and presented by Simonyan and Zisserman for their contribution in the ImageNet Challenge 2014 [27].

The traditional building blocks of a CNN, described in Section 2.1.1, establish the layers of VGG19. The network consists of five blocks of convolutional layers, where the filters are of size 3×3. The five blocks are separated by max pooling layers and lastly, three FC layers are added. The last layer of the FC layers produces the network's predictions as output. The number of channels correspond to the number of classes in the predictions, and softmax is used as activation function in order to return a probability [27].

2.2 Learning

In the previous section, the artificial neural network was introduced as a deep learning model. The purpose of this model is to describe the relationship between a given set of input and output data but in order to accurately map this relationship, the model needs to be trained on the task. Learning can be categorised into two subgroups; supervised and unsupervised learning. In supervised learning, a desired output is given for each input. In unsupervised learning on the other hand, target labels are not given but the algorithm learns properties from the structure of the data set [25]. In this thesis, only supervised learning will be used.

Training of a neural network is performed in cycles where one cycle is characterised

by passing the entire data set through the network, denoted as an epoch. The goal of training is to find the optimal parameters for the model, and as mentioned when describing the convolutional neural network, the parameters of the model are the weights of the kernels [25]. The weights should be initialised before training, preferably randomly to ensure diversity among the gradients in parameter updates [3]. The optimal weights are found through the process of optimisation, minimising a cost function. The weights will be updated based on the value of the cost function, but the goal of the training process is not to reach the minima per se. Minimisation is a tool for reaching the overall goal which is improved performance on the specified task. In order to measure how well the algorithm performs, performance needs to be quantified by a performance metric [25]. If the performance metrics start to stagnate, converge, it is fair to assume that the network will not perform better than it has done up to that epoch and the training is considered finished.

2.2.1 Cost Function

In machine learning, a common cost function to minimise is based on maximum likelihood estimation. The function to be minimised will therefore be the negative log-likelihood, which can also be considered as the cross-entropy between the distribution of the training data and of the current model output [25]. In the case of only two classes, binary cross-entropy is given by

$$J(y, \tilde{y}) = -\Big(y \cdot log(\tilde{y}) + (1 - y) \cdot log(1 - \tilde{y})\Big), \tag{2.2}$$

where y is the true probability derived from the training data and $\tilde{y}$ is the output probability from the network. The cost function is typically also referred to as the loss function.

2.2.2 Optimisation

In most cases, the goal of optimisation is to minimise a cost function by changing the model parameters. A tool to use for this purpose is therefore the derivative of the cost function with respect to the model parameters, as it is a natural choice for quantifying the response for a change of the parameters. If the purpose is to reach the cost function minimum, a suggestion is to change the parameters in the opposite direction of the cost function derivative. This process is the very fundamental idea of gradient descent, which is the foundation of many optimisation methods. The size of the steps taken in the desired direction is named learning rate, a parameter set before learning. In machine learning terms, this kind of tuning parameters are collectively referred to as hyperparameters. To allow the value of the loss function to alter the values of the parameters, gradients need to be calculated in parameter updates. The method of back-propagation is used for this purpose [25].

With gradient descent as starting point, many improvements have been made in the development of other optimisation methods. In contrast to gradient descent where

all samples are used for computing the gradient of the cost function in each parameter update, one improvement includes performing updates by drawing a batch of samples from a training set. In minibatch stochastic gradient descent, a gradient estimate is computed for the current batch and each parameter in the model is then updated according to the gradient. This introduces another hyperparameter, the batch size, to be determined before learning. Parameter updates can also be performed for each sample, an approach used in stochastic gradient descent (SGD). One of the method's main advantages is its ability to maintain the same computational time per update regardless of the number of samples. This will enable quicker convergence even if the number of samples is large. Still, SGD can have relatively slow convergence and accelerating convergence further can be achieved using methods such as momentum. In momentum, an average of past gradients is accumulated during parameter updates. This enables past gradients to be considered during new parameter updates, and their contribution is determined by a hyperparameter [25].

As a hyperparameter will need to be adjusted in order to achieve an adequate effect of momentum, alternative methods for faster convergence without the need of adding hyperparameters are preferred [25]. An alternative is to use an adaptive step size, or learning rate, for each parameter in the model. One of the methods utilising this kind of adaptive learning rate is Adam optimisation, a method developed with the aim of being used for machine learning applications [28]. In addition to its potential of faster convergence, Adam optimisation is regarded as more robust considering the choice of hyperparameters which makes the method a possible candidate for challenging SGD [25].

2.2.3 Performance Metrics

Machine learning is driven by the improvement of performance, which is accomplished through the process of optimisation. In order to quantify learning, a performance metric is used. The performance metric set to evaluate the learning progress is normally accuracy or error rate. In the case of classification, this corresponds to the proportion of outputs correctly or incorrectly classified. In addition to this, precision and recall are also commonly used as performance metrics. Precision reports the fraction of correctly classified samples among all samples classified as positive. Recall on the other hand reports the fraction of correctly classified samples among all samples that truly are positive. Precision and recall can be summarised as follows,

$$\text{Precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}, \tag{2.3}$$

$$\text{Recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}. \tag{2.4}$$

In the equations, positive and negative denote the two classes considered during binary classification. True positive is the number of samples correctly classified as positive. False positive is the number of negative samples incorrectly classified as positive, and vice versa for false negative.

2.2.4 Regularisation

As mentioned previously, the progress made in learning is quantified by a performance metric which enables any improvement to be detected. However, the aim is to have an algorithm with preserved performance even when encountered with new, unseen data. In addition to a decreased training error, minimised in the process of optimisation, a low generalisation error is desired as well. This generalisation error is defined as the expected value of the error made by the algorithm when encountered with new data [25].

The size of the generalisation error is retrieved by testing the algorithm on a validation set, separated from the training set. The magnitude of the errors and their relationship can be used to investigate the behaviour of the algorithm. A large training error indicates underfitting while a large gap between the training error and generalisation error is a sign of overfitting, representing two major challenges in machine learning. When it comes to overfitting, there are a number of different strategies for decreasing an algorithm's generalisation error while keeping the training error unaffected. These methods are collectively referred to as regularisation methods [25].

Approaches commonly used to decrease the generalisation error, or validation error, include data augmentation and dropout. Augmentation is the very intuitive solution to expand the data set. For the purpose of image classification, new images can be created by altering the images already present in the data set. Possible augmentations include alterations such as rotation, adding noise, scaling and flipping vertically and horizontally. However, a certain degree of resemblance is to be maintained between the original data set and the additional training samples created by augmentation. Therefore, it is important to consider the nature of the original data when determining the type and amount of augmentations added. Dropout on the other hand, enables a number of subnetworks to be trained as different units are excluded from the network. The output units are kept intact, and the units are removed by setting their output to zero. The probability of setting an output to zero is a hyperparameter determined before training begins. Fortunately, the probability that all connections between input and output will be erased will be insignificant as long as the network is large enough [25].

Another regularisation strategy is to add a penalty to the loss function [25], according to

$$\widetilde{J}(\mathbf{w}; x, y) = J(\mathbf{w}; x, y) + \alpha\Omega(\mathbf{w}), \tag{2.5}$$

where J is the original loss function. The penalty term is denoted by Ω and α is a hyperparameter determining its contribution with $\mathbf{w}$ denoting the network parameters. Different penalty terms could be used, for example L^1 and L^2 norm according to

$$\Omega(\theta) = ||\mathbf{w}||_1, \tag{2.6}$$

$$\Omega(\theta) = \frac{1}{2}||\mathbf{w}||_2^2. \tag{2.7}$$

As a penalty term, Equation 2.7 forces the network parameters closer to the origin, the network parameters will therefore shrink with each update. Equation 2.6 regularises the size of the weights as well but can also introduce sparsity, parameters with the value of zero. This property enables the model to separate the important features from the less important, performing a so-called feature selection.

2.2.5 Transfer Learning

As mentioned previously, one of the building blocks of a machine learning algorithm is the specification of a task to be solved, learned by the algorithm in the process of learning. However, the assignment might be too difficult without any reference regarding desired features and convergence during optimisation might be hard to reach. Therefore, one alternative is to solve a simpler task first. Pre-trained networks are appropriate to use for this purpose, being trained on a different data set containing enough information for the network to perform well on the simpler task. In this way, the output weights from the pre-training can be used for initialisation instead of selecting random starting points. For images, basic shapes and structures such as blobs and lines are recognised in the earlier kernels of the network as mentioned in Section 2.1.1. The detection of these basic features will be utilised, improving the network's performance since it does not need to learn all features from scratch. Some examples of databases used for pre-trained networks are ImageNet, WordNet and MNIST depending on the task of the network, whether it is to determine or classify images, words or numbers [25].

Pre-trained networks for image classification are usually trained on the huge image database ImageNet, which contains millions of images portraying different objects such as different mammals, cars, flowers, etc. [23, 29]. ImageNet was introduced in a paper from Princeton University in 2009, emphasising the necessity to gather different kinds of information, e.g. images, in one easily accessible place [29]. ImageNet arranges a competition every year where the challenge is to correctly classify different pictures. Some of the participants have published their network architecture and among the most successful CNN architectures are AlexNet, GoogLeNet, ResNet and VGGNet [27, 30–32]. The architectures are open to use as pre-trained models to be applied for different types of classification.

Employing the weights from a pre-trained network to continue training on new data is referred to as fine-tuning, and is especially well suited for training CNNs as these require large amounts of training data and computational power [33]. Fine-tuning is therefore usually applied when working with a limited set of data, as this might not be necessary with a large enough data set [34]. In order for the pre-trained model to be compatible with the new task of the network it is sometimes necessary to modify some of the layers in the architecture. The pre-trained network might have been trained on a different number of classes than the ones required by the current task and if so, the final layers are adjusted to fit the number of classes needed. Furthermore, it could be favourable to only retrain parts of the network during fine-tuning. As more general features from the data are recognised in the

early layers of a CNN, it is generally redundant to include more than the final layers in the fine-tuning. If the task of the CNN is to classify images that differ significantly from the images the pre-trained network was trained on, it can be beneficial to train more layers. A suggested fine-tuning technique is to gradually add layers until the performance of the CNN is improved [33].

2.3 Deep Learning Frameworks

A machine learning framework is a tool generally applied when constructing a neural network, as it provides the structure in which the calculations for training are performed. There are different frameworks available, some examples are Caffe, TensorFlow, PyTorch and Theano. In Python, the frameworks are modules that can be imported to a machine learning script where different commands or operations from the framework are implemented.

Caffe is an example of a framework, compatible with C++, Python and MATLAB. Available functions within Caffe will help the user to train, test and fine-tune their network, and pre-trained models are also provided. Additionally, since it is open source it is possible to find the code online and even contribute to the script. Users can design a network from scratch using Caffe's layers, since all types of CNN layers are accessible [35].

