CHALMERS

Detection of malignant melanomas using neural networks

Degree project report in Computer Science

Simon Länsberg
Anna Manfredsson

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY / UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022
www.chalmers.se

Degree project report 2022

Detection of malignant melanomas using neural networks

Simon Länsberg
Anna Manfredsson

CHALMERS

Department of Computer Science and Engineering
Chalmers University of Technology / University of Gothenburg
Gothenburg, Sweden 2022

Detection of malignant melanomas using neural networks
Simon Länsberg
Anna Manfredsson

 Supervisor: Björn Bergholm, Broccoli Engineering AB
Examiner: Lars Svensson, CSE Degree project report 2022
Department of Computer Science and Engineering
Chalmers University of Technology / University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2022

Detection of malignant melanomas using neural networks
Department of Computer Science and Engineering
Chalmers University of Technology / University of Gothenburg

Abstract

Approximately 60 000 people in Sweden are diagnosed with skin cancer each year, and around 500 of these patients die from their disease. There has been an increasing number of skin cancer cases in Sweden every year. This coupled with the highly stressed health care industry may result in a significant increase in mortality rates. Therefore, in an effort to detect possible malignant lesions on the skin, an image classification model was developed. The model in question was a convolutional neural network, a type of deep learning that specialises in classifying image data. In order to construct the dataset we used images found in the ISIC archives and divided them into two classes, malignant and benign. Several attempts were made before the best model was developed with a combination of transfer learning and the loss function ADAM. The model demonstrated an average performance of 73%. Using the Flutter framework it was possible to build an accompanying application with which the model could be presented to the general public. Ultimately, the app provided its users with the ability to take a picture of their lesion and then receive an indication based on the recommendation provided by the model. The connection between the application and the model was made possible through a Firebase database and a Python script that housed the model.

Keywords: AI, Melanoma, CNN, Transfer Learning, Python, Flutter, Firebase.

Acknowledgements

We would like to express our deep gratitude to to Björn Bergholm, Broccoli Engineering and all there employees for all their support and guidance throughout the project. We would also like to extend our thanks to our supervisor Firooz Shahriari, for his advice and assistance. Our grateful thanks are also extended to the other degree project students at Broccoli, for all of their support.

Glossary

ADAM Adaptive Moment Estimation. The loss function used for development of AI model.
AI Artificial Intelligence. In essence AI is simulated human intelligence in machines.
API Application Programming Interface. Software that enables intermediary communication between two applications.

CAD Computer Aided Diagnosis. Diagnosis assisted by computer related tools or algorithms.
CNN Convolutional Neural Network. The network of choice for identifying potential malignant lesions.
CSV Comma-separated values. Data file using comma between each value.

DCNN Deep Convolutional Neural Network. The more advanced, i.e. deeper version of CNN.
DOMMUDL Detection Of Malignant Melanomas Using Deep Learning. The name of the project.

IDE Integrated development environment. The environment were the code is developed.

JSON JavaScript Object Notation. A lightweight data interchange format. It uses human readable text to store or transmit data..

ML Machine Learning. A subset of AI.
MVSC Microsoft Visual Studio Code. The IDE tool for this project.

NN Neural Network. Discussed in section 4.1.

SDK Software Development Toolkit. The tools used to develop platforms and applications.

TL Transfer Learning. Discussed in section 4.2.

ZIP Zipped archive. A compressed file format used for file storage.

Contents

List of Figures

List of Tables

1
Introduction

The Swedish healthcare system is today very heavily burdened and the waiting times for specialist procedures like operations are sometimes longer than the national care guarantee set for health care in Sweden [2]. Regarding specialist procedures only 60 percentage of the patients received the care within the guaranteed 90 days during 2021 [3]. A possible way to aid in reducing these waiting times could be to, in an early stage, filter out people who search for healthcare when they, in fact, are healthy and just anxious about a hypothetical illness. These filters could help free up vital time and resources that could be better spent on those who need them. One area that is a potential candidate for such a filter is the assessment of potential malignant melanomas that are benign nevus.

In Sweden, around 60,000 people are diagnosed with skin cancer every year, and unfortunately, the number of cases increases each year. Of all those who are diagnosed, around 500 people die each year. The amount of skin cancer cases are exceptionally high compared to other countries [4]. In an attempt to shed some further light on a rising problem and hopefully start decreasing the stress on the Swedish health care system, an effort will be made to create an application that could classify the potential malignant melanomas.

Henceforth this project will examine the possibility of training a model for image analysis to distinguish between malignant melanoma and a benign nevus. Additionally, investigate if it is possible to develop an application where users can take a photograph on their smartphone and receive a recommendation based on the results. A potential benefit of this project could be to free time from the health care system by focusing on the areas mentioned above. Another effect of the project is to contribute to better patient health by reducing the stress and anxiety levels involved with potential malignant melanomas. Health care is a necessity for a sustainable society, and any attempts at increasing the availability of healthcare could help aid society in the long run.

Previous works in the field of malignant melanoma detection have been published and present a wide variety of methods to achieve the best-performing systems. Two publications, in particular, have served as a great source of inspiration for this project, *Fu'adah et al* [5], and *Naser-Esfahani et al* [6]. Both studies present similar approaches to classifying potential malignant lesions using Convolutional Neural Networks (CNN) and datasets built from the ISIC archives. Furthermore, the models were developed to act as Computer-Aided Diagnoses (CAD) to be used "in-house" by medical professionals. This differs from the proposed purpose of the project (Detection of Malignant Melanomas Using Deep Learning (DOMMUDL)), that instead will focus on providing the general public with a recommendation tool for poten-

tial malignant melanomas. The results found from these studies are primarily from *Fu'adah et al* [5], argues that the loss function Adam delivers the best performance and will therefore be the loss function for the developed models in this project. The studies attempted both Transfer Learning (TL) and creating a specialized Deep Convolutional Neural Network (DCNN) and seem to result in well-performing systems. Therefore DOMMUDLs developing phase will utilize both these methods.

Considering this project will deal with medicine and potentially deadly diseases, the goal is not to provide a complete diagnosis. Instead, a recommendation will be issued. This recommendation will be based on how much the skin leisure in question resembles the training material used. Furthermore, it needs to be stressed that malignant melanomas are not the only type of skin cancer. Skin cancer is a collective name for the uncontrolled growth of abnormal cells in the outermost layer of the skin. These growths are included but are not limited to melanomas, basal cell carcinomas, and squamous cell carcinomas. The current scope of this project only allows for the detection of melanomas. As of right now, only a few attempts of multi-class classification are planned to be performed during the developing phase.

1.1 Purpose and goals

With this project, the goal is to determine if, with the help of a photograph, it is possible to distinguish between malignant melanoma and a benign nevus and provide a recommendation with this information. In addition to this goal, it is desired to examine if it is possible to train a model for image analysis of melanomas. If so, is it possible to implement an application available for the general public?

1.2 Report Organization

To begin this report a description of the intended performance from an earlier stage will be presented. Followed by a presentation of the different technologies applied, such as the languages and platforms used. In addition, we will provide an explanation of what skin lesions are since this may be helpful when reading the following chapters. This project contains several complex concepts, most of which involve data and machine learning. To further clarify these concepts, we will dedicate a section to them. Detailed information on how the project was implemented and how the steps were completed are presented below. By demonstrating the what, why, and how of the project, we arrive at the result. Additionally, a section addressing ethical and sustainability aspects of this subject will be included. In order to conclude the study, we will discuss the results and explore further research possibilities.

2
Method

To achieve the goals for this project, the decision was made to split it into two main parts that are to be developed in parallel with each other. The two parts are the application and the Artificial Intelligence (AI) model. Due to previous experience, it was decided to work in an agile manner. To satisfy this method a scrum board will be set up using the online tool Trello, and online repositories will be used to house the code. The agile way of working was based on continuous feedback from the product owner which will be achieved by weekly meetings. Further, it was desired to achieve as much functionality as possible each sprint, which was made possible by working with all the different ”layers” of the project at the same time. To create a well suited AI model possible a Convolutional Neural Network (CNN) will be used. To guarantee that the model will be fine-tuned in a ”correct” manner extensive research into the architecture of CNNs will be conducted. This research will then be combined with several different tests to achieve the best performing model possible, and aid in answering the question of whether malignant melanomas in fact could be identified using a single photograph and a CNN-based classifier.

All collected data comes from medical sources and more data will not be collected since it is impossible in the current situation. It is not ethical to collect data that are not medically approved. Making data gathering an important part of the project to make sustainable recommendations.

To develop the application there were two good options of what language and framework to use. Mainly used for application development are React native and Flutter [7], due to previous experience in the Flutter framework it was chosen. There were discussions about how to develop the communication between the application and the machine learning model. The research of different database options concluded with the use of Firebase, which communicates well with Flutter and is easy to work with. Compared to other options, Firebase requires a low effort which is helpful in a project of this size.

To receive an early oversight of the project work, a weekly schedule has been made in the form of a GANTT schedule (See appendix B). This schedule is likely to receive small changes and fine-tuning during the project, but the basic structure is to remain during the entire process. The project work is planned to primarily be conducted at the Broccoli Engineering offices but due to the current situation brought on by the COVID-19 pandemic, efforts will be made to ensure the ability to work from home.

3
Technical Background

The following section will explain the concepts and tools used during this project. Firstly, the different programming languages and tools for software development will be presented, followed by the platforms utilized for the application and the script implementation. Machine learning is quite demanding, especially in regards to GPU(Graphics Processing Unit) memory. Therefore the GPU used for this project will be presented and discussed in this chapter. Since this project also handles several medical terms, these will be explained for a fundamental understanding to follow through the report.

3.1 Programming Languages

The different languages used to develop the application will briefly be introduced alongside their proposal for the project. Firstly, an introduction to the language dart, which will be used for all the code in the application. Followed by a presentation of Python, the primary language used for the script and machine learning model, along with the specific libraries utilized.

3.1.1 Dart

Dart is the foundation of Flutter [8]. Developing apps in Flutter entail writing code in the dart programming language. Dart was designed to work specifically well for client programming, with the goal to be the most efficient language for multi-cross platform development. It is specified for web and mobile development but could be used for many use-cases. In Dart, there are similarities with Java, C++, C# and JavaScript.

A typical dart code could look like this:

```
// Define a function.
void printInteger(int aNumber) {
  print('The number is \$aNumber.'); // Print to console.
}

// This is where the app starts executing.
void main() {
  var number = 42; // Declare and initialize a variable.
  printInteger(number); // Call a function.
}
```

Dart is type-safe, ensuring that the variable's value and static type always match, also called static type checking. Dart also offers the option to set the type of a variable to dynamic. These features and our experience leads to choosing Dart and Flutter.

3.1.2 Python

Every aspect of the machine learning model has been coded in python. Python was designed with code readability in mind which can be seen in both the language constructs used and its object-oriented approach. Unlike other popular languages like Java and C++, python uses significant indentation instead of the popular curly-brace approach. This helps the program's readability to be even more simplified and straightforward [9].

```
def count_Ones(arr):
    count = 0
    for n in arr:
        if n == 1:
            count+=1
    return count

a = [1,0,0,1,1,1,0,0,1]
print(count_Ones(a))
```

Output:

```
5
```

There are numerous libraries available for Python for any possible use case. It is common to use Python when working with machine learning. Consequently, many libraries have been created for this area, such as TensorFlow, Keras, Pandas, Pillow, and openCV2.

TensorFlow is an open-source library developed for machine learning, created to help ease the building and deployment of machine learning models, which are otherwise very complex. It can help with acquiring data, training, serving predictions, refining results, and more. Since the TensorFlow update 2.0, Keras API has been used for model training and is the version used in this project. TensorFlow uses dataflow graphs, a series of processing nodes, and describes how data moves through a graph. Each of these nodes is represented by a mathematical operation. The connection between the nodes each represents a multidimensional array, also called a tensor. TensorFlow builds the app frameworks in python, and the mathematical operations are executed in C++ binaries. Python only redirects the traffic between the operations and the nodes and tensors, leading to better performance. In summary, TensorFlow offers abstraction and solves all the background details of machine learning [10].

TensorFlow is suited for larger projects and complex workflows. The library is also easy to learn and pleasant to work with, which is, together with the other qualities mentioned, a primary reason for its usage in this project.

3.2 Platforms

For this project, two platforms are used. Firstly, the application is entirely developed in Flutter, a framework created by Google based on dart as the programming language. Secondly, Firebase will be presented as it is the database used.

3.2.1 Flutter

Flutter is a cross-platform framework, and there is one code base for developing multiple apps, supporting both iOS, android, and the web. Flutter is built from the code-language dart, Google's programing language introduced in October 2011. Flutter was launched in 2017, making it relatively new compared to other frameworks and languages for app development, like React Native from 2015, Ionic in 2013, and Xamarin in 2011. Flutter consists of an SDK (Software Development Kit) and a Framework, User Interface (UI) Library based on widgets. The widgets are ready to use, and by using them, the app can have a similar look to a native app. The widget library consists of sliders, buttons, and text inputs. It is also possible to create a new widget for a specific need. Considering Flutter is relatively new, there is a large community, and the framework is well documented [11].

3.2.2 Firebase

Firebase is a platform that helps build, improve and grow app development. Firebase is also referred to as a tool-set, providing the tools app developers do not want to have in focus. These tools are authentication, database, configuration, file storage, analytics, and many more. Firebase was developed by Google and is maintained fully by them. In other words, it is cloud-hosted [12]. Firebase provides a web console for managing all their services. With a Google account, a user can create a project and configure the desired app, either an android, an iOS or a web app. After configuration the user can enable the functionalities wanted for that specific project. For example Firebase offers authentication and if configured, it is possible to manage all accounts in the Firebase web console. For this project, we utilized two of Firebase's database services, Firebase Storage and a Real-Time database. As the name states, the real-time database updates in real-time. When time is crucial and when data requires constant updates, the real-time database is necessary. The Storage is a service for storing and loading files of different types and sizes. It is an ideal service when there is a need to store images. The Google Developers have created several instructions with examples in multiple code languages and a documentation that is easy to use, making it an easy task to get started in Firebase. Thanks to its user-friendly tools, Firebase is a good choice for this project where alternatives might need more time and work, which the scope of this project can not afford.

3.3 Graphics Cards

The training of machine learning algorithms and models is heavily reliant on the available hardware. In an attempt to reduce the overall dead time generated by the training of the AI model, a computer was specially designed for the project. One major issue that arose during the design and selection of parts for the computer was the ongoing global shortage of semiconductors. Semiconductors are used in several computer components, with the worst affected component being the GPUs. The shortage of semiconductors and the spike in interest in GPUs resulted in very few GPUs being available for purchase. The cards that were being sold had an increased price of sometimes up to 142 percent of the manufacturer's suggested retail price.

Table 3.1: Price data of current GPU-price Vs MSRP. Source Techspot [1].

GPU	MSRP	eBay Average Price December 2021	Inflation
GeForce RTX 3090	$1500	$2882	92%
GeForce RTX 3080 Ti	$1200	$1968	64%
GeForce RTX 3080	$700	$1584	142%
GeForce RTX 3070 Ti	$600	$1268	111%

Furthermore, the chief executive of Intel predicts that the shortage will last for "a year, or two" [13]. With these predictions combined with the availability of GPUs, the decision was made to procure the best-suited GPU available at the time of purchase (December 2021), and this card was the RTX 3080Ti.

When dealing with machine learning models, the memory of the GPUs is a crucial feature. The selected card has 12 GB of DDR6X memory combined with 320 tensor cores that excel at matrices equations frequently used during the development of machine learning models. It is worth noting that the RTX 3080Ti is not a pure workstation GPU since it is developed as a gaming GPU. The RTX A5000 is a workstation GPU with similar performance, but the needs for this project did not motivate the increased price of the A5000.

3.4 Skin Lesions

Skin lesions are widespread and can occur from birth. These are most often benign, "a mild type of character that does not threaten health or life" [14]. However, there are multiple types of malignant skin lesions, defined as "tending to produce death or deterioration" [15]. The malignant lesions, also labelled skin cancer, can be highly dangerous and cause death, which compels a thorough diagnosis of all skin lesions. A high medical discipline is required to distinguish between these skin conditions [16]. There are only a few malignant lesions and benign lesions we focus on in this project. The model will only detect and distinguish between the malignant melanoma, basal cell carcinoma and squamous cell carcinoma, as well as the benign nevus and dermatofibroma.

3.4.1 Malignant

Malignant skin lesions are dangerous and often life-threatening. There are three types of cells in the top layer of the skin, where skin cancer usually starts. They are basal cells, Squamous cells and Melanocytes. When old cells die, the basal cell's function produces new skin cells. During the formation of new cells, squamous cells are shed [17]. While there are many types of malignant skin lesions, those mentioned and covered in this report are melanoma, squamous cell carcinomas, and basal cell carcinomas.

Melanomas are the most dangerous and aggressive type of skin cancer. Melanomas develop in the melanocytes cells, which produce melanin. It is possible for an ordinary nevus to transform into melanoma, but more commonly the melanoma occurs from anywhere on the skin. Melanoma most typically appears on the breast, back or legs. It can be challenging to spot since it often develops from a regular non-dangerous skin lesion. The first signs of malignant melanomas are changes in an existing lesion or the increase of colour or unusual growth on the skin [18]. Its distinctive features are its irregular shape and non-consistent colouration, which usually shifts between brown, black, red, pink, blue and white. Additionally, it is often larger than 5 millimetres. If there are any signs of melanoma, it will be treated immediately because of the danger it applies, and if detected early, physicians can successfully treat it [19].

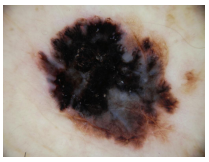


Figure 3.1: Example of malignant melanoma.

Another common malignant skin lesion is the Basal cell carcinomas. This type of skin cancer originates in the basal cells and is usually caused by exposure to sunlight. It is usually found on the head and neck since those areas receive much sunlight. Basal cell carcinomas can resemble a sore and have one of the following looks. Either a bump that looks shiny and transparent or a slightly raised lesion with brown, black or blue spots. There is also a possibility of a white lesion resembling a scar with wax or a flat patch that can grow very large over time [20].

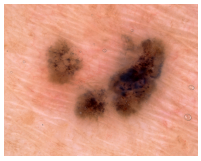


Figure 3.2: Example of malignant basal cell carcinoma.

Squamous cell carcinomas are the third and last type of skin cancer that will be covered in this report. They usually depend on excessive sunlight exposure and are the second most common skin cancer. Squamous cell carcinomas can appear

anywhere on the body where the squamous cells exist and tend to grow and spread rapidly. The most common places to find it is on the scalp, hands, ears and lips. This type of skin cancer is usually not as dangerous as melanoma or basal cell carcinomas. However, it can cause inconvenience when it grows or spreads [21]. Some sign of a potential squamous cell carcinoma is a lump or a spot with either the skin colour or a pale red. It can be covered by a firm scale and is occasionally sore from touch [22].

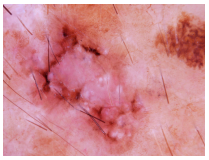


Figure 3.3: Example of malignant squamous cell carcinoma.

3.4.2 Benign

Benign skin lesions are harmless skin abnormalities. They appear often and can sometimes be mistaken for malignant lesions, which provokes worry even if they are non-dangerous marks. They also have the potential to become malignant and should be carefully surveyed. The similar appearances of malignant and benign marks cause difficulty distinguishing between them. Below is a short description of the two benign lesions: nevi and dermatofibroma [23].

One of the most common skin lesions is a benign nevus, also known as a mole or birthmark. A person can have around 10-40 nevi on their body, most of which appear at a later age. Nevi are usually harmless, but they can change in appearance and shape over time. On rare occasions, they can become malignant, which is why they need to be closely monitored for changes [23].

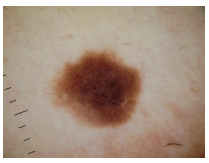


Figure 3.4: Example of benign nevi.

Dermatofibroma is a non-life-threatening skin lesion. It is firm and has a brown or brown-red colour. It often appears on the lower legs of women, though it can also be seen on the arms [24]. Although characterized as benign, dermatofibroma can be very painful. This skin lesion is often mistaken for malignant melanoma and even though it usually does not require treatment, it can be removed if there are symptoms or uncertainty about its benign nature [25].

Figure 3.5: Example of benign dermatofibroma.

4

Machine Learning Fundamentals

One of the core functions of this project stems in the classifying of potential melanomas. To achieve this, machine learning model was used. The name ”Machine Learning” is a straightforward explanation of what this technology does behind the scenes. Applying these methods can effectively give a computer abilities, including but not limited to seeing and identifying objects in real-time, understanding the meaning of a written text, and making predictions based on historical data. In essence, the ML model is given a collection of data related to the current field, a dataset. This data is then split into a training set, and a validation set used to train the model's abilities. In the current application, a classification task is implemented and trained.

4.1 Deep learning and Neural networks

Deep learning is a subset of machine learning that is particularly useful for developing this project. Deep learning can be viewed as an attempt to implement the learning method used by the human brain but is not close to matching the human brain's performance yet [26]. Networks consisting of several, or at bare minimum three layers are commonly used to enable deep learning. The layers in neural networks always contain one input, and one output layer combined with the ”deep” part of the network referring to the hidden layers found between them. The level of depth in the network corresponds directly to the sum of hidden layers in the model.