As Caffe, TensorFlow provides tools for constructing a CNN, both pre-trained and built from scratch. TensorFlow is based on the system DistBelif which works similarly to the Caffe model. TensorFlow is designed to be more efficient and flexible than its ancestor. An important difference is that TensorFlow uses mutable data, meaning that the object can be changed even after declaration. With this framework it is possible to try different set-ups since the neurons are updated and represented as tensors [36].

Another alternative is to use a machine learning library with a framework already implemented, an example of such a library is Keras. This library is constructed in Python and has three frameworks available; TensorFlow, Theano and CNTK. Most tools that might be used when designing a neural network is implemented in Keras, from pre-trained networks to loss functions. The process of implementing a machine learning algorithm can be made simple and straightforward using Keras as an interface, due to its many useful features. The built-in package *Applications* can be used to retrieve different pre-trained networks, allowing users to test different networks in order to choose the most suitable architecture. By calling the already defined classes or functions in the library, evaluating different designs and architectures is not as time-consuming as if the users would have designed the network themselves [37].

Chapter 3

Data and Methods

As the aim of this project was to develop an algorithm for identifying venous leg ulcers in images, a CNN was employed in order to perform binary classification. The training was performed on a computer with graphics card NVIDIA Titan X with a memory of 12 GB. This chapter presents all details regarding the processing of data and implementation of the algorithm. First, a description of the data will be given together with the data collection and pre-processing of the images. The implementation of the algorithm will be presented as well, including chosen framework, network architecture and the training of the network.

3.1 Data

Two data sets were used for training the network, including a separate set for pre-training in addition to the images depicting different ulcers. The network was pre-trained using dermoscopic skin images, retrieved from the open source archive ISIC. This database is a part of the ISIC project, aiming to reduce the number of unnecessary biopsies and deaths caused by melanoma [38]. A total number of 1995 skin images were used, depicting melanomas in different stages but also benign skin lesions. Each image belonged to one of two classes; benign or malignant, and was given a label accordingly. A total of 1591 benign lesions and 404 malignant were divided into two data sets, 80 % for training and 20 % for validation.

Various images of patients' ulcers were provided by head of the dermatology department at Skaraborg Hospital in Skövde, Alexandra Forssgren. A large part of the data obtained was collected from prior research studies of hers but she also collected new images during the thesis from patients with their consent. Additional images were also obtained from the web page *Sårwebben*, published by Skaraborg Hospital. A total of 300 images were accessed for training the network, depicting both venous and non-venous leg ulcers. The data set was divided into a training and validation set corresponding to approximately 85 % and 15 % of the data set, or more exactly 252 and 48 images, respectively. For some cases there were multiple images of the

same patient. If the same ulcer was visible in several images, the images were divided ensuring that this ulcer was not encountered during both training and validation. In total, images of 163 venous leg ulcers and 137 non-venous ulcers were obtained where both sets contained approximately equal amounts of both type of images. The training set consisted of 55 % venous leg ulcers and 45 % non-venous ulcers while the validation set contained exactly 50 % of each class. The non-venous samples contained images of wounds such as arterial leg ulcers, diabetic and cancerous wounds. As some images consisted of several ulcers, images were in some cases also cropped to separate the wounds. An additional set of 10 images was kept separate for testing the network. To increase the test set, images from a document published for health care personnel in the region of Östergötland were used [39]. By adding these additional images, the test set increased to a total of 34 leg ulcer images, 22 non-venous and 12 venous.

A variation could be seen in the obtained images regarding camera distance, light setting and angle. Some of these variations are visible in the images below, see Figure 3.1. To allow the ulcer to be the main part of the image, the images were pre-processed to remove parts of the background. Furthermore, data augmentation was performed as the data set provided was limited, in order to extend the data set and improve performance during training. For a thorough description of the augmentation, see Section 3.3.1.

Figure 3.1: Depicting the variation seen among the received images.

3.1.1 Pre-Processing

Upon receiving the data, a MATLAB script was constructed for pre-processing. The script was utilised to save appropriate coordinates for cropping together with the centre point of the wound in a separate file. The coordinates and centre points were applied in the script for augmentation, ensuring the images would be cropped according to the saved coordinates before being fed into the network. The centre points were utilised when rotating the images, as the centre of rotation. Moreover, all images were normalised before training in order to maintain all data within the same unit interval.

3.2 Network Implementation

The selected framework was TensorFlow, used as backend together with the machine learning library Keras. For this project, different networks were retrieved from the module Applications as mentioned in Section 2.3, including VGG16, VGG19, Inception V3 and ResNet50. In the early phase of the project, additional tests were done with the NVIDIA Deep Learning GPU Training System, DIGITS, with Caffe as framework [40]. DIGITS provides the opportunity to try the networks LeNet, AlexNet and GoogLeNet, pre-trained on a suitable database such as ImageNet or MNIST [40]. Smaller tests were performed using the received data set on the different networks, before settling for the VGG19 network using TensorFlow as framework. Training this network indicated trends of a declining loss and an increasing accuracy early on, producing the most promising results.

3.2.1 Baseline

The pre-trained network chosen was VGG19, and it was implemented using weights from pre-training on the ImageNet database. In order to classify images using binary classification, the final layer of the network was modified to only handle two classes and the activation function was changed from softmax to sigmoid. To avoid overfitting, two dropout layers were added after the first and second FC layer. The final network architecture, with modifications, is displayed in Figure 3.2.

3.3 Network Training

In order to train a network using Keras, a training set-up needs to be compiled where settings such as optimiser, learning rate and loss function are to be set. The same settings were applied in both pre-training using skin images of melanomas and fine-tuning using images of ulcers. Adam optimisation was used together with binary cross-entropy as loss function. Furthermore, the additional regularisers L1, L2 and

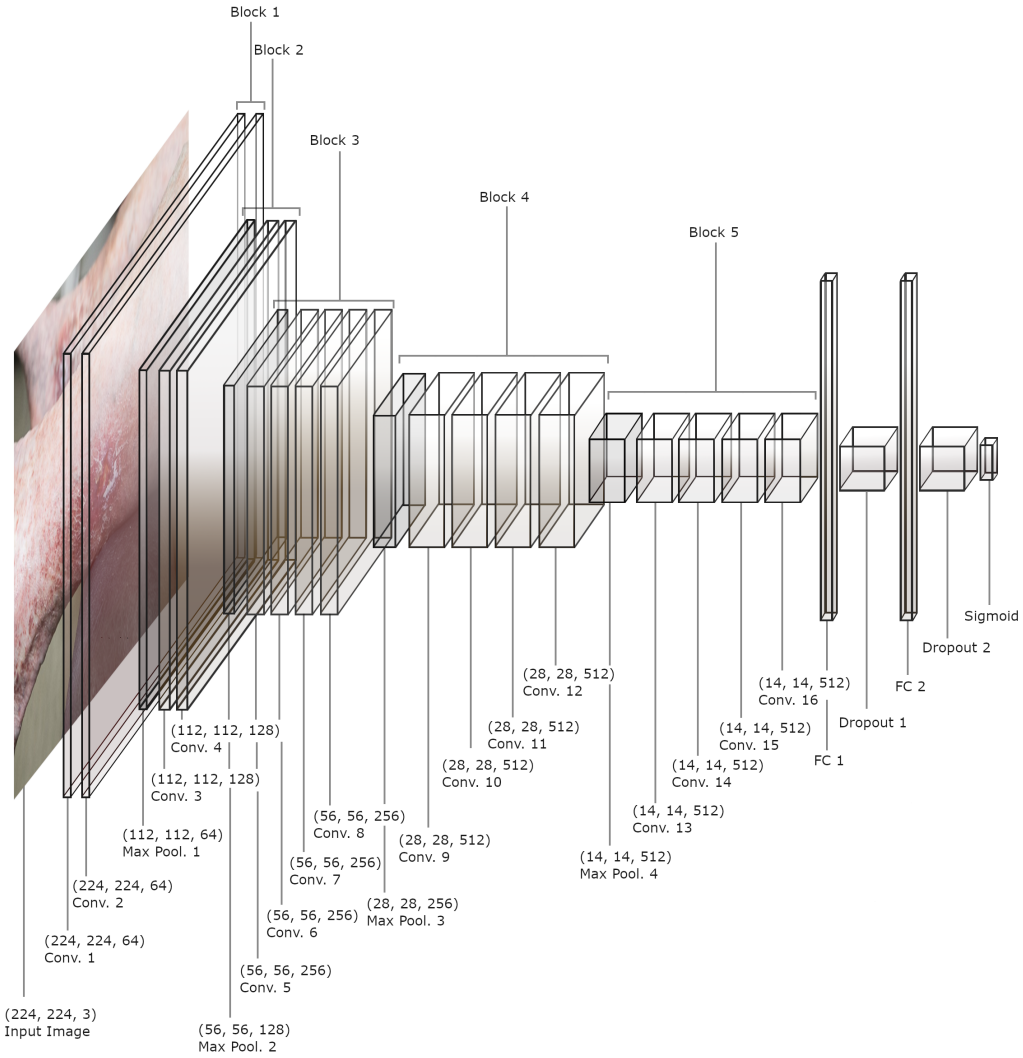


Figure 3.2: The network architecture used, with VGG19 as foundation but with added dropout layers and softmax function changed to sigmoid. All convolutional blocks are shown, as well as the sizes of the individual layers. The input image in the figure is from the training set.

L1 together with L2 were also added to further reduce overfitting tendencies. The hyperparameters set before training were number of epochs, learning rate, learning rate decay, class weight for the loss function, regularisation parameter and dropout level. Different levels of each hyperparameter were tested and adjusted several times, retraining the network every time to evaluate the performance metrics, before the most suitable levels were determined.

In addition to being pre-trained on ImageNet, the network was trained using a set of dermoscopic skin images from the open source archive ISIC before encountering the ulcer images. The data set used for the pre-training was not balanced and the loss function was rescaled with a weight factor corresponding to the class ratios to ensure equal importance for both classes during training. The number of epochs for the pre-training was decided based on the time required to finish one epoch, otherwise it

would have taken too long before being able to run tests. For this reason, the epochs for the pre-training was set to 50. Regarding fine-tuning using the wound images, the number of epochs was originally set too high, to 200 epochs, but was adjusted based on the convergence of the performance metrics. The number of epochs for fine-tuning was therefore finally set to 50 as well. For both pre-training and fine-tuning, the training and validation sets were evaluated using loss, accuracy, precision and recall. The performance of the final network structure and hyperparameter set-up was finally tested using a smaller test set of images, 10 skin lesion images for the pre-trained network and 34 ulcer images for the fine-tuned network.