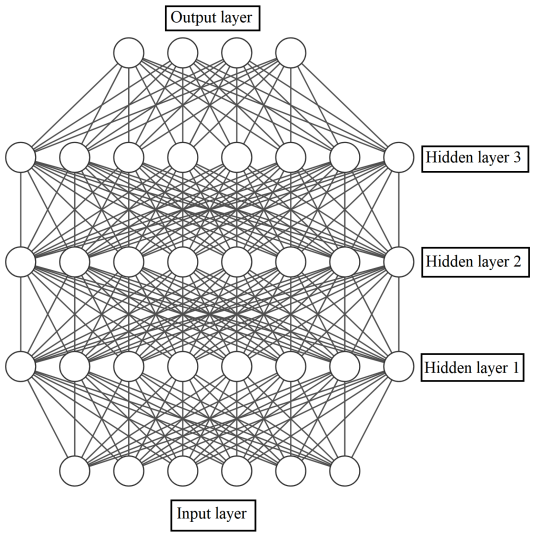


Figure 4.1: A basic sketch of a Neural Network.

As illustrated, the neural networks are built with several layers, each consisting of a set of nodes with individual connections. In the case of neural networks, these nodes can be considered individual linear regression models. These models are comprised of some input data, a weight, and lastly, either a threshold or a bias. [27]

Given these attributes, each node produces a formula like the one found by IBM [28].

$$\sum_{i=1}^{m} W_i X_i + \text{bias} \tag{4.1}$$

Where the output is given by:

$$\text{output} = f(x) = \begin{cases} 1 & \text{if } \sum_{i=1}^{m} W_i X_i + \text{bias} \geq 0 \\ 0 & \text{if } \sum_{i=1}^{m} W_i X_i + \text{bias} = 0 \end{cases} \tag{4.2}$$

When the data is passed through the different layers of the network, the neurons that have an output of one will be viewed as activated and therefore act as the input for the next layer of the network. The way that data passes between the layers of the model is referred to as feedforward, making the neural networks feedforward networks.

Neural networks exist in different configurations that excel at select tasks, like the Convolutional Neural Network (CNN), selected especially for this project because of its excellent ability to deal with image-based data. The CNNs have different types of ”layers” used when building models, some of the ones used for DOMMUDL are the following: convolutional layers, pooling layers, and dropout layers.

The Convolutional layers used during this project are of the type 2D Convolutional layers. These layers act as a sort of filtering layer. The layer takes an input in the form of an image and abstracts the contents to a feature or activation map. This abstraction is made by defining a filter of size n and then letting this filter slide across the image and summarize each step into a new pixel in the output. The

convolutional layers excel at dealing with image based task and are a crucial part of the convolutional neural networks.

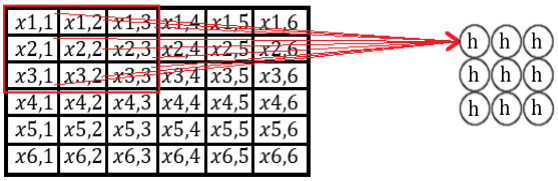


Figure 4.2: Visualization of a convolutional layer.

Pooling layers is a way to summarise the info from previous layers. In the layers used to create the model, the type of pooling layers used was MaxPooling. These layers contain a ”filter” like those used by convolutional layers, but instead of finding patterns, they find the highest valued pixel and transfer it to the next layer.

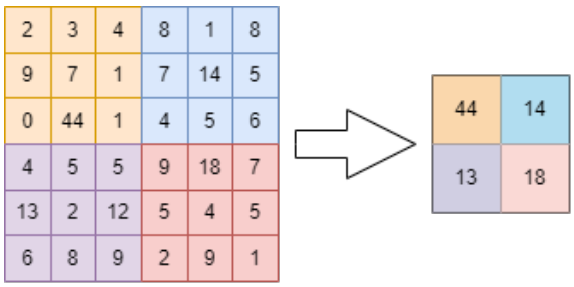


Figure 4.3: The way a 3x3 filter ”Maxpool” the highest value for the next layer.

Since the projects dataset will be a small one there will be a risk for overfitting when using a larger model, and therefore there is a need for a regularizer that can reduce this. This is where the dropout layers come in. These layers work by randomly ignoring, i.e., dropping a percentage of the nodes within the layers. This random risk of ignoring nodes makes so that the model no longer can rely on a single node when predicting, and must therefore spread the weights between the nodes.This helps to reduce the overfitting since a high weight associated to a single node is a sign of overfitting [29]. The frequency of dropped nodes is set when adding the layer to the model.

4.2 Transfer learning

Transfer learning (TL) is a way of ”piggyback” the success of a previous model and utilize it to improve another models performance. A more formal definition of TL [30] is: If there exists a source domain, a source learning task, a target domain, and a target learning task then the the goal of TL is to use the knowledge in the source domain and the the source learning task to improve the performance of the target domain and learning task. TL is a technique with increasing popularity and is frequently used when dealing with image classification tasks [31] [32] [33]. Since TL

is a option when only a small data amount is available [34], some attempts using TL will be planed for both the multi class and binary model attempts. The following sections will give a brief introduction to the different TL models included in the planned TL attempts.

4.2.1 VGG-16

VGG-16 is a pretrained deep convolutional neural network (DCNN) based on the sugestion of K. Simonyan and A. Zisserman 2015 paper [35], with the "16" part of the model names refers to the model depth being 16 layers. The model was trained on the ImageNet dataset consisting of 1281167 labeled images [36]. The VGG models were submitted to the ImageNet Large Scale Visual Recognition Challenge of 2014 were they displayed great performance and was the first runner up of this year. VGG-16 has been used for several TL image classification tasks [37] [38] [39], this was one of the contributing factors for selecting this model for the TL attempts.

4.2.2 Mobilenetv2

The mobilenetV2 is a Google developed DCNN originally designed for use in mobile devices, but is still commonly used for image classification tasks [40] [41]. Just like the previous VGG-16 model, mobileNetV2 has been pretrained on the ImageNet dataset, and is downloaded with these weights when used for TL. Unlike the VGG model, MobileNetsV2 is considerable deeper with 51 layers. This model would only be used for the binary TL attempts since it was added later to the project and the then current time frame would now allow it.

5
Data

Data is one of if not the most essential components when dealing with machine learning because if we do not have any information to learn from, it is impossible to create an AI model. Therefore, locating data well suited for this project was necessary before the work to find a functioning model could begin. To satisfy the needs of this project, the freely available database ISIC was selected. The International Skin Imaging Collaboration was created to facilitate digital skin imagining in an attempt to reduce the mortality of melanomas [42]. The ISIC archives contain over 60 000 images of different skin lesions ranging from malignant melanomas to benign nevi. From this archive, the photos for all the datasets will be fetched.

5.1 Data augmentation

When working with neural networks there always exists a risk of poor generalization. This is often a consequence of overfitting in the model. Overfitting means that the model has trained to fit exactly to the available data and therefore can not perform well on generalized test data. The opposite of this is referred to as underfitting and it occurs when your model have not trained enough to find the required patterns needed to predict accurately [43].

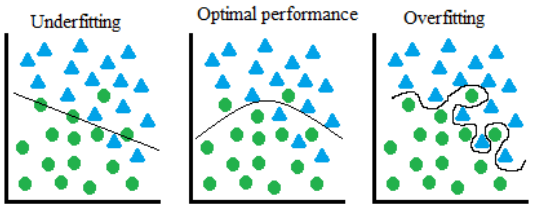


Figure 5.1: Illustration of overfitting, underfitting, optimal performance.

One way of minimising the overfitting of a model is to expand the dataset to create a greater variation of samples. But if the data is limited like in this case, one can instead utilize data augmentation. This method helps to expand the existing data by augmenting its contents. Some of the more common ways this is done include, flipping the images horizontally or vertically, shifting the color or contrast, rotating a certain amount of degrees, or cropping the images by applying a zoom factor [44] (See fig 5.2).

Figure 5.2: Data augmentation effects on one of the data images.

However, the usage of data augmentation could end up lowering the performance even further if not used properly. Per example if the augmentation includes rotating and the angle of rotation is low enough the augmented images wold become duplicates and therefore not contribute to the variation of data. This is illustrated in figure 5.3.

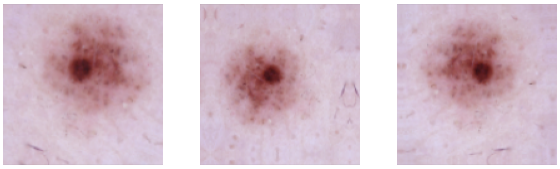


Figure 5.3: Creation of duplicates from augmentation.

5.2 The importance of data quality

When dealing with machine learning the quality of the model is not the only key factor for success. The quality of the data can, and will impact the final performance of the model and therefore some consideration is required when creating the datasets. Furthermore, it is crucial that the data used represents the models final use case, i.e., if a model for detecting Swedish road signs is developed one can not use a dataset consisting of American road signs even though they both are examples of road signs. The developed models intended use case was featuring cellphone taken images of the potential malignant lesions, therefore the data set needed to be constructed of cellphone taken images OR images taken in a similar way. Now the lack of data representing the final use resulted in the use of medically taken images instead, since they were the closest and only data available. Possible future attempts of recreation could most likely gain performance if the dataset instead was created with cellphone taken images instead.

When dealing with other models like, regression etc. it is good to filter out the outliers of the data since they can affect the final performance of the models [45]. For the dataset used in this project the outliers could be compared to the images containing other information than just the melanomas or nevi, like the images with ink shapes etc. These images were treated like outliers and therefore removed from the dataset. Other examples of images removed were the ones that not displayed clear lesions, i.e, lesions found on the patients scalps, or simply images of quality so poor that no distinct features could be identified. We will provide more information regarding our data and data processing in the next chapter.

6

Implementation

Following is a description of how the major part of the project was implemented. Observing the steps in order of their occurrence, the first process to notice is the installation of the hardware for the model. After the decisions were made and the order placed, the hardware needed to be installed. Once this was completed, both the application and AI model could be developed. The different steps involved in developing the application will be presented first, followed by the machine learning model and the steps needed to implement it. Further, a script has been developed to integrate these two parts, which will be discussed last.

6.1 Hardware Installation

Once all the components were delivered to the office the process could begin with the construction of the computer. The start of building any computer is to install the processor, followed by the ram, storage drives, and backplates required by the processor cooler. The storage drive used for this computer was an M.2 Solid State Drive or M.2 SSD. This type of hard drive requires no external power or cables and is simply installed right into an availed M.2 slot in the motherboard. Once all these components had been installed on the motherboard the next step was to install the motherboard itself onto the case. Once the motherboard was fitted correctly the cooler was installed and all the auxiliary cables from the case like power reset and USB ports were installed. Now the GPU could be installed with its special holder to avoid the weight of the card damaging the motherboard. To finalize the computer all components were supplied with electricity from the power supply and the installation and set up of drivers and operating systems could commence. In figure 6.1 the project computer is visualized.

Figure 6.1: The final product.

6.2 Software installations and tests

Once the computer setup was completed and all the essential software and tools were working as intended, the process of finding a suitable model could begin. Different models were created, tested, and finally evaluated during this task to meet the project's requirements. Other than the AI models, different supporting python scripts were developed to aid the data sorting and prepossessing. This section will present the development and deployment of the projects deep learning section.

6.3 Anaconda and Jupyter

As mentioned previously, only the essential software had been installed and configured. A program was needed to create the link between the GPU and the IDE used to research the different models. There are various ways to implement such functions, but a virtual environment with TensorFlow and Cuda was created for this project. The virtual environment was created using Anaconda's package manager through their Graphical User Interface, Conda Navigator. Other than the libraries required to utilize the GPU, several other essential libraries were installed onto the environment.

The IDE (Integrated Development Environment) of choice for trying out and developing the models was Microsoft Visual Studio Code (MVSC) combined with the Jupyter notebooks plugin. One of the main reasons for selecting this IDE and plugin was the simplicity of using the virtual environment combined with Jupyter and MVSC. No extra plugins or software was required, and the only thing needed was to select the right environment and then run the code.

6.4 The dataset

In the data section, the source of the images was introduced, and then it was time to create of the dataset using said images. ISIC archives provided a few methods to download the pictures. These included individually selecting the photos to use, downloading the data via the ISIC API, or simply downloading the entire database in one single ZIP file. These methods provided some pros and cons, but ultimately, downloading the whole database was selected since it would be the most time-efficient process. The entire database file totaled around 100GB and contained all the 69500 images available. The first data-related problem occurred, namely, the images were sorted by their provider and not the diagnosis or lesion type. Luckily a CSV (Comma Separated Values) file containing the metadata for each image was provided. Sorting these images manually would be an impossible task, and therefore a method of sorting these images was needed. To satisfy this need, a two-step sorting script was developed and used.

The two-step part of the script refers to the two main components that deal with the split of the metadata and the split of images between the different classes of lesions. The first subscript, ”CSV Sorter,” took the main CSV file containing all the metadata and selected rows based on two parameters given by the function call. These parameters were the diagnosis, type of lesion, and type of diagnosis. The reason for the second parameter is that some of the nevi and melanomas were not given a diagnosis, .i.e, malignant or benign. Therefore, the decision was made to exclude the images of melanomas or nevi that were not confirmed to be malignant or benign. The scope further supported this decision only to include data classed by doctors. Once the desired type of lesion had been specified, the script would pick out the rows, turn them into a separated data frame, and then convert the data into a CSV file using the library pandas. Once finalized, the script was run once for each of the five types of lesions found on ISIC and delivered the five new CSV files.

The next step was to split the images into the correct folders, a task solved by the second subscript named ”Image Sorter.” Instead of providing the main metadata CSV file, this script was given the newly created type-specific CSV files and a folder containing the image archive. Using pandas to read the CSV file and the OS library to only select the images for the specific class and remove the rest. The entire process of developing and testing the scripts took one day, and the sorting process could now be done in around 5 minutes, solving the task of sorting the archive data.

Table 6.1: Summary of data left for each class after first round of sorting.

Type	Images
Nevus	27869
Melanoma	5597
Dermatofibroma	245
Basal cell carcinoma	3395
Squamous cell carcinoma	655

A visual inspection of the collected data (Table 6.1) revealed that several images

displayed features other than the skin lesions desired by the data. The goal of the data was only to contain the lesion and skin without any additional noise, and to ensure that this goal was reached, a second round of sorting was required. A distinct unwanted feature appeared within the melanomas class, a vignette surrounding the ”important features”; therefore, these images needed some editing.

The main reason that keeping this vignette was not possible is that the data should be as similar to the use cases as possible. And since the usage is primarily intended for cellphone taken images, the vignette needed to be removed from the test data. An initial attempt to remove this data manually was made, but this quickly proved to be an extremely time-consuming task. Each image needed around 20-30 seconds to have its vignette removed, and the total amount images that needed editing totaled up to about 1600. If this task were to be performed manually, it would take at least 8 hours of continuous editing. Therefore it was decided to utilize the python libraries Pillow, CV2, and OS image editing tools to perform the edits instead. Removing the vignettes involved cropping out a certain percentage of the image edges. After some fine-tuning, it was found that the optimal rate to crop was 28 percent of the height and width form each of the images. This edit was made by calculating the new start and end pixels for both the height and width, and for this task, the following formula was used:

$$\text{StartPixel} = (\text{Image\_height} * 0.28) \tag{6.1}$$

$$\text{EndPixel} = (\text{Image\_width} - \text{StartPixel}) \tag{6.2}$$

Since all the images shared width to height ratio of 1:1 we could use the same variables for both height and width. Now that the start and end points were found it was just to crop and save the edited image.

```
crop_img = img[StartPixel:EndPixel, StartPixel:EndPixel] #Crop the image
cv2.imwrite((name+'.jpg'), crop_img) #Save the image
```

Once tested and optimized, the script completed the edit of all the images in 70 seconds, including the time for developing the script. This task was solved under two hours, proving to be the most time-efficient method of removing the vignettes (See figure 6.2).

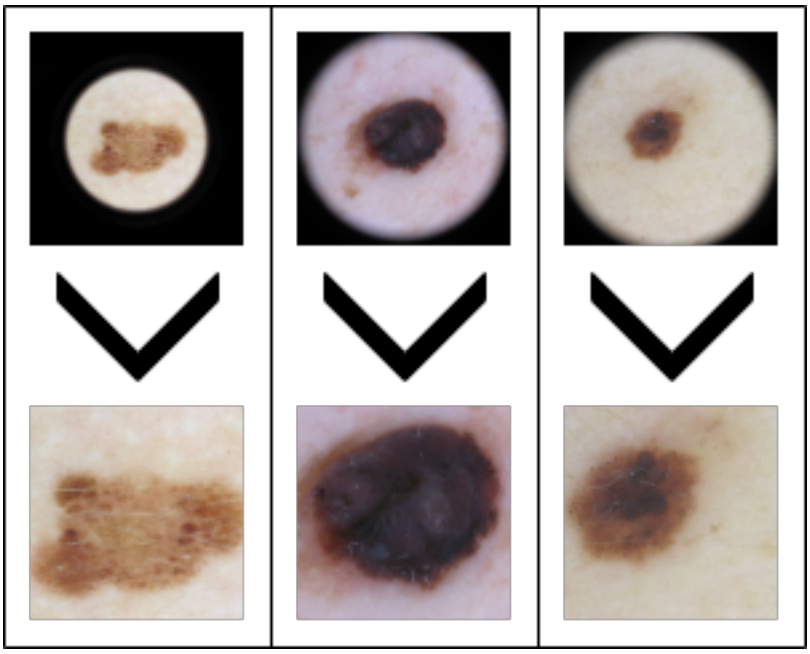


Figure 6.2: The result of removing vignettes using the developed script.

The vignettes were the largest but not the only noise found during the inspection. Some of the other most commonly occurring was the usage of pens to highlight or circle the potential melanomas or simply images too noisy to find the lesions. These images were discarded from the datasets, and the task was not big enough to require the development of additional scripts.

After completing the second round of data sorting, the images were saved in labeled folders, and a unique test set was created from around 10% of the data. This test set will be kept separate from all the other data to ensure that it is new data used to evaluate the model performance.

Table 6.2: The final data set.

Type	Images	Test set
Nevus	8000	800
Melanoma	5216	520
Dermatofibroma	243	24
Basal cell carcinoma	1698	170
Squamous cell carcinoma	371	37

6.5 The Multiclass model

Before the creation of a model could begin, a test model was set up and trained on the computer. This was done for two main reasons: to validate that both the virtual environment and the IDE did work as intended. The second is to get some experience working with a machine learning model. The guide in question was made by the authors who also created the TensorFlow library and included a step-by-step guide

on the setup and training of a small CNN intended for multi-class classification [46]. The guide's goal was to classify different flowers based on a TensorFlow-provided data set of 3700 flowers. Once the manual had been implemented and would run as intended, it was clear that the project's computer could complete training in half the time of google collab when using their GPU option.

When the basic CNN could run on the project computer, an attempt was made to use the same sequential model, but this time to detect the skin lesions from the procured dataset. This attempt aimed to see if the TensorFlow provided model would function on the more difficult task of predicting the correct lesion. If this attempt did not succeed, it would be regarded as a training round to get more familiarized with developing and training a machine learning model. The sequential model provided by the guide was built in a pattern that began with an input followed by three rounds of convolutional layer and max-pooling. Lastly, the model would flatten the outputs and follow up with one fully connected layer and the output layer.

Table 6.3: The Model.

Layer (type)	Output Shape	Parameter
Input Image	128,128,3	0
Rescaling	128, 128, 3	0
Conv2D	128, 128, 16	448
MaxPooling2D	64, 64, 16	0
Conv2D	64, 64, 32	4640
MaxPooling2D	32, 32, 32	0
Conv2D	32, 32, 64	18496
MaxPooling2D	16, 16, 64	0
Flatten	16384	0
Dense	128	2097280
Dense	5	645

When the model was ready to be trained, the dataset had to be converted to a format that the model could utilize, and this data had to be annotated since this is not an unsupervised task. This converting was done by using the *image_dataset_from_directory* function from the Keras utility library provided by TensorFlow:

```
    ds_train = train_datagen.flow_from_directory(
    'data/',
    target_size = (128, 128),
    validation_split=0.25,
    subset=''training'',
    seed=123,
    batch_size = 32
)
```

In this case, the *'data/'* image data was placed in a separate folder for each class

of lesions within the given directory. The dataset *ds_train* would consist of five classes and 80% of the available data. And the remaining 20% would be used to create the validation set:

```
ds_val = train_datagen.flow_from_directory(
    'data/',
    target_size = (128, 128),
    validation_split=0.25,
    subset=''validation'',
    seed=123,
    batch_size = 32
)
```

Once completed, these codes provided the finalized dataset containing five classes and a 75/25 training/validation split. The data is now divided into batches of 32, and all the images have been re-scaled to have the height-width ratio 1:1 with the measurements 128x128 pixels. Now that the dataset and model both were prepared, the model training would begin, yet a new issue was observed when inspecting the training times. From performing well compared to the times achieved by Google Colab to taking around ten times the time per epoch. This proved to be an issue concerning the selected virtual environment and was resolved relatively quickly.