When retrieving images, Keras allows real-time processing of the images as they are collected from the correct folder by the use of a generator. This tool enables augmentation to be performed by Keras as well, without the need of saving every augmented image. However, as the images were cropped before being fed to the network, the rotation provided by Keras caused artifacts in the images such as lines and black edges. As this pre-implemented augmentation did not yield desired results, as the produced artifacts could potentially affect the training of the network, the function for augmentation was rewritten. An elaborate description of this customised augmentation will be covered in Section 3.3.1. Additional modifications were made to some of the pre-implemented functions in Keras, including a customised callback function in order to evaluate the model with precision and recall.

3.3.1 Augmentation

For protecting patient integrity, the access to patient data is restricted. Due to this reason, the data set received from Skaraborg Hospital was limited and data augmentation was implemented as a solution to the problem. New data was thereby created from the original data set using modifications such as rotation and colour alterations. However, images of ulcers are sensitive to excessive modifications as the wounds' characteristics must be preserved. One example of when this might be an issue is when altering an image using contrast enhancement or other colour alterations. The wound might become unnaturally bright red, or not red enough, when compared to the original image and hence, this can cause the network to train on implausible cases. Therefore, the augmentation was restricted to rotation, flipping the images vertically and horizontally, slightly changing the brightness and contrast and adding noise. As with changing the brightness and contrast, the amount of noise added was also restricted in order to not alter the image beyond resemblance to its original.

For adding noise as well as adjusting the brightness and contrast in Python, the module `skimage.exposure` was used. Different parameters for specifying the augmentation was chosen randomly from a predefined interval. To adjust the brightness, a gamma correction was added with the module `skimage.exposure.adjust_gamma`. To obtain a darker image, the allowed interval for the gamma value was 0.7 - 0.9, and for a brighter image the range was 1 - 1.1. In addition to modifying the gamma value for a brighter image, the gain was also adjusted to increase the brightness

intensity. For this purpose, the interval of the gain factor was set to be 0.85 - 0.95. For achieving a darker image the gain factor was set to the default value, one. The contrast of the image was modified by changing the scale of the intensity levels, using the Python module `skimage.exposure.rescale_intensity`. An interval was set to determine an intensity range, where the lower boundary was always set to zero and the upper boundary was set within the range 208 - 238.

The module used for adding noise was `skimage.util.random_noise` and the types of noise added were speckle, Gaussian and Poisson noise. Speckle noise, or multiplicative noise, is the result of adding an image multiplied with uniformly distributed noise to the original image [41]. Mean was set to zero for speckle and Gaussian noise. The variance was selected randomly within the interval 0.02 - 0.1 for speckle noise and 0.009 - 0.04 for Gaussian noise.

Some examples of the added augmentations are visible in the images below. Two images of the same ulcer are shown in Figure 3.3, comparing the original to a case with three augmentations added which was the maximum number of augmentations added. Image (B) in Figure 3.3 shows the augmented image with the most extreme modifications regarding gamma and gain for brightness as well as the variance for Gaussian noise. Moreover, this image is flipped compared to the original photo in Image (A). Individual examples of the most extreme cases are shown in Figure 3.4, where it is possible to see how the same image differs when the maximum value of brightness, darkness, Gaussian noise and contrast are added.

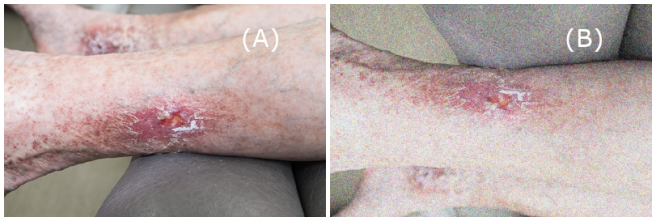


Figure 3.3: Depicting the effect of adding a number of augmentations to an image. (A) The original image of the ulcer, with no augmentations added. (B) The resulting image after adding the maximum number of augmentations. The augmentations added include; flipping the image, adding maximum value of Gaussian noise and increasing the brightness to its maximum.

For rotation, the coordinates for cropping were taken into consideration in order for the images to be rotated before cropping. Rotation was performed using a rotation matrix and applying it to every coordinate of the image, and selecting the corresponding pixel intensity after rotation. Moreover, the coordinates for the centre of the ulcer were used as the centre of rotation. The rotation matrix used is described by the following matrix,

$$R = \begin{bmatrix} \alpha & \beta & (1-\alpha)\cdot C_x - \beta \cdot C_y \\ -\beta & \alpha & \beta \cdot C_x + (1-\alpha)\cdot C_y \end{bmatrix}, \tag{3.1}$$

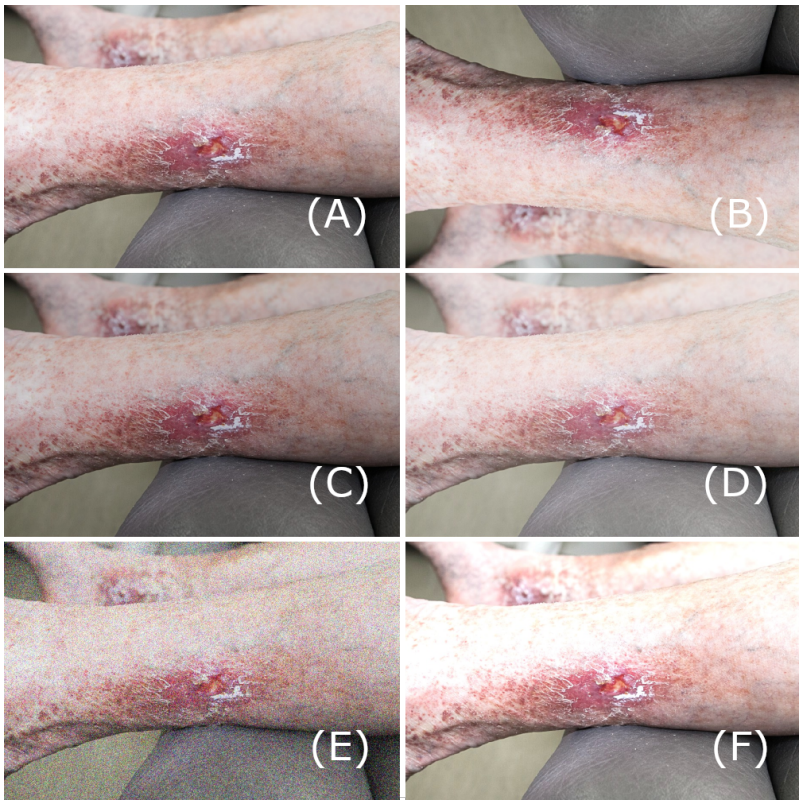


Figure 3.4: A selection of the possible augmentations. (A) The original image depicting a venous leg ulcer. (B) The resulting image after flipping the image. (C) The result after darkening the image, using gamma 0.7. (D) The resulting image after brightening the image, with gamma set to 1.1 and the gain factor to 0.85. (E) The image after adding Gaussian noise to the image with a variance of 0.04. (F) The resulting image after changing the contrast with a intensity range 0 - 208.

$$\alpha = \cos(\theta),\ \beta = \sin(\theta). \tag{3.2}$$

C_x and C_y represent the x and y coordinates for the centre of the ulcer, and θ denotes the angle of rotation. An angle was chosen randomly and the coordinates for cropping were rotated first, to ensure that they remained within the boundaries of the image before rotating the rest of the image. As many of the images were cropped closely to the leg in order to remove unnecessary background information, a large rotation could cause large parts of the leg to be excluded from the image. In order to ensure that a larger part of the ulcer and leg was visible after rotation, the angle was limited to the range of 5 - 50 degrees. If rotation was not possible, a flipping of the image was performed instead.

As with the ulcer images, the dermoscopic images depicting melanomas used for pre-training were not to be altered drastically in order to ensure resemblance between the original and newly added images. Thus, the same set of augmentations were applied as for the ulcer images. However, as the images depicting melanomas were not cropped, neither were they rotated.

A customised function was created in Python for augmentation, having the coordinates for cropping of the images as input together with the image itself. During training, the amount of data augmentation was chosen randomly for each input image which resulted in a slightly different image and therefore, a slightly different set of images for each epoch. The number of different types of augmentation techniques to use for each image was chosen at random, but could never be more than three. No augmentation was performed on the validation set, but all images were resized to the size of $224 \times 224 \times 3$.

Chapter 4

Results

This chapter presents the results of the project, including both pre-training using dermoscopic images and fine-tuning using ulcer images. Different combinations of regularisers and levels of dropout were tested for fine-tuning. All hyperparameters were manually tuned in order to achieve satisfactory learning.

The hyperparameters considered were; epochs, batch size, learning rate, learning rate decay, class weight, steps per epoch and regularisation parameter. The number of epochs needed for considering training to be finished is characterised by converged performance metrics and loss. Learning rate specifies the size of the steps taken in optimisation, and learning rate decay enables a decay over each parameter update. Class weight represents the weighting of the loss function for both classes, indexed 0 and 1. Class weights were only used for pre-training as the dermoscopic data set was not balanced. The listed parameters also include batch sizes during training and testing of the network. Steps per epoch denote the number of batches retrieved for every epoch. Batch sizes and steps per epoch were chosen to ensure that the entire data set was retrieved in each epoch. A regularisation parameter was also determined for the different combinations of the L1 and L2 regulariser. This parameter was only set for fine-tuning as these regularisers were not applied in the pre-training. Table 4.1 presents all hyperparameters for both pre-training as well as fine-tuning.

4.1 Pre-Training on Dermoscopic Images

Additional pre-training using images of melanomas yielded slightly better results compared to initialising the weights with the network pre-trained on only ImageNet. To evaluate the effect of the initialisation, the network was fine-tuned using the ulcer images with the same set-up but changing the initialisation of the weights. Hence, two cases were compared; only pre-training the network on ImageNet and pre-training the network on both ImageNet and dermoscopic images. Both cases resulted in equal accuracy, and therefore, precision and recall were compared for the validation set. For comparison, the same values of precision and recall were

Table 4.1: Chosen hyperparameters for pre-training and fine-tuning, which are not altered during testing of different regularisers and levels of dropout.

	Pre-training	**Fine-tuning**
Epochs	50	50
Batch size, training	30	6
Batch size, validation	10	6
Batch size, testing	10	34
Learning rate	$1 \cdot 10^{-6}$	$1 \cdot 10^{-6}$
Learning rate decay	$1 \cdot 10^{-4}$	$1 \cdot 10^{-4}$
Class weight	1:4, 0:1	1:1, 0:1
Steps per epoch, training	50	42
Steps per epoch, validation	40	8
Steps per epoch, testing	1	1
Regularisation parameter	0	0.01

observed at the same epoch during training, after convergence. Precision was slightly higher when pre-training the network on both ImageNet and dermoscopic images while recall was higher for the weights only pre-trained on ImageNet. A higher precision was favoured after consulting Alexandra Forsgren due to the severity of false positives, making it more important that the ulcers classified as venous truly are venous. Thus, the decision was made to include the additional pre-training on skin lesions. However, to perform this pre-training was time-consuming, making it difficult to test and tweak hyperparameters and the number of epochs was therefore limited to 50 and a dropout of 20 % was used.