The training was initially run for 50 epochs, and once completed, the performance was visualized in a plot displaying the training accuracy versus the validation accuracy. Further, this model was tested on the previously mentioned test set. The issue found during the evaluation was that the model had a decent performance on the training with around 93% but closer to 30% once evaluated on the test set. As discussed in the data section data augmentation, a poor generalizing of the test set could be a side effect of overfitting. Therefore it was decided to apply some data augmentation to increase the size of the test data. The augmentation methods added included rotation, horizontal flipping, and zoom. These augmentations were added as a prepossessing step for the sequential model, and the training process was run once again. And this time, it was run for an additional ten epochs to increase the total to 60 epochs. The evaluated performance of both models is illustrated in figure 6.3

Figure 6.3: Initial training (Left) versus the augmented training set (Right).

A minor improvement was not detected in the augmented test set, but the generalization was again poor when evaluating the test set, still around 30%. At this time, each training round took around 2.5 hours, the remaining time of the day was spent tweaking the augmentation and retraining the model. After several attempts, no improvements were made, and therefore the decision to try a total of one hundred epochs was taken. With these epochs, a training time of around 5 hours was expected. The training went on without any issues, and the training accuracy reached an impressive 96% (See fig 6.4). Yet, once the evaluation of the test set was made, there was no real improvement in the model's generalization for new data.

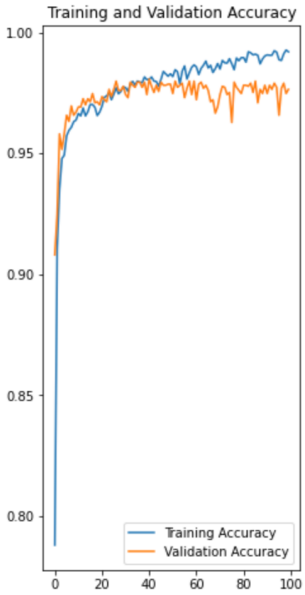


Figure 6.4: Performance of TensorFlow model with 100 epochs and data augmentation.

Two additional attempts were made for the 100 epochs trying to alter the data augmentation, but this only led to a decrease in the already low performance of the test set. Instead of further tweaking the augmentation of the training data now, an extra layer would be added to the model. After the convolutions part, a dropout layer was added to improve the performance further.

Table 6.4: The The model with added dropout layer.

Layer (type)	**Output Shape**	**Parameter**
Input Image	128,128,3	0
Rescaling	128, 128, 3	0
Conv2D	128, 128, 16	448
MaxPooling2D	64, 64, 16	0
Conv2D	64, 64, 32	4640
MaxPooling2D	32, 32, 32	0
Conv2D	32, 32, 64	18496
MaxPooling2D	16, 16, 64	0
Flatten	16384	0
Dropout	**16, 16, 64**	**0**
Dense	128	2097280
Dense	5	645

The model featuring dropout has been tested a total of 5 times, the first three tests tested different dropout factors (0.2, 0.4, 0.6, 0.8), and the fifth time used the ”best” performing model combined with augmentation. Neither of these models could improve the results of the test sets, so the next step was to explore the options

of using a pretrained model, so-called transfer learning to improve the performance hopefully.

The chosen model for this task was the VGG-16 model (See fig 6.5), and only a few tweaks had to be made to the available training data. The main feature that required changing was the resolution of the data. Instead of the previous 128x128, the VGG-16 instead required a 224x224 resolution. The data augmentation was kept the same for this during these tests.

Table 6.5: The VGG-16 model with added top layer.

Layer (type)	**Output Shape**	**Parameter**
InputLayer	224, 224, 3	0
Conv2D	224, 224, 64	1792
Conv2D	224, 224, 64	36928
MaxPooling2D	112, 112, 64	0
Conv2D	112, 112, 128	73856
Conv2D	112, 112, 128	147584
MaxPooling2D	56, 56, 128	0
Conv2D	56, 56, 256	295168
Conv2D	56, 56, 256	590080
Conv2D	56, 56, 256	590080
MaxPooling2D	28, 28, 256	0
Conv2D	28, 28, 512	1180160
Conv2D	28, 28, 512	2359808
Conv2D	28, 28, 512	2359808
MaxPooling2D	14, 14, 512	0
Conv2D	14, 14, 512	2359808
Conv2D	14, 14, 512	2359808
Conv2D	14, 14, 512	2359808
MaxPooling2D	7, 7, 512	0
Flatten	25088	0
Dense	5	50178

Preparing a TL model requires a slightly different approach than the previous sequential models. The first step is to download the model without including the top part since this part will be added specifically for this project. Once downloaded with the specification, *Include top = false*, and *Image weights = 'Imagenets'*, these layers were ”locked” to ensure that they retain their pretrained values. Now the top layers could be added. They consisted of a flattened layer to reshape the outputs, followed by a fully connected layer and a final output layer shaped for the different classes of skin lesions. And with that, the transfer learning was ready to commence. The first test was only run on ten epochs to detect possible errors in the new model, but no errors were detectable from the recorded performance. No further evaluation was done after this initial test. Instead, a new training session was initiated for one hundred epochs. Upon completion of this session, the model was evaluated in figure 6.5 only to present little to no improvements.

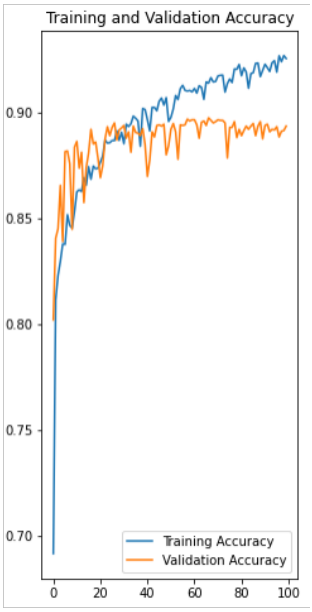


Figure 6.5: Performance of VGG-16 model with 100 epochs and data augmentation.

Since no significant improvement had been made with the multi-class models, the choice was made to no longer pursue the idea of a multi-class model. and instead shift the focus of the last project weeks to find a binary model that could differentiate between benign and malignant lesions instead.

6.6 Binary

Once the focus was shifted to a binary model instead, a few alterations had to be made, with the main changes being to the dataset. The previous data included all the skin lesions separated by class, which did not fit the requirements for binary, i.e., healthy or cancer. The main discussion regarding this change was how the new dataset should be designed, should it ”just” be sorted, and have all malignant data in one class and the benign in the other. Or completely disregard every class but melanomas and nevus. The issue with including all the types of lesions was that not all the lesions had equally serious health effects. Despite the different lesion’s health effects, they all shared the same treatment protocol of being surgically removed to ensure that they would not spread or metastasize to other parts of the body. With this in mind, the decision was to build the malignant data set with all the lesions whose treatment included surgical removal. While the benign data set would be made from the nevi. Unlike the previous dataset, this data set is balanced, meaning that every class was of the same volume. The construction of the binary dataset was performed similarly to the previous one using the same method discussed in section 6.5. The one difference was that the dataset creating method was set to make a binary dataset instead of a categorical dataset. And just like when dealing with the multiclass dataset, there was a reserved test set for the data to ensure that the evaluation was made with new data.

In regards to the model, the same structure was used as the one discussed in the introduction [5]. With the main difference now being that the final output layer only contained one node, i.e., malignant or benign. With these minor tweaks, the models were now ready to be trained on our new binary structured dataset.

Table 6.6: The binary model.

Layer (type)	**Output Shape**	**Parameter**
Input Image	128,128,3	0
Rescaling	128, 128, 3	0
Conv2D	128, 128, 16	448
MaxPooling2D	64, 64, 16	0
Conv2D	64, 64, 32	4640
MaxPooling2D	32, 32, 32	0
Conv2D	32, 32, 64	18496
MaxPooling2D	16, 16, 64	0
Flatten	16384	0
Dropout	16, 16, 64	0
Dense	128	2097280
Dense	1	645

The first round of training went without any noticeable incidents during the training, and a total of 100 epochs were run. Yet some issues arose when the evaluation was to commence. Firstly, the plot of the model accuracy appeared out of order. Once it was time to evaluate the data on the test set, the model output only gave the same prediction no matter the image provided. The initial idea was that something was missing from the dataset or that the compilation of the dataset had somehow failed. This proved to not be the case after re-compiling the data and performing the training again since the result remained. Upon closer investigation, it was instead revealed that the model's output layer was to blame. The final fully connected layer was added with the parameter "*numClasses*," just like the previous models. But, since swapping from multiclass to binary, this variable had not been tweaked and was still set to 5. Once this was changed to the correct value of 1, the model was again trained, and this time the graph appeared more as expected (See fig 6.6).

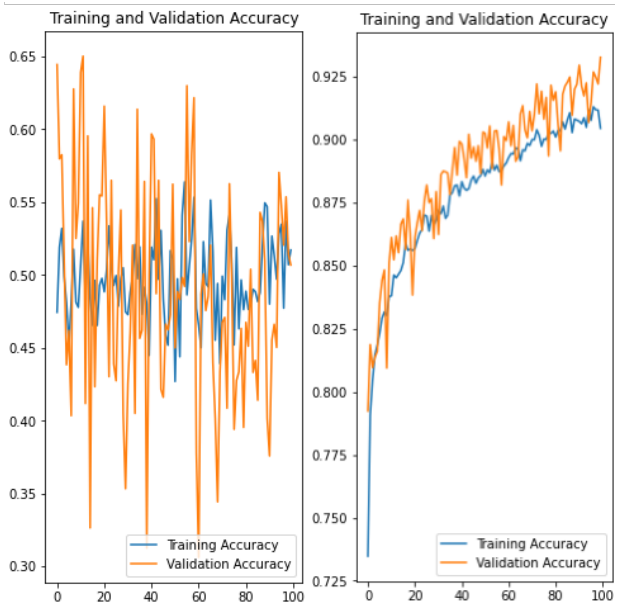


Figure 6.6: Comparison of the faulty and the correct model.

The training accuracy of the binary models reached heights of around 90% from using the model with a bare minimum of augmentation. The new binary model proved to outperform the previous best performance with a new score of 56 %. In comparison, this score is better than before, but it still needs to improve the poor generalization. Therefore, the next attempt included a more aggressive deployment of data augmentation. Other than the data augmentation, some tests were made with the different dropout factors. Similar to the previous attempts of the multiclass, the dropout factors ranged between 0.2 and 0.8. Combining these specifications led to a total of 8 conducted training attempts, with four featuring the augmentation and different factors of dropout and four just featuring the range of dropout. An evaluation was made between each test, and the best-performing model was saved. It was found that the best combination for this attempt was data augmentation and a dropout factor of 0.6, which could achieve a training accuracy of around 95% and an evaluation score of 56,7% (See fig 6.7).