The results from the pre-training using dermoscopic images are presented in Appendix A. Applying a weight to the classes alters the loss function, which can be seen in an initially very high loss. From the results it is however visible that the loss is decreasing, and it is still decreasing when the last epoch finishes and none of the performance metrics has converged yet. This indicates that if the training would have been given more time to complete more epochs, better results could have been achieved.

4.2 Fine-Tuning on Ulcer Images

For the fine-tuning using the ulcer images, VGG19 was initialised with the weights from the network pre-trained on ImageNet and dermoscopic images. Experiments were performed for three cases of regularisers; L1, L2 and L1 combined with L2. For each test, three different dropout levels were compared; 10, 20 and 30 %. The hyperparameters set before training are summarised in Table 4.1. Different levels of the regularisation parameter α, see Section 2.2.4, were tested and it was finally set to 0.01 for all regularisers. In the following tests, the values of all performance metrics and the loss from the last epoch are documented. By adapting the number

of epochs based on the convergence of the performance metrics, the training was ended in time, and overfitting of the network was avoided. Since there were no indications of the performance changing drastically, the last epoch was considered representing the best result.

4.2.1 L1 Regulariser

The test results from applying a L1 regulariser with different dropout levels gave almost identical results. The loss, accuracy, precision and recall are listed in Table 4.2 for the training and validation set for all three dropout levels. When comparing the three cases, the best performance metrics for the validation set are obtained for the dropout levels 10 % and 20 %, but their losses differ. To determine the best set-up, the difference between training loss and validation loss was compared. A small gap between training and validation loss implies less overfitting and thus a better fit, compared to a greater gap. According to Table 4.2, the difference between the training and validation loss is the smallest with a dropout at 20 %. A dropout level of 20 % is therefore considered the best option for the case of L1 regularisation.

Table 4.2: Test results when training the network with L1 regulariser and different dropout levels. The values documented are from the last epoch.

Training				**Validation**			
	Level of dropout				**Level of dropout**		
	10 %	**20 %**	**30 %**		**10 %**	**20 %**	**30 %**
Loss	0.070	0.075	0.113	**Loss**	0.292	0.262	0.287
Accuracy	98.41 %	97.62 %	96.43 %	**Accuracy**	89.58 %	89.58 %	87.50 %
Precision	98.57 %	97.16 %	95.10 %	**Precision**	88.00 %	88.00 %	87.50 %
Recall	100 %	99.28 %	98.55 %	**Recall**	91.67 %	91.67 %	87.50 %

4.2.2 L2 Regulariser

Regulariser L2 was applied in the second test, with the same settings. The results are summarised in Table 4.3, presenting similar results when compared to those in Table 4.2. Equal percentages can be seen when observing the accuracy, precision and recall for validation in both tests. To determine the most promising set-up using the L2 regulariser, the gap between the training and validation loss is evaluated. The results show that L2 regulariser with dropout level 20 % has the best fit out of all the cases with the same accuracy, precision and recall.

Table 4.3: Test results when training the network with L2 regulariser and different dropout levels. The values documented are the values for the final epoch.

Training				Validation			
	Level of dropout				Level of dropout		
	10 %	**20 %**	**30 %**		**10 %**	**20 %**	**30 %**
Loss	0.084	0.109	0.095	**Loss**	0.274	0.263	0.277
Accuracy	97.22 %	96.83 %	96.03 %	**Accuracy**	89.58 %	89.58 %	87.50 %
Precision	98.57 %	97.14 %	97.14 %	**Precision**	88.00 %	88.00 %	87.50 %
Recall	100 %	98.55 %	98.55 %	**Recall**	91.67 %	91.67 %	87.50 %

4.2.3 L1 and L2 Regulariser

The third test was performed using a L1 and L2 regulariser with all the three dropout levels used in the previous tests, 10, 20 and 30 %. As seen in Table 4.4, the results are similar to each other, but also when compared to the other tests. Highest accuracy, precision and recall achieved are given with the dropout set to 20 % which was therefore considered as the most appropriate dropout level for this regulariser.

Table 4.4: Test results when training the network with L1 together with L2 regulariser and different dropout levels. The values documented are from the final epoch.

Training				Validation			
	Level of dropout				Level of dropout		
	10 %	**20 %**	**30 %**		**10 %**	**20 %**	**30 %**
Loss	0.088	0.103	0.097	**Loss**	0.281	0.272	0.305
Accuracy	96.83 %	96.83 %	97.22 %	**Accuracy**	87.50 %	89.58 %	87.50 %
Precision	97.86 %	95.83 %	95.71 %	**Precision**	87.50 %	88.00 %	87.50 %
Recall	99.28 %	100 %	97.10 %	**Recall**	87.50 %	91.67 %	87.50 %

4.2.4 Comparing Regularisers

Different levels of dropout did not affect the results considerably, but all three cases showed slightly better performance for a dropout level of 20 %. Using L2 as regulariser results in the smallest difference between training and validation loss, and is therefore considered the top performing network. In Figure 4.1, the losses for both training and validation data during training are shown, where the gap between the orange and blue graph is the smallest compared with the results from the other set-ups tested. The trends for the different performance metrics during training can be seen in Figure 4.2 and 4.3 below.

As mentioned, very small differences could be seen when comparing the results from all tests. Considering all values of accuracy on the validation set, a very limited range

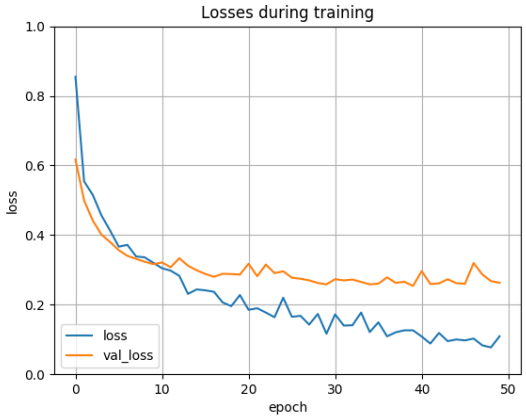


Figure 4.1: The loss versus epochs with L2 regulariser and dropout 20 %. The difference between the losses at the last epoch is approximately 0.15.

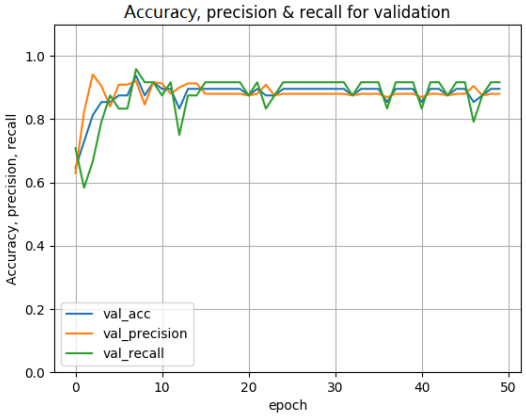


Figure 4.2: Shows the the performance metrics during training for the validation set with the best set-up from all tests, L2 regulariser with 20 % dropout level.

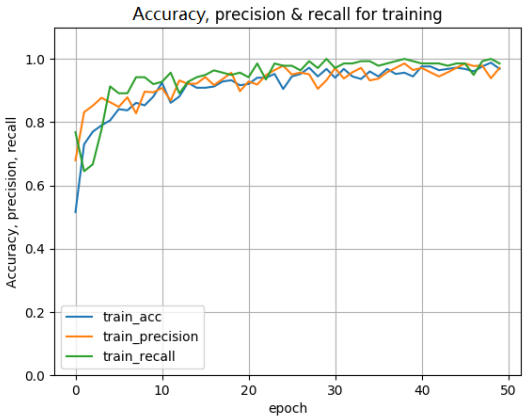


Figure 4.3: Illustrates the performance metrics during training for the training set with the best set-up from all tests, L2 regulariser with 20 % dropout level.

is seen and no network surpassed an accuracy of 89.58 %. This indicates the presence of a number of images that the network is never able to classify correctly, preventing a higher accuracy to be achieved. The incorrectly classified images were examined, and a number of frequently recurring images were identified. In Figure 4.4, the recurring incorrectly classified images of venous leg ulcers can be seen, while the non-venous can be seen in Figure 4.5. These images occurred for the majority of the networks, regardless of regulariser or level of dropout.

Figure 4.4: Recurring incorrectly classified images of venous leg ulcers.

Figure 4.5: Recurring incorrectly classified images of other ulcers, not venous leg ulcers.

Recurring correctly classified images, when comparing the classifications from different network set-ups, are displayed in Figure 4.6 and Figure 4.7. In Figure 4.6, the correctly classified venous leg ulcers are depicted and in Figure 4.7, other types of ulcers are shown. All images were classified with a high certainty, 95 % probability or greater.

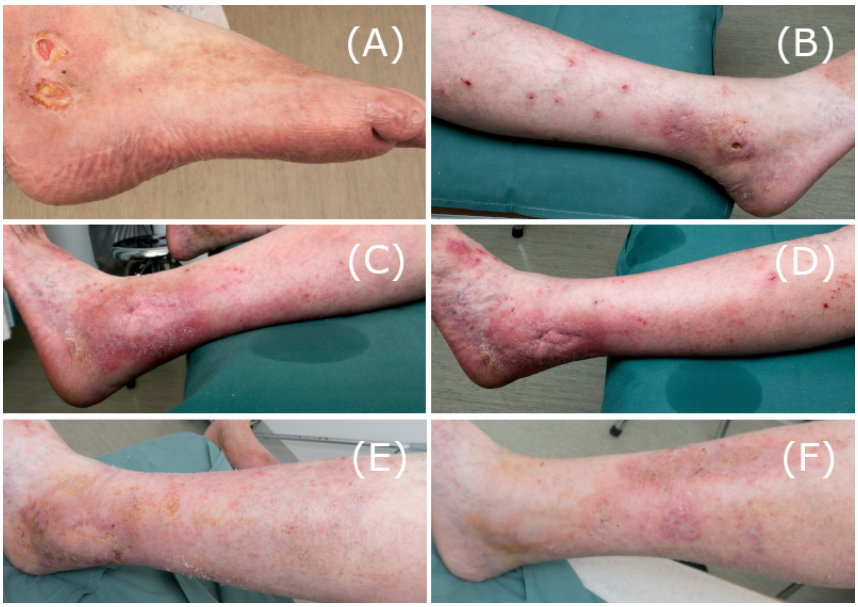


Figure 4.6: Recurring correctly classified images of venous leg ulcers, all classified with a probability equal to or greater than 95 %.

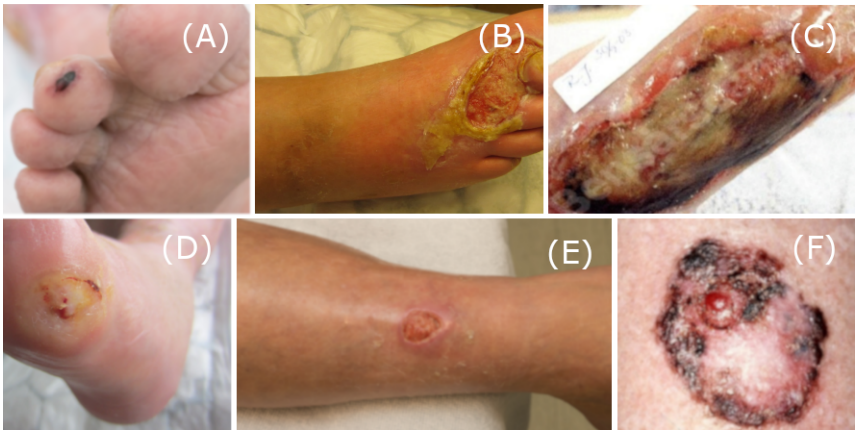


Figure 4.7: Recurring correctly classified images of other ulcers, not venous leg ulcers. All images have been classified with a probability equal to or greater than 95 %.

Correctly classified images, but with a lower certainty, were also observed. Recurring images for the different network set-ups were compared, finding images classified with a probability in the range 50 - 75 %. Even though still uncertain, a slightly better prediction was obtained for images taken from a larger distance. Examples of these images can be seen in the upper row in Figure 4.8, all depicting venous leg ulcers. The images in the bottom row depict the same ulcers taken from a closer distance, two of which were classified with higher degree of uncertainty and one being incorrectly classified. This indicates an increased uncertainty regarding images taken from a closer distance.

Figure 4.8: Images of venous leg ulcers, all but one classified correctly, but with low certainties. In the figure, two images of each wound are presented. The images taken at a further distance, (A) - (C), have been classified with a probability above 75 %. Two of the pictures of the corresponding ulcers taken from a closer distance, (D) and (E), have been classified with a lower probability of 50 - 75 %, whereas the third image, (F), was incorrectly classified.

4.2.5 Results for the Test Set

Lastly, the network's performance was tested using images it had never encountered before. By testing it on the test set, which was not included in the training, the network's true performance was evaluated. The test was performed using the final network, with a L2 regulariser and a dropout level of 20 % since this was considered the best set-up according to the analysis in Section 4.2.4. On the test set, consisting of 34 various ulcer images, the network achieved an accuracy of 85 %, a precision of 82 % and a recall of 75 %.

For comparison, a final test was performed with the set-up using a L2 regulariser and a dropout of 20 %. An important modification was made compared to the previous tests, moving all images in the validation set to the training set and thereby enabling training of the network using the entire data set. As a result of an increased training set, a larger number of epochs was needed for convergence of the performance metrics. Thus, training lasted for 100 epochs.

A slightly better performance could be seen for the training set when using the entire data set as training data, 300 images, than when being divided into a training set and a validation set. The values of loss, accuracy, precision and recall are summarised in Table 4.5. In this table, the results obtained when a validation set was included is shown as well, denoted as *Training with 252 images* for comparison. For a visualisation of the trends during the entire training without a validation data set, see the graphs in Figure 4.9 and Figure 4.10.

For comparison, the performance of the network was investigated using the same test set as before. Moving all images to the training set did not affect the results tremendously, and both tests gave identical results for the test set. For the test set, the network reached approximately 85 % accuracy, 82 % precision and 75 % recall as was obtained from the training including a validation set.

Table 4.5: Comparison between the results obtained from the training set in the final performance test of the network. Results shown both with and without the validation set, where the training set consists of 252 and 300 images respectively, for L2 regularisation and with a dropout level 20 %. All values documented are the values from the last epoch.

	Loss	**Accuracy**	**Precision**	**Recall**
Training with 300 images	0.060	98.00 %	98.17 %	98.77 %
Training with 252 images	0.109	96.83 %	97.14 %	98.55 %

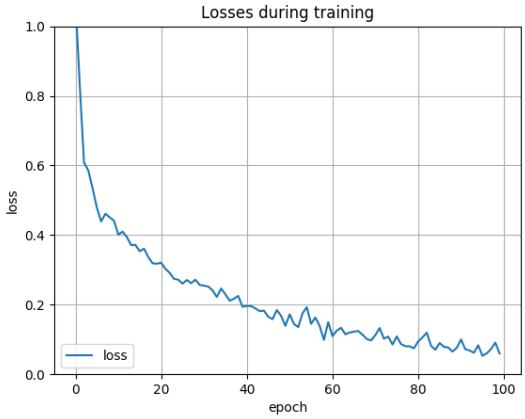


Figure 4.9: Illustrates the loss during the final evaluation of the networks performance with L2 regulariser with 20 % dropout level.

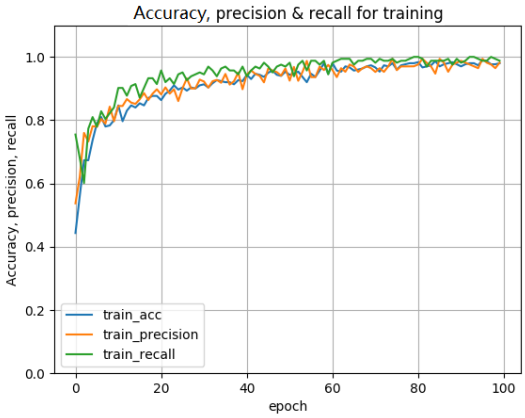


Figure 4.10: Illustrates the performance metrics during the final evaluation of the networks performance with L2 regulariser and 20 % dropout level.

Chapter 5

Concluding Discussion

The aim of this master thesis was to develop an algorithm for classifying images of ulcers, for the purpose of creating an aid for diagnosing ulcers. The proposed algorithm can in many aspects be considered successful, the results being proof of the solution's feasibility. However, a number of improvements could be made regarding the implementation and should be taken into consideration for further development.

5.1 Discussion Regarding the Method

The most important proposal regarding possible improvements to be made, having the potential of producing more reliable and accurate results, is increasing the size of the data set. The limited data set was the major concern during the project, undoubtedly affecting the results as larger data sets are typically applied for training CNNs. Possibly, a better performance could be achieved with a larger data set. In order to use medical images from novel patient data, a written application is required, that needs to be processed and approved. The master thesis had a limited time span and therefore, such an application was not possible within the scope of this project. Due to this reason, a large part of the data was from Alexandra Forssgren's previous research, already approved for these kinds of studies.

Another possible improvement concerns the pre-training of the network. Pre-training using images depicting different melanomas seemed somewhat promising. This can be seen in Appendix A, most notably by a decreasing loss for both validation and training data but also by an increased accuracy, precision and recall for training data. This indicated that if continued, the pre-training might have reached an even higher accuracy and a lower loss. These pre-training results could be improved by performing additional hyperparameter tuning. However, the time needed to finish the desired number of epochs was unreasonably long, partly due to the large amount of images, and additional tests were therefore not performed. If overcome, additional data sets could be used for pre-training, enabling the network to train on more data.

One possibility is to train the network on images with a higher degree of similarity when compared to the ulcer images, including other digital photographs.

One issue regarding the implementation of the network includes the drawbacks of Keras. The library has many useful tools and is simple to use, but the structure for building networks does not allow editing of individual building blocks in many cases. This was the reason for rewriting some of the functions and the classes they belong to, e.g. `ImageDataGenerator` and `DirectoryIterator`. Rewriting the tools already provided in Keras became the focus of the project for a longer time than should have been necessary. Without the need of rewriting the functions for augmentation, more time could be spent tuning the parameters for fine-tuning and especially, pre-training.

5.2 Discussion Regarding the Results

Despite the issues during implementation, promising results were obtained. However, it should be noted that the results were similar regardless of set-up concerning regulariser and level of dropout. When evaluating the correctly and incorrectly classified images, recurrent patterns can be seen.

For the incorrectly classified images, the network fails to classify the same five images, two of which were depicting venous leg ulcers and three of which were depicting other ulcers. Images from the validation set can be seen in Section 4.2.4, where these incorrectly classified ulcers are depicted in Figure 4.4 and Figure 4.5. When examining the training samples for the venous ulcers, these images were normally depicting ulcers from a distance. As close-up images of venous leg ulcers were more rare, this might explain the reason for the network not being able to classify Image (B) in Figure 4.4 as a venous leg ulcer. Furthermore, Image (A) and Image (B) in Figure 4.5 might be difficult for the network to classify as there is more of a resemblance to the images of venous leg ulcers. This can be seen considering both the actual ulcer but also in the composition of the image, as a larger part of the leg is visible which is normally seen in the training images of venous leg ulcers.

The discussed images can be compared with the images that were correctly classified, seen in Figure 4.6 and Figure 4.7. For most cases, the correctly classified images of venous leg ulcers depict the entire leg. For a limited number of ulcers, the validation set contained both images of the entire leg as well as close-ups. In these cases, the networks classified the images taken from a distance with a greater certainty compared to the close-ups. These images can be seen in Figure 4.8, where one of the close-ups was not classified correctly at all.

For future data collection, introducing additional variation in the images should be taken into consideration. Variation includes different lighting, angles and taking the images from different distances. This would enable the network to recognise the ulcers in a larger variety of settings which could potentially make the network's

predictions on images taken from a closer distance more certain.

5.3 Future Work

A number of improvements remain to be applied before implementing a deep learning based solution of this kind in the health care sector. The presented results are promising but as mentioned, increasing the data set could yield better performance for classifying venous leg ulcers. A larger data set also introduces the possibility of including more classes, enabling classification of other wounds as well. Including ulcers such as arterial leg ulcers, diabetic wounds etc., would thereby widen the area of use. Future developments also include adding additional, non-imaging parameters, normally evaluated when setting a diagnosis for an ulcer, listed in Table 1.1. This introduces the issue of having different types of input data, both images as well as tabular data, but if solved it could enable a full automated system for diagnosing ulcers to be developed.

Regardless of the level of complexity, with an automated classification aid the possibility of developing many kinds of easily accessible tools arises. The solution could be developed as a smartphone application, to be used within health care as a support system. Health care personnel could photograph a wound and receive a prediction after a few seconds. If developed with this target group in mind, the application could assist medical staff in their work and ease their workload. In the long run, this would be beneficial for patients as well who will be offered accurate treatment faster, reducing discomfort and improving life quality.