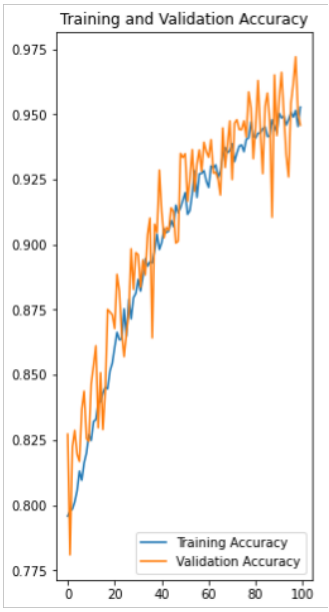


Figure 6.7: The best model was found by a combination of data augmentation and dropout.

The plot from the dropout and data augmentation attempts did not present any clear signs of overfitting, so an effort was made to train the model for an additional 100 epochs to see if this would aid the system's generalization. The increase in epochs leads to a rather time-consuming training, and therefore, this attempt would be the last 200 epoch run before shifting towards a transfer learning approach instead. Once completed, the 200 epoch attempt showed little to no improvement with either the training accuracy of 95,7% (See fig 6.8) or the evaluation score of 57,1%. With this result in mind, the focus was shifted toward transfer learning.

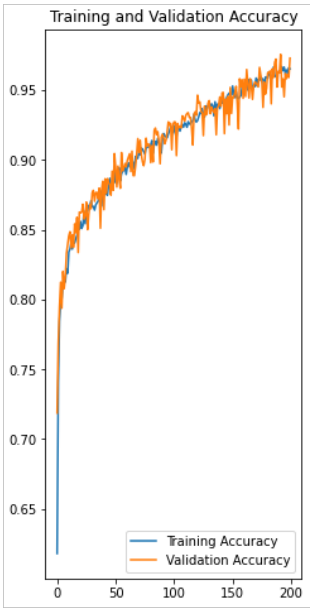


Figure 6.8: The 200 epoch model.

After not achieving the needed results for the model, a second attempt with TL was attempted. Unlike the previous results with multiclass, now a second transfer model would be used as well. The second model, called MobilenetV2, required a similar preprocessing of data as the one used for the VGG-16, with the main difference being that the images needed the 160x160 resolution instead of the 224x224 used by the VGG-16. The first model used for the test was the VGG-16, and all preparations were made like the previous ones used for multiclass attempts. The difference between these attempts is that the final output layer now only consists of one node instead of five. Other than this, the model is identical to the previously used one (See figure 6.5). For the first training attempts, 25 epochs were run only to present a volatile system with a training accuracy of around 80-85 % and a test score of 53 %. A second attempt was made to see if an increase in epochs could improve these scores, but the poor model performance remained unchanged (See fig 6.9). A few tweaks and different data augmentations were applied, and the training epochs were rerun without improvement. Unable to increase the model performance above a measly 54%, the decision to not pursue a VGG-16 solution further was made, and instead, the project focus shifted to the MobilenetV2.

Figure 6.9: The attempts of a binary transfer model with VGG-16.

As mentioned previously, the MobilenetV2 model requires a different resolution of the images, and therefore the dataset was recompiled with the new resolution. Other than the training, a validation and test set was created. The training data was augmented to create a more significant variance in the data and, hopefully, minimize the risk of overfitting. The model then needed to be configured and prepared to start the training, and this was done by following the general method discussed in the previous sections and also used to create the VGG-16 transfer learning model. Firstly, the MobilenetV2 model was downloaded, and the trainable layers of the model were locked to preserve the pretrained weights. These weights were trained using the dataset "Imagenets," the same dataset used for VGG-16. The output layer

used was a fully connected layer consisting of one node, and instead of using a flatten layer, global average pooling was utilized instead. Global average pooling is designed to replace the flatten layer in CNNs and can also replace the fully connected layers commonly used after flattening layers. The general pooling layers can not overfit, unlike the fully connected layers, and are, therefore, a preferable option [47]. To see the complete model, please see the appendix A.

Once all the necessary components were finalized, the training was ready to commence. The selected settings for this attempt were first to perform a five epoch test to find if the model was compiled correctly and proceed to the actual training and evaluation. This test training was a success, and therefore the more extended training sequence was prepared with a total of 40 epochs and using the previously mentioned dataset. The training run was completed without any issues, and upon first inspection, the plotted model values presented a low test accuracy compared to the previously trained models. The MobilenetV2 model did only reach an accuracy of 89%. Disregarding the lower test accuracy, the evaluation was done and presented a new highest performing model with a test accuracy of 73% (See fig 6.10). This new model was saved, and then further attempts were made to improve the performance. These included applying more augmentation and increasing the training duration. These factors were like previous attempts combined, and ten additional training attempts were performed. Throughout these, the evaluations did not record any performance higher than our base attempts for the MobilenetV2 model. The current time frame of the project did not allow for much further development of the model, and therefore the attempt was made to use the MobilenetV2 as the base model for the application.

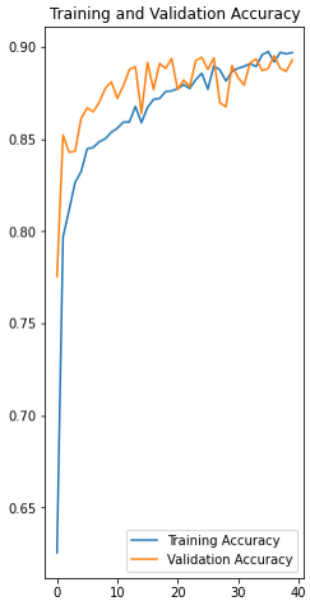


Figure 6.10: The base model that the DOMMUDL app will use.

6.7 Working with the application

One goal of this project was to develop an application as the user interface. There were several steps to go through to create this application. First, there was a planning phase, to decide on tools to use and ways of working. After this phase, it was time to set up the development environments. Lastly, the largest and most time-consuming part, the development process, to develop the application.

6.7.1 Planning

Before developing the application, we had to make some preparations, such as planning the structure and design of the app, see figure 6.11. We decided to make a simple application with only as many pages as necessary to implement all the wanted functions. The necessary functions are to take a photo, upload it to the database and then receive a recommendation.

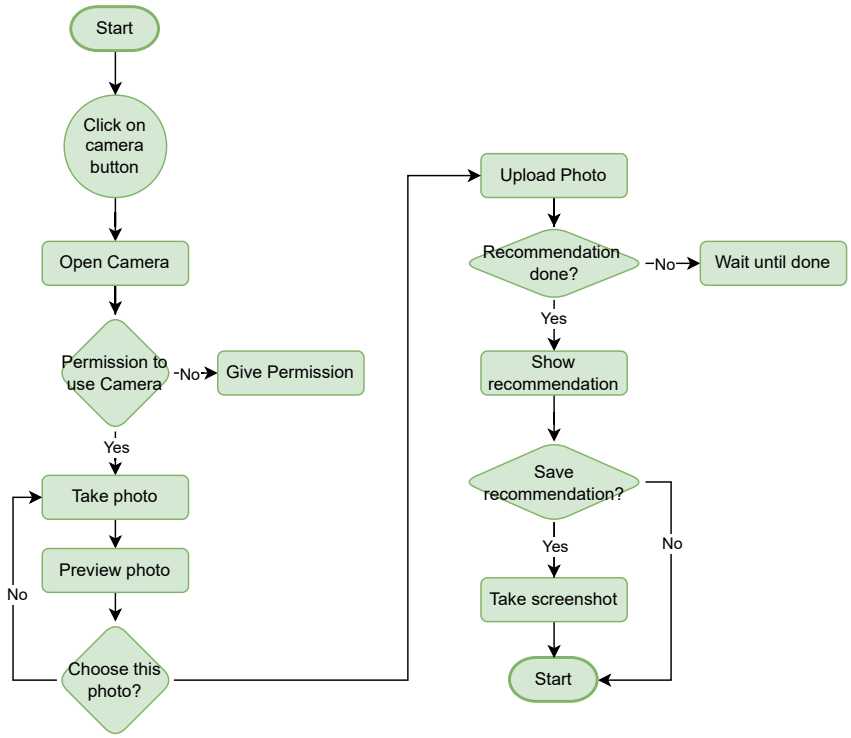


Figure 6.11: Flowchart diagram of the application.

For the development of an application there are several frameworks available. The idea was to create a publicly available application. To achieve this, we aimed to develop the application on iOS and Android platforms. A possibility is to create two apps for the different platforms, but this would be too time consuming. After researching numerous types and considering their pros and cons, it was decided to use Flutter, which is cross-platform, meaning writing one code for two applications.

Another required feature is the communication between the application and the machine learning model. For this, we needed a database. After investigating the options of which database would be an appropriate choice, we decided to use Firebase. It was said to work effectively with both the Flutter framework and the AI modelling. This seemed like an excellent choice seeing how easy was to learn [48]. For developing an iOS app, it is necessary to have a MacBook computer running on Mac OS X version 10.8 or above since apple creates exclusively for their platform. A MacBook Pro from 2019 with the proper specifications for app development was in our possession. Hence this was the computer to be used. An Apple and an android device were accessible, which was needed for device-testing and live-camera-usage-testing later in the project.

6.7.2 Setup

After the planning phase, we started setting up The Flutter framework. The developers behind Flutter have developed a guide on how to initiate a project [49]. We followed this guide for the iOS operating system, and it was a straightforward process. The Flutter SDK was installed on the MacBook by following the instructions. In order for Flutter to run, it requires both the android studio software and the xcode software since it compiles for both platforms, these were installed as instructed. When creating a new project in Flutter, a template establishes. However, this template code would not run and compile on the iOS simulator for some unknown reason. Further on, we decided to test run only on the Android simulator to keep up with the ML model development.

6.7.3 Development

When the frameworks had been installed, and the setup for the application finished, we could start developing the application. A solution to the iOS compiling error was identified, allowing the project to resume testing the application on both platforms. A *home-screen* with a welcome message and a button with a camera icon was designed using widgets from Flutter, predefined UI elements. By clicking this button, the user will be redirected to a camera page. There are several plugins available for the Flutter framework created by the community. One of these plugins is the *Camera*, which allows the user to take a photo, either from the device's camera or select one from the device's gallery. Implementing the *Camera* required permission from the active user to utilize the camera. Changes were made in the specified settings for Android and iOS to activate the permission dialogue, which asks the user for permission when launching the camera. However, Apple does not allow for camera usage in their simulator. As a result, testing was only conducted on Android phones. When the camera function was implemented, compiling the application caused multiple errors. These errors depended on the simulation environment, allowing us to ignore them until testing on an actual device.

When it was possible to take a photo, the question of which format the image needed to be in emerged and if there was a required format or size for the model to accept it. We concluded that having a square image in JPEG format was desirable.