5.4 Final Thoughts

Considering the limited data set provided and the promising results obtained, a deep learning based solution for leg ulcer image classification should be considered plausible. The high values of precision, recall and accuracy for the validation set when comparing the different set-ups indicate the potential of a machine learning based solution. As a conclusion, the suggested solution is a good starting point for assisting health care personnel in ulcer diagnosis, potentially beneficial both when it comes to the financial aspect as well as the well-being of patients.

Bibliography

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning", *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] T. Hirasawa *et al.*, "Application of artificial intelligence using a convolutional neural network for detecting gastric cancer in endoscopic images", *Gastric Cancer*, pp. 1–8, 2018.

[3] O. Enqvist, *Lecture Notes in Image Analysis*, Lecture, pp. 47–55, 2017.

[4] F. J. Veredas, R. M. Luque-Baena, F. J. Martín-Santos, J. C. Morilla-Herrera, and L. Morente, "Wound image evaluation with machine learning", *Neurocomputing*, vol. 164, pp. 112–122, 2015.

[5] Västra Götalandsregionen. (Aug. 2017). About Region Västra Götaland, [Online]. Available: `http://www.vgregion.se/en/about/` (accessed 01/15/2018).

[6] E. Sthål, *Artificiell Intelligens för Diagnostik och Behandling av Svårläkta Sår*, Presentation, Jan. 2018.

[7] A. Forssgren and L.-M. Persson, "Venösa bensår – utred, behandla och förebygg recidiv", *BestPractice*, 2015.

[8] O. Nelzén, "Fifty percent reduction in venous ulcer prevalence is achievable - Swedish experience", *Journal of Vascular Surgery*, vol. 52, no. 5, 39S–44S, 2010.

[9] H. Brem, R. S. Kirsner, and V. Falanga, "Protocol for the successful treatment of venous ulcers", *The American Journal of Surgery*, vol. 188, no. 1, pp. 1–8, 2004.

[10] N. Cullum, E. Nelson, A. Fletcher, and T. Sheldon, "Compression for venous leg ulcers", *Cochrane Database Syst Rev*, vol. 2, no. CD000265, 2001.

[11] Västra Götalandsregionen. (Oct. 2017). Sårwebben, [Online]. Available: `http://www.vgregion.se/s/skaraborgs-sjukhus/vardgivare/sarwebben/bensar/` (accessed 01/24/2018).

[12] H. H. Aghdam, E. J. Heravi, and D. Puig, "A practical approach for detection and classification of traffic signs using convolutional neural networks", *Robotics and Autonomous systems*, vol. 84, pp. 97–112, 2016.

[13] D. Marmanis, M. Datcu, T. Esch, and U. Stilla, "Deep learning earth observation classification using ImageNet pretrained networks", *IEEE Geoscience and Remote Sensing Letters*, vol. 13, no. 1, pp. 105–109, 2016.

[14] X. W. Gao, R. Hui, and Z. Tian, "Classification of CT brain images based on deep learning networks", *Computer Methods and Programs in Biomedicine*, vol. 138, pp. 49–56, 2017.

[15] J. Ma, F. Wu, J. Zhu, D. Xu, and D. Kong, "A pre-trained convolutional neural network based method for thyroid nodule diagnosis", *Ultrasonics*, vol. 73, pp. 221–230, 2017.

[16] D. Meng, L. Zhang, G. Cao, W. Cao, G. Zhang, and B. Hu, "Liver fibrosis classification based on transfer learning and fcnet for ultrasound images", *IEEE Access*, vol. 5, pp. 5804–5810, 2017.

[17] A. Esteva *et al.*, "Dermatologist-level classification of skin cancer with deep neural networks", *Nature*, vol. 542, no. 7639, pp. 115–118, 2017.

[18] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision", in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2818–2826.

[19] S. Zahia, D. Sierra-Sosa, B. Garcia-Zapirain, and A. Elmaghraby, "Tissue classification and segmentation of pressure injuries using convolutional neural networks", English, *Computer Methods and Programs in Biomedicine*, vol. 159, pp. 51–58, 2018.

[20] Wound Care Solutions Telemedicine. (2018). Welcome to wound care solutions, [Online]. Available: `https://www.wound-doc.co.uk/` (accessed 05/18/2018).

[21] digitalMedLab. (2018). Mobile enhanced wound management, [Online]. Available: `https://wounddesk.com/` (accessed 05/18/2018).

[22] GNOSCO. (2016). Svårläkta sår, [Online]. Available: `http://www.dermicus.com/products/sar/` (accessed 05/18/2018).

[23] Stanford Vision Lab, *ImageNet*, Database of different images and synsets indexed, `http://www.image-net.org/`, 2016.

[24] H. Brodda, "AI ska hitta melanom", *MedTech Magazine*, 2018. [Online]. Available: `https://www.medtechmagazine.se/article/view/581593/ai_ska_hitta_melanom` (accessed 06/01/2018).

[25] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. Cambridge, MA: MIT Press, 2016, pp. 1, 96–179, 197–198, 221–251, 266–304, 286–309, 314, 321–333, 411.

[26] K. L. Priddy and P. E. Keller, *Artificial neural networks: An introduction*. Bellingham, Wash: SPIE, 2005, vol. 68.

[27] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition", *CoRR*, vol. abs/1409.1556, 2014.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization", *CoRR*, vol. abs/1412.6980, 2014.

[29] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database", IEEE, 2009, pp. 248–255.

[30] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks", *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[31] C. Szegedy *et al.*, "Going deeper with convolutions", *CoRR*, vol. abs/1409.4842, 2014.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition", *CoRR*, vol. abs/1512.03385, 2015.

[33] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu, R. T. Hurst, C. B. Kendall, M. B. Gotway, and J. Liang, "Convolutional neural networks for medical image analysis: Full training or fine tuning?", *IEEE transactions on Medical Imaging*, vol. 35, no. 5, pp. 1299–1312, 2016.

[34] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?", *CoRR*, vol. abs/1411.1792, 2014.

[35] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding", *arXiv preprint arXiv:1408.5093*, 2014.

[36] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning.", *CoRR*, vol. abs/1605.08695, 2016.

[37] F. Chollet *et al.*, *Keras*, `https://keras.io`, 2015.

[38] The International Society for Digital Imaging of the Skin (ISDIS). (2018). Welcome to ISDIS, [Online]. Available: `https://isdis.net/isic-project/` (accessed 05/22/2018).

[39] E. Tarpila *et al.* (2009). Svårläkta sår, vårdprogram för sydöstra sjukvårdsregionen, [Online]. Available: `https://vardgivarwebb.regionostergotland.se/pages/68015/Sv\%C3\%A5rl\%C3\%A4kta\%20s\%C3\%A5r\%202009\%202.pdf` (accessed 02/16/2018).

[40] NVIDIA. (2018). NVIDIA DIGITS, [Online]. Available: `https://developer.nvidia.com/digits` (accessed 06/05/2018).

[41] A. Maity, A. Pattanaik, S. Sagnika, and S. Pani, "A comparative study on approaches to speckle noise reduction in images", Jan. 2015.

Appendix A

Appendix 1 - Graphs and Tables

A.1 Pre-Training Results

In addition to the results presented in Chapter 4, a more thorough description of the results from the pre-training are presented in this section. Values for loss, accuracy, precision and recall for the final epoch are documented in Table A.1. Figure A.1 illustrates the loss both for the training set as well as the validation set over all 50 epochs. The accuracy, precision and recall during training can be seen in Figure A.2 and Figure A.3, for validation and training, respectively.

Table A.1: The results from the pre-training on skin lesion images with a dropout level of 20 %. The values documented are the values from the last epoch.

Training		**Validation**	
Loss	0.712	**Loss**	0.597
Accuracy	80.20 %	**Accuracy**	72.25 %
Precision	49.13 %	**Precision**	38.63 %
Recall	88.54 %	**Recall**	62.96 %

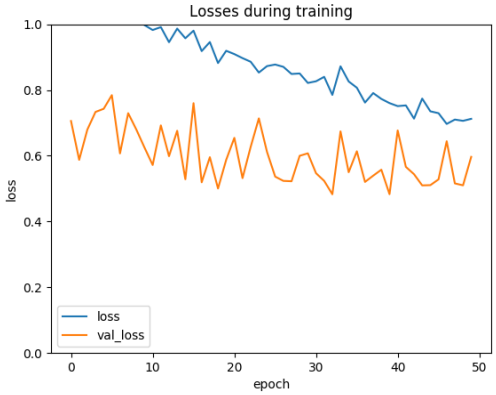


Figure A.1: The loss obtained over all epochs during pre-training. The loss starts at a high value due to the unbalanced data set. The training loss is clearly decreasing, while the validation loss is more difficult to interpret.

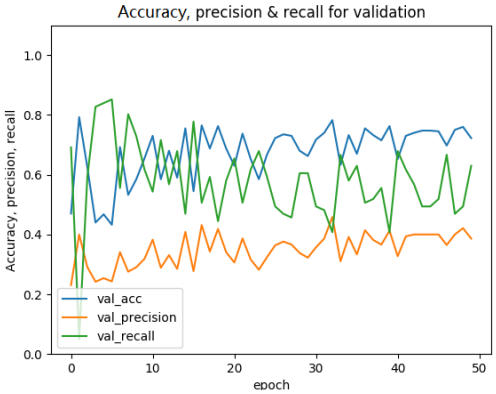


Figure A.2: Accuracy, precision and recall for the validation set. The graphs are noisy and do not converge, indicating a less successful learning.

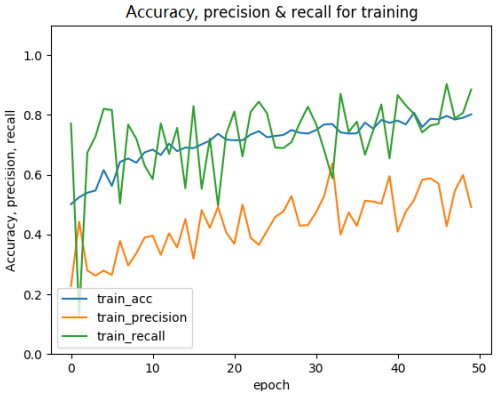


Figure A.3: Accuracy, precision and recall for the training set. All performance metrics are increasing over the shown epochs, and are still increasing when the final epoch is reached. This behaviour indicates that a further increase might have been seen, if given more time.

A.2 Fine-Tuning Results

This section presents the results from the fine-tuning. The results include values for loss, accuracy, precision and recall for all epochs during training, from the tests covered in Chapter 4.

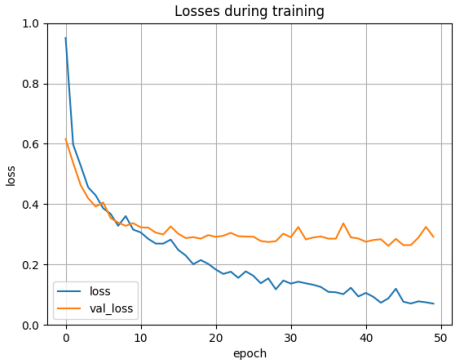


Figure A.4: The loss obtained over all epochs during training with a L1 regulariser and a dropout of 10 %. The difference between the losses for the final epoch is approximately 0.22.

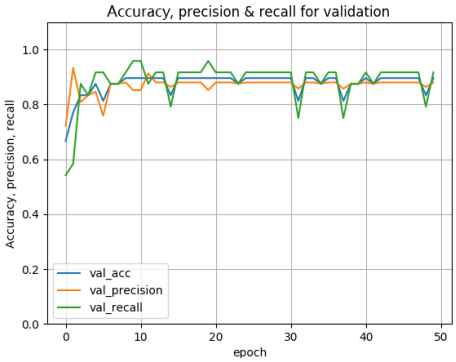


Figure A.5: Accuracy, precision and recall for the validation set using L1 regulariser and 10 % dropout.

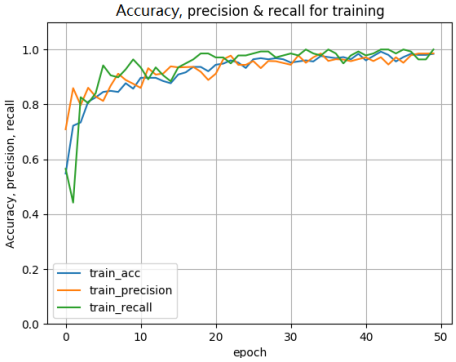


Figure A.6: Accuracy, precision and recall for the training set. L1 regulariser was used together with a dropout level of 10 %.

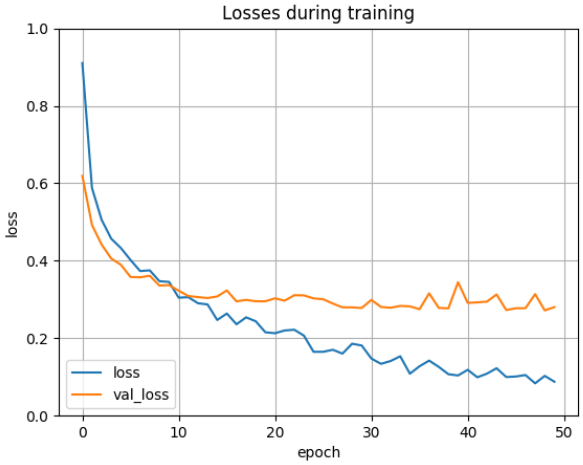


Figure A.7: The loss obtained over all epochs during training with a L1 regulariser and a dropout of 20 %. The difference between the losses at the last epoch is approximately 0.19.

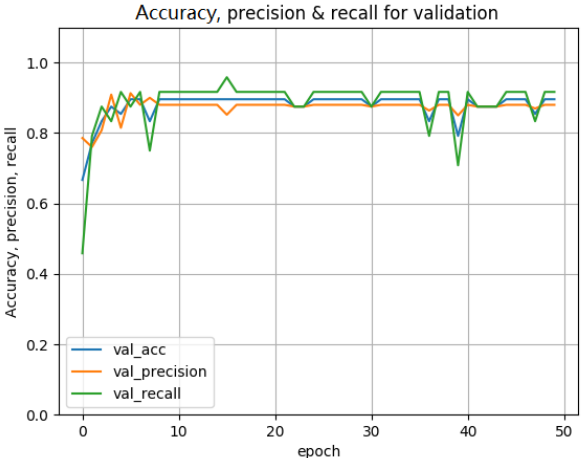


Figure A.8: Accuracy, precision and recall for the validation set using L1 regulariser and a dropout of 20 %.

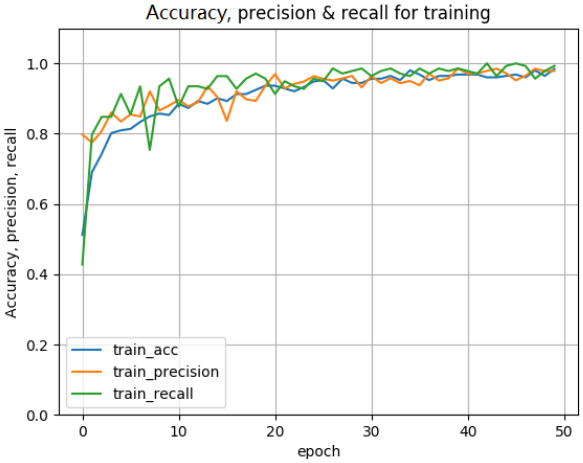


Figure A.9: Accuracy, precision and recall for the training set when training with a L1 regulariser and a dropout level of 20 %.

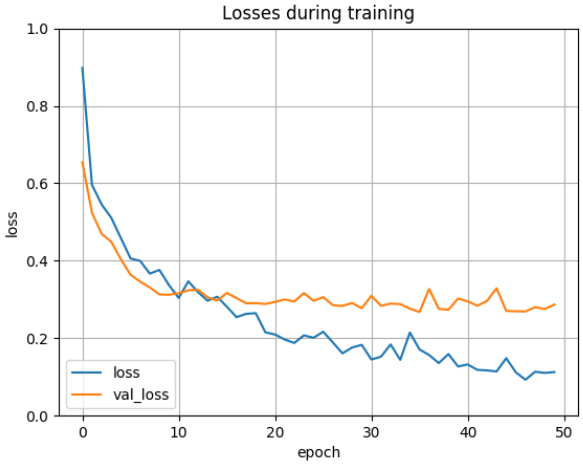


Figure A.10: The loss obtained over all epochs during training with a L1 regulariser, with a dropout of 30 %. The difference between the losses for the final epoch is approximately 0.17.

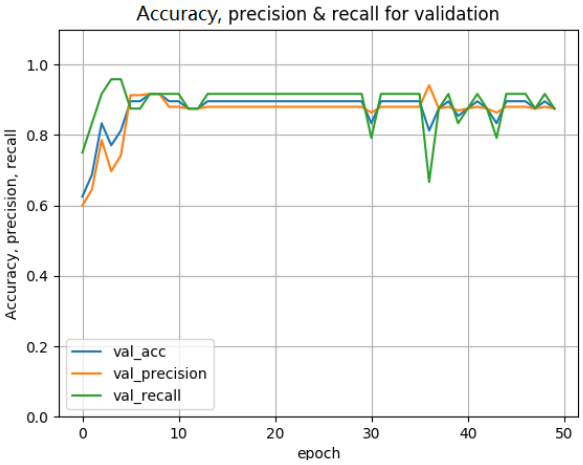


Figure A.11: Accuracy, precision and recall for the validation set. L1 regulariser was used with 30 % dropout.

Figure A.12: Accuracy, precision and recall for the training set, with L1 regulariser and a dropout level of 30 %.

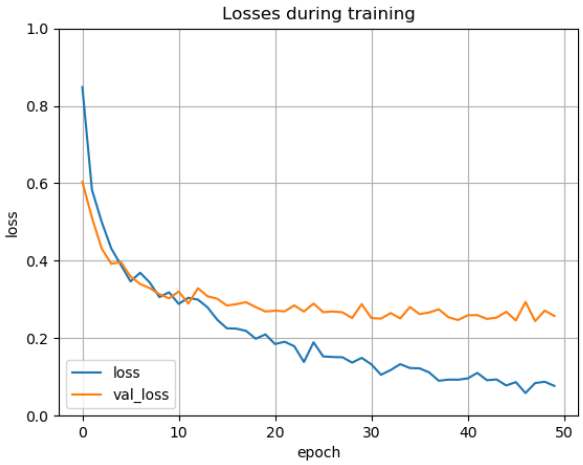


Figure A.13: The loss obtained over all epochs during training with a L2 regulariser and 10 % dropout. The difference between the losses at the last epoch is approximately 0.19.

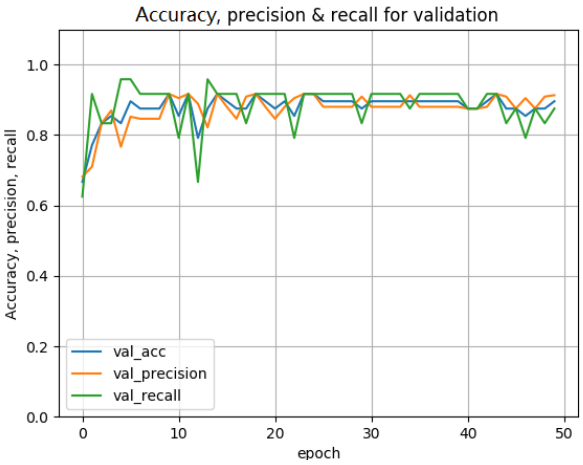


Figure A.14: Accuracy, precision and recall for the validation set using L2 regulariser and 10 % dropout.

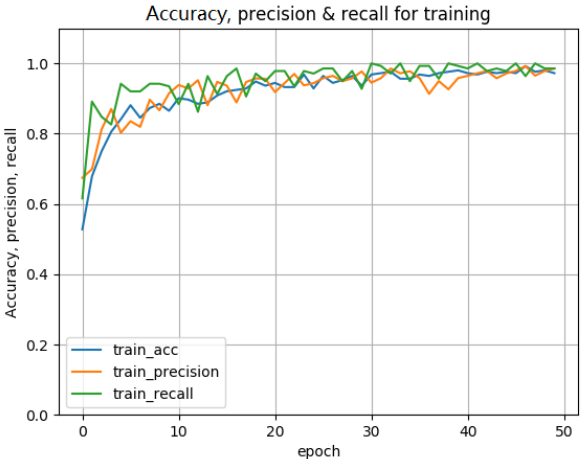


Figure A.15: Accuracy, precision and recall for the training set. L2 regulariser was used together with a dropout level of 10 %.

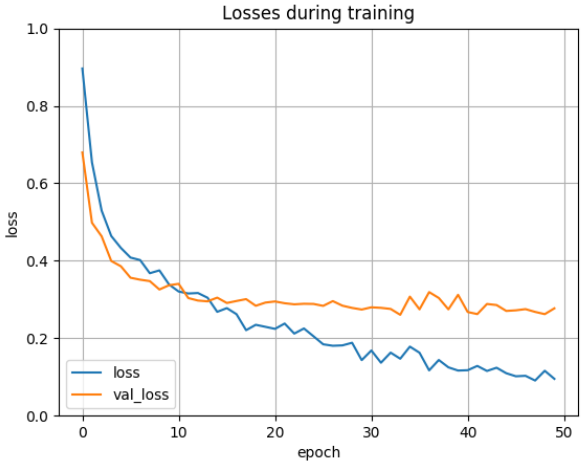


Figure A.16: The loss over all epochs during training with a L2 regulariser and a dropout of 30 %. The difference between the losses for the final epoch is approximately 0.18.