The image taken from a mobile device camera is more often or never square. To do this translation, we cropped the image after it was taken. However, if a user took an image where the mark was not centered, it could be accidentally cropped out of the image. After many attempts at scaling the camera preview into a square and ending up with a stretched image, we added layers on top of the camera preview to indicate where the photo would be cropped so that the user would know and center the mark. When the application allowed for taking a photo and the database was entirely set up, we implemented a way to upload the image to the database. As mentioned earlier, there are several Flutter plugins. One of them is the *provider*, which primarily is an app state manager. In the application, we implemented a Service layer, from heron out it will be called *FireService*, which handles all communication with the database. Creating a provider for *FireService* would serve as a bridge between the application's front end and the database. The database setup is explained in more detail in the next section.

Further, there had to be an interface for showing the recommendation, which was implemented with a dialogue widget. To deal with the data usage integrity, we had an idea of a question about image and data handling, and the user would be required to approve using the functionality of the application. However, this was not a priority, and there was not enough time.

6.7.4 Database setup

For this project a database was set up, and the platform *Firebase* was used during. For Flutter usage the communication with Firebase requires installing the Firebase by simply writing the following line in the system prompt inside the application root folder.

```
flutter pub add firebase_core
```

When installed the application needs to be configured in the Firebase web console [50], which is accessed with an account and a created project. There are instructions for the configuration process for android and the iOS, which mainly consists of downloading a JSON (JavaScript Object Notation) file and placing it in a specific folder. After configuring the applications(android and iOS), the communication with Firebase was possible.

For this project two steps were needed to communicate between the application and the machine learning model. One to upload and temporarily store the images, and another to read and write the recommendation on the image. The services used were *Firebase storage* for image storing and *Firebase Real-Time Database* for the recommendation. Thanks to Firebase being cloud hosted and having a user friendly UI, the implementation was rather simple. The previously mentioned *FireService* was intended as a communication with Firebase and the front-end of the application. Since there were two databases in use, two methods for uploading the image were created. Firstly, the service uploads an image to Firebase Storage, see code below.

```
Future<void> upload(File image) async {
   if (await image.exists()) {
     await _firestorage.ref(_storageName(image)).putFile(
           image,
```

```
        );
    } else {
      print('no file exists');
    }
  }
```

Secondly, the application initiates a template reference in the real-time database with the name of the uploaded image included. This made it possible for the script to find the image to diagnose and connect it to where the recommendation was to be set. The code looks as follows.

```
  Future<void> setName(File image) async {
    await _database.ref('images/${_refName(image)}').set({
      ''date'': ''d'',
      ''imgName'': _storageName(image),
      ''recomendation'': ''tbd'',
      ''time'': ''t''
    });
  }
```

All fields except the *"imgName"* are later updated by the script with the correct information. The *FireService* listens for changes in the *"recommendation"* field and fetches this change to display for the user of the application. The *FireService* also has the responsibility to delete the image from Firebase, this is performed immediately when a recommendation is fetched. In parallel with the application connection to the Database, the script was configured. The implementation worked similarly by first making configurations in the Firebase web console. As well as android and iOS, Firebase also supports web applications, which was adapted for the script connection. The process of establishing the connection for the script was similar to the Flutter process, same steps were to be made, however in Python for the script. Eventually, the database was setup, functional, with uploading images, reading and writing text working unexpectedly quickly.

7
Results

In the introduction, it was mentioned that the project aimed to determine if a machine learning model could be trained to distinguish between skin lesions. And if it was possible to develop an application to reach the general population.

Despite obstacles that were encountered during this project, an AI model has been developed and trained. However, this does not imply that the model can distinguish between the skin lesions well, opening the question of how well the model performs. In the following section, the success rate of the model will be discussed in more detail. In addition to the model, the project also produced a functioning application. Additionally, the script was successfully implemented, combining the application and the model into one seamless system.

First the results of the two best performing models will be demonstrated. Secondly, the script working between the application and the model will be presented. Finally, the application with all its functions will be illustrated.

7.1 Model performance

To evaluate the performance of the developed models, specific metrics needed to be calculated for each attempt. Before these metrics could be found, the models needed to perform their predictions on the 500 nevi and 508 melanomas test sets. The projections were saved and split into one of four categories, True Positives (TP) for melanomas predicted as such, False Negatives (FN) for melanomas predicted as nevi, False Positives (FP) for nevi predicted as melanomas, and finally, True Negatives (TN) for nevi predicted as such.

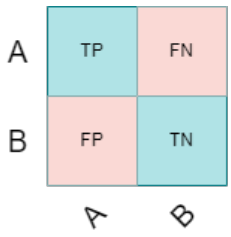


Figure 7.1: A table of different evaluations cases.

Once these results were recorded, the performance values were calculated for each of the three metrics. Where accuracy represents the total percentage of correctly predicted entries, recall represents the correct amount of TP within all the posi-

tive predictions found (TP and FP). While precision shows, the relation of cases predicted as malignant that indeed were malignant(Correct TP of TP and FP).

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{7.1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{7.2}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{7.3}$$

The two models in question are both binary models, one was found by creating a CNN from scratch (see figure 6.8) and the second one was found by utilizing MobileNetV2 TL model (see figure 6.10). After applying the formula 7.1 - 7.3 to both models and using the results of the recorded prediction in the following scores.

Table 7.1: The two models compared.

Model	Accuracy	Recall	Precision
Second best model	57,1 %	18,7%	83,3 %
MobileNetV2 TL model	73%	63,78%	78,6%

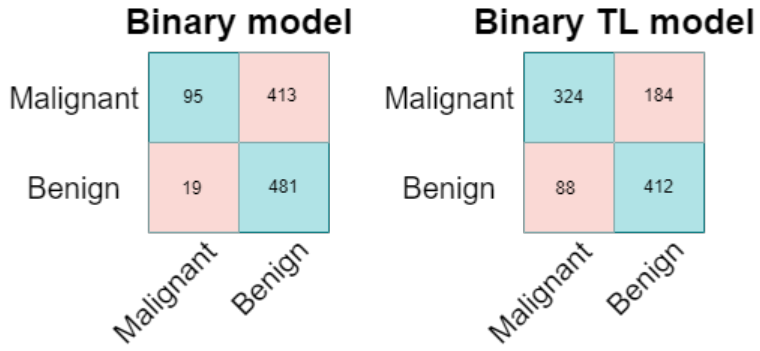


Figure 7.2: The TP and TN table of the two best models.

The final selected model was found using a combination of data augmentation and TL provided by the MobileNetV2 network. It reached a total accuracy of 73% and outperformed the previously best model in every aspect except the precision score. When comparing the models it is found that the model improved its ability to classify malignant lesions (from 95 to 324) while losing some accuracy of benign (from 481 to 412). This decrease in TN resulted in a drop in precision. Despite this, the binary TL model was selected as the preferred model due to the higher accuracy (73% versus 57,1%) and the improved ability to predict the malignant lesions correctly.

One of the goals for this project was to find out if a model produced and trained could identify and class suspected melanomas can now be argued as achieved. This is

because the accuracy achieved in these attempts only reached 73% on the tests. This accuracy could likely be increased with additional time and more data, something that is not available at the time being. Taking this into consideration, this goal should be regarded as met. Achieving this goal concludes the ML section of this project.

7.2 Script

"*Is it possible to implement an application available for the general public?*" was one of the questions asked during the early phases of DOMMUDLs development. And yes, it was possible. One topic implicitly covered by this goal was how we implement the link between our model and the application. Now, this may seem like a minor part compared to the more significant undertakings of this project, but without this link, the application would be nothing by an empty shell. Therefore, a script was created to realize this crucial link. Thanks to the several Firebase features discussed in 3.2.2, this was a straightforward process. Using the real-time database and web application services, this link was implemented. It could now receive entries from the application and provide them with a recommendation within milliseconds. An initial concern was implementing a queue system for the entries. Still, it was later revealed that this was not an issue since the queuing was dealt with automatically by the real-time database. Furthermore, this functionality was utilized to queue entries added to the database when the script was offline. This enabled them to receive recommendations in a first in, first out manner once the script was online again. Overall this script was a success and reached a speed better than initially expected when it, without any issues, could predict and update around three entries per second.

7.3 The application

The development resulted in a functional application with almost all the functionality we aimed for. When started, the application greets the user with a welcome message, the user can click on the camera icon and take a photograph of their skin mark. The user can also choose to read more information about skin lesions or how we handle the data. When the user clicks the camera icon, a camera starts, and they can take their photograph or choose an already taken one from the device's gallery. When they take a photograph, it is possible to make a new attempt. When the user is content with an image, they click "ask for a recommendation," and a loading circle appear. During this time, the application uploads the photograph to Firebase and then listens for changes in the common field "recommendation" where the machine learning model is connected. When the model has made its image classification, it will change this field, and the application receives an indication that there was a change. This change is in the form of text, and the user will see a popup with the image they uploaded and the text recommendation of what to do next. The user then can save this as a screenshot on their phone. If not, the application goes back to the start page, removing the image.

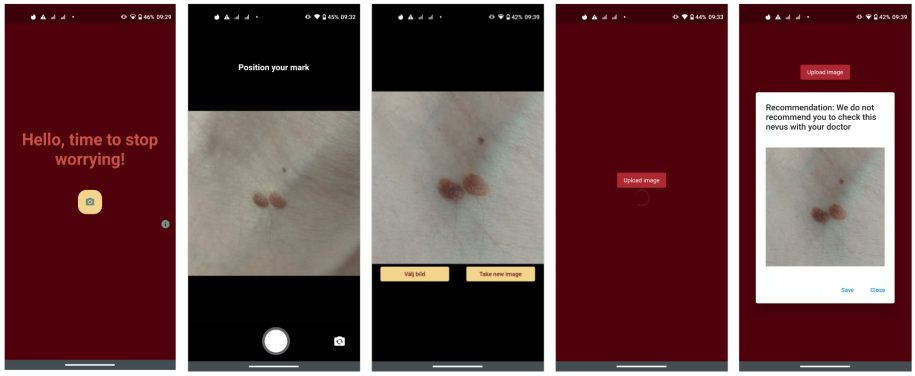


Figure 7.3: A preview of the application.

Explained above is the complete life-cycle of the application in the finished version and is runnable on an android simulator and an actual device in test mode. Due to the cost of becoming an apple developer, there has been no testing on an iOS device. Neither has there been testing on the iOS simulator since it does not have a camera environment. Hence the application only theoretically works on the iOS platform. However, when testing it on android, it works as expected. One goal for this project was to create an application with the above-presented functionalities available for the general public. The application is ready and has the potential to be launched to the general public, which in the scope of this project is an acceptable result that we consider successful.

8

Ethics and Sustainability

The purpose section briefly mentioned how this project could help society to become more sustainable in the future. This argument is based on one of the 17 global goals set by the UN [51]. The goal in question is the third goal and relates to good health and well-being. As the project could enhance access to health care without abusing the existing system, it could contribute to reaching the goal.

This project is not medically-based since the developers of this project do not come from a medical background. Can a recommendation without this background in medicine be trusted? The affected patients could be in a life-threatening situation, and this raises the question: Could it be ethically wrong to make these recommendations using an AI model? One could argue that an AI model with the right amount of data and time potentially is better than a human. However, in order to collect the necessary data there might be the need for a collaboration with the health industry.

Another potential issue is the privacy of the data providers. The data in the form of photographs are the provider's property unless explicit permission is given, which results in difficulties when gathering necessary data. There is also the problem with not violating the provider's integrity when gathering data. The idea of this project is to gather photographs from users in order to provide a recommendation. To keep the users as safe and protected as possible there is an opportunity not to save any photographs or information given.

9
Conclusion

Throughout the course of this project, an AI model with the ability to differentiate between benign and malignant skin lesions was created. However certain constraints, mainly the lack of data and tight time frame of the project halted the development once the model had reached an accuracy of 73%. Several different AI models have been developed and evaluated using both models built from scratch and pretrained models that enable TL. The best-performing model is found by utilizing TL with the mobileNetV2 model. To further develop this model one would need an influx of new medically assessed and preferably cellphone taken image data, combined with a more extensive time frame, something left for potential further research.

Together with the goal to develop an AI model, the other main goal was to find a way of distributing the model in such a way that it could reach the general public. For this task, an android application was developed using the tools provided by the Flutter framework. To enable the link between the application and the AI model, a special script was designed and implemented once again using the Flutter framework.

The necessary functions to provide the user with the AI recommendation were implemented using the script and application. Therefore, we consider the application and script successful. Even though a better-performing AI model would have been desirable if this application was to be launched, it was just not possible to implement with the given constraints. Taking these factors into consideration the results of the said model are deemed satisfactory.

9.1 Discussion

Looking at the planning for this project, see appendix B, the majority of the time was to be spent on development. In retrospect, the schedule coincides fairly closely with the actual process. We managed the project well, using only the planned weeks for planning and all the weeks for development. However, when it came to writing the report, the actual time spent deviated slightly from the schedule. Writing began later than anticipated and took significantly longer than expected. To devote sufficient time to the development, writing was postponed, which resulted in a heavy writing workload. While the plan was well thought out, the execution could have been improved.

When working with machine learning, we spent a substantial amount of time cleaning and sorting the data. These lengthy processes were caused by significant levels of noise within the data. This could include images with too much hair cover-

ing the mark or other factors that obscure the mark. It proved to be more difficult than anticipated to correctly sort the images since special scripts were developed to avoid the long times required otherwise. Consequently, less time was available for the development of models. The training of the models also required a remarkable amount of time despite being provided with an enhanced graphics card, and a powerful computer. The shortest training session lasted approximately three hours, while the longest lasted eight hours. On average, they were given four hours of training. The time we had left quickly elapsed, given that there are only eight hours in a day. This comprehensive training process is another factor contributing to the shortage of time for the development of models. We believe that this research is successful based on our existing knowledge, as well as research conducted in the areas of self-made models, transfer learning, tuning and more. Even though the outcome is not flawless, we are pleased with the results.

Implementing the script, the communication link between the application and the model, was surprisingly easy. The time set aside for this part was not utilized. One possible explanation could be the use of Firebase which did not require the development of triggers. The results show that many of the necessary functions already exist in Firebase and are available for use. One example is the queuing system. Before launching this project, we set out a goal of classifying an image in 10 seconds, which was considered the longest time possible before someone would lose patience. The result, however, was a classification rate of three images per second. In other words, our response time was less than 0.33 seconds for a single image, exceeding our expectations. By developing the script in a short time, we were able to devote a great deal of time to the complicated and time-consuming ML part.

The development of the application could have been more seamless. Working with Apple devices presented several challenges. According to our results, the finalized product has not been tested in the iOS environment, and at this time there is no possibility of doing so. These issues were not anticipated or taken into consideration at the outset of the project. The plan was to develop for both platforms. Despite having the computer and necessary equipment, we discovered that this was not possible. Prior to beginning the project, it may have been advantageous to review the extent of the iOS simulator. By doing this, we would have been able to understand the nuances of developing for iOS and the limitations related to using the camera. Although there are several things to consider and several features we would like to see in the application, we consider the application's functionality on Android satisfactory.

When working with machine learning, there are multiple ethical issues. Who is responsible? What information is being collected? Has someone's integrity been violated? As a result of the model's performance, it can be debated if it would be ethical to use it to provide recommendations based on it. However, it may be possible to advance the model with more time and more data and achieve better performance. If the model performed better, the recommendations would be more reliable, and other arguments could be enough to justify its use. As an example, consider improvements in people's health care as mentioned in the introduction and the sustainability chapter. The issue of time recurs throughout the report. Additional time would enable the application to be improved further. The application

could be enhanced with improvements like better image managing and more background information about the lesions. Furthermore, the application could be tested on an iOS device if an investment was made in the Apple developer program.

In the introduction, we present the health benefits this product could provide. Such as lightening the workload of the health care system and decreasing people's worry about their potential skin cancer. Although the final product can be improved in the future by spending additional time on model development, we do see that the product has the potential to help relive the health care system. This will decrease people's worry about potential skin cancer, which was our aim.

9.2 Further research

To continue this project, we see the need for better performance in the machine learning model, which needs more data and training. The data needed is images that are analyzed and diagnosed by a doctor. To get such images, a possibility could collaborate with a hospital to collect the data. After this train, a better model and an app that the hospital can stand behind make it more trustworthy and usable.

The different types of malignant skin lesions have characteristics that differentiate them. Mostly it has to do with the appearance, but sometimes it can matter when they arrived or if something has changed recently, how big they are and if they have grown, where on the body they are. Also, information about the mark holder can be substantial, like the patients age and gender (see chapter 3.4.1). These are things that doctors consider when detecting skin lesions. Could an AI improve by receiving this kind of data about the image provider? If so, this gives for a further future development possibility, where a user is given a chance to provide such important information.

Bibliography

[1] "GPU Availability and Pricing Update: December 2021."

[2] 1177, "Vårdgaranti, URL: https://www.1177.se/sa-fungerar-varden/lagar-och-bestammelser/vardgaranti/," accessed: 2022-02-21. [Online]. Available: https://www.1177.se/sa-fungerar-varden/lagar-och-bestammelser/vardgaranti/

[3] SKR, "Väntetider i vården," Sveriges Kommuner och Regioner, blog, accessed: 2022-02-21. [Online]. Available: https://skr.se/vantetiderivarden.46246.html

[4] Strålsäkerhetsmyndigheten, "Sverige har snart samma nivåer av hudcancerfall som Australien," accessed: 2022-02-21. [Online]. Available: https://www.stralsakerhetsmyndigheten.se/press/nyheter/2021/sverige-har-snart-samma-nivaer-av-hudcancerfall-som-australien/

[5] Y. N. Fu'adah, N. C. Pratiwi, M. A. Pramudito, and N. Ibrahim, "Convolutional Neural Network (CNN) for Automatic Skin Cancer Classification System," in *IOP Conference Series: Materials Science and Engineering*, vol. 982. IOP Publishing, 2020, p. 012005.

[6] E. Nasr-Esfahani, S. Samavi, N. Karimi, S. M. R. Soroushmehr, M. H. Jafari, K. Ward, and K. Najarian, "Melanoma detection by analysis of clinical images using convolutional neural network," in *2016 38th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. IEEE, 2016, pp. 1373–1376.

[7] J. Paul, "7 best frameworks libraries for cross-platform android and ios apps in 2022," accessed: 2022-05-23. [Online]. Available: https://medium.com/javarevisited/top-5-frameworks-to-create-cross-platform-android-and-ios-apps-in-2020-d02edf3d01f1

[8] Dart.dev, "Dart guides," accessed: 2022-04-27. [Online]. Available: https://dart.dev/guides/language/language-tour

[9] Python, "General Python FAQ — Python 3.10.4 documentation," Apr. 2022, accessed: 2022-04-27. [Online]. Available: https://docs.python.org/3/faq/general.html#what-is-python

[10] Simplilearn, "What is keras: The best introductory guide to keras," accessed: 2022-05-11. [Online]. Available: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html

[11] G. Thomas, "What is flutter and why you should learn it in 2020," accessed: 2022-04-11. [Online]. Available: https://www.freecodecamp.org/news/what-is-flutter-and-why-you-should-learn-it-in-2020/

[12] D. Stevenson, "What is firebase? the complete story, abridged," accessed: 2022-04-13. [Online]. Available: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0

[13] BBC, "Intel chief warns of two-year chip shortage," *BBC News*, Jul. 2021. [Online]. Available: https://www.bbc.com/news/technology-57996908

[14] Merriam-Webster.com, "Benign," accessed: 2022-05-11. [Online]. Available: https://www.merriam-webster.com/dictionary/benign#note-1

[15] ——, "Malignant," accessed: 2022-05-11. [Online]. Available: https://www.merriam-webster.com/dictionary/malignant

[16] G. I. Henry, "Benign skin lesions," accessed: 2022-05-11. [Online]. Available: https://emedicine.medscape.com/article/1294801-overview

[17] T. A. C. S. medical and editorial content team, "What is melanoma skin cancer?" accessed: 2022-05-11. [Online]. Available: https://www.cancer.org/cancer/melanoma-skin-cancer/about/what-is-melanoma.html

[18] M. C. Staff, "Melanoma, symptoms and causes," accessed: 2022-05-11. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/melanoma/symptoms-causes/syc-20374884

[19] S. Schultz, "Malignt melanom – hudcancer," accessed: 2022-05-11. [Online]. Available: https://www.1177.se/Vastra-Gotaland/sjukdomar--besvar/cancer/cancerformer/malignt-melanom--hudcancer/

[20] M. C. Staff, "Basal cell carcinoma, symptoms and causes," accessed: 2022-05-11. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/basal-cell-carcinoma/symptoms-causes/syc-20354187

[21] ——, "Squamous cell carcinoma of the skin, symptoms and causes," accessed: 2022-05-11. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/squamous-cell-carcinoma/symptoms-causes/syc-20352480

[22] S. Schultz, "Skivepitelcancer – hudcancer," accessed: 2022-05-11. [Online]. Available: https://www.1177.se/Vastra-Gotaland/sjukdomar--besvar/cancer/cancerformer/skivepitelcancer--hudcancer/

[23] M. C. Staff, "Moles, symptoms and causes," accessed: 2022-05-23. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/moles/symptoms-causes/syc-20375200

[24] Åsa Schelin, "Godartade hudförändringar," accessed: 2022-05-12. [Online]. Available: https://www.1177.se/Vastra-Gotaland/sjukdomar