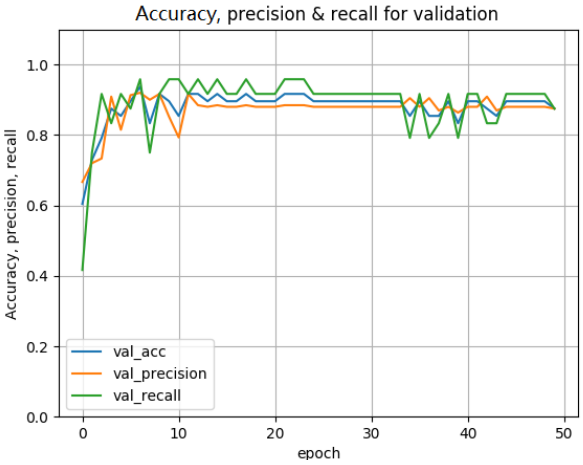


Figure A.17: Accuracy, precision and recall for the validation set when training with a L2 regulariser and 30 % dropout.

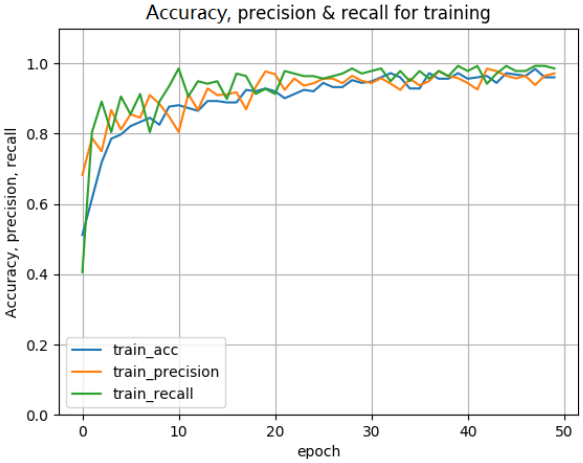


Figure A.18: Accuracy, precision and recall for the training set, using a L2 regulariser and a dropout level of 30 %.

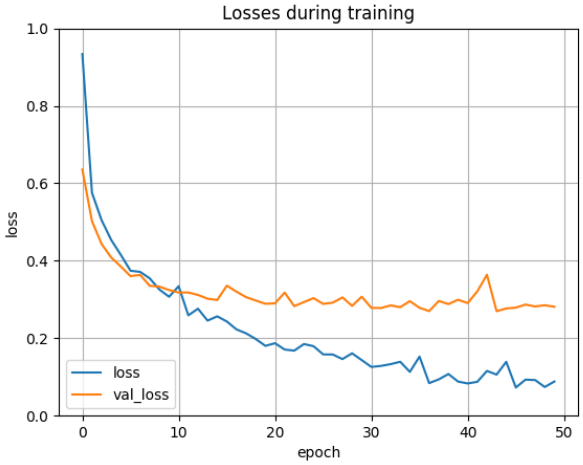


Figure A.19: The loss obtained over all epochs during training with a L1 together with a L2 regulariser, with 10 % dropout. The difference between the losses at the last epoch is approximately 0.19.

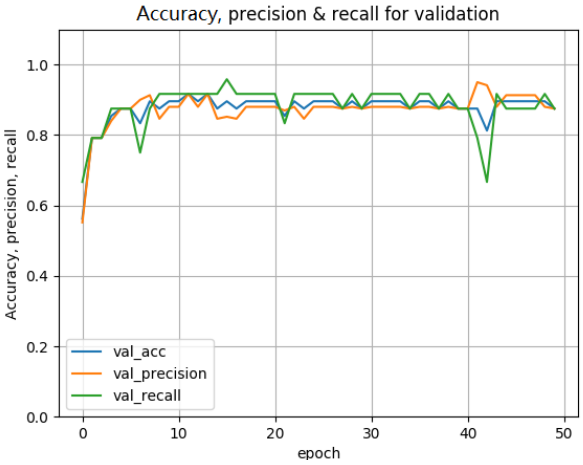


Figure A.20: Accuracy, precision and recall for the validation set, using L1 together with a L2 regulariser and 10 % dropout.

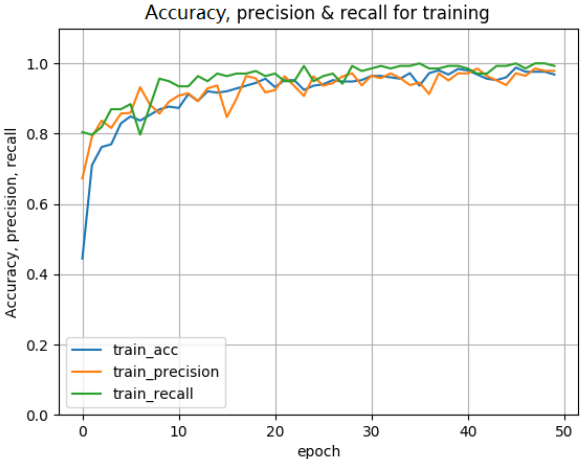


Figure A.21: Accuracy, precision and recall for the training set, using L1 together with a L2 regulariser and 10 % dropout.

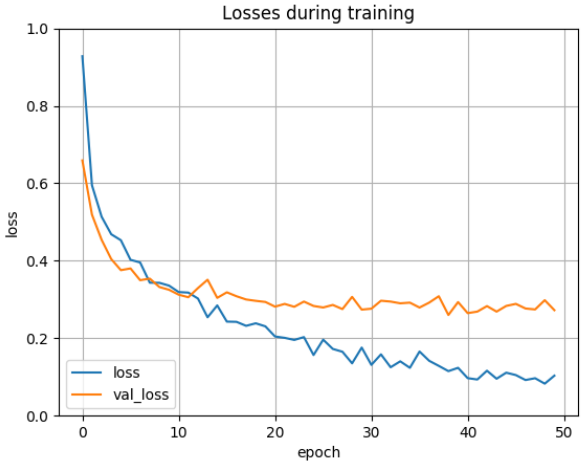


Figure A.22: The result obtained over all epochs during training with a L1 together with a L2 regulariser and a dropout of 20 %. The difference between the losses at the last epoch is approximately 0.17.

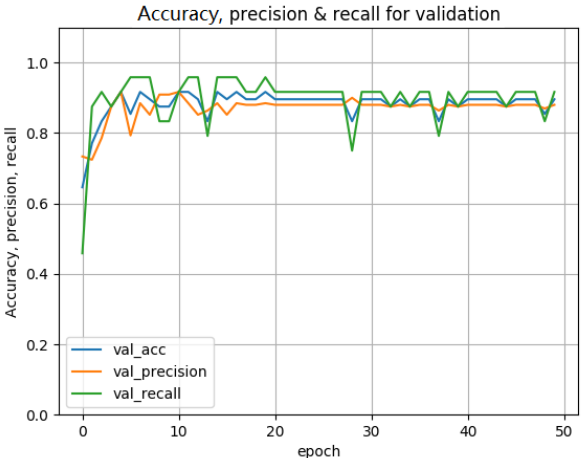


Figure A.23: Accuracy, precision and recall for the validation set. L1 together with L2 has been used as a regulariser, with a 20 % dropout.

Figure A.24: Accuracy, precision and recall for the training set, using L1 together with L2 and 20 % dropout.

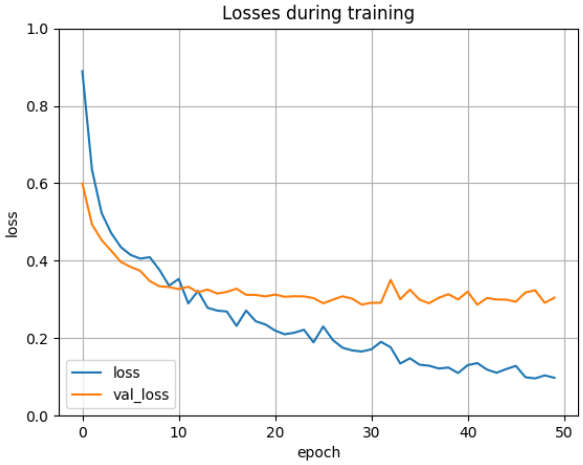


Figure A.25: The loss obtained over all epochs during training with a L1 together with a L2 regulariser and 30 % dropout. The difference between the losses at the last epoch is approximately 0.21.

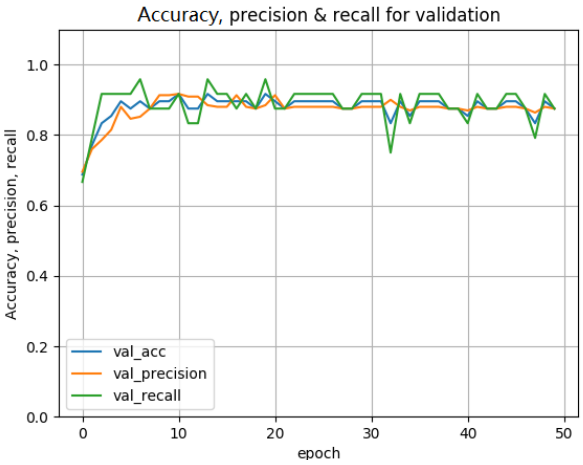


Figure A.26: Accuracy, precision and recall for the validation set. L1 together with L2 has been used as a regulariser, with a 30 % dropout.

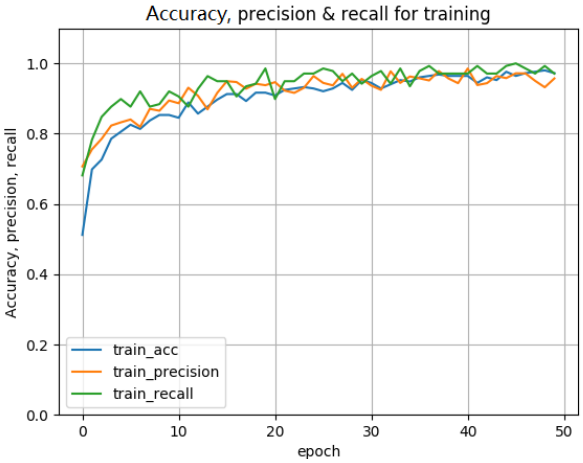


Figure A.27: Accuracy, precision and recall for the training set. L1 together with L2 has been used as a regulariser together with 30 % dropout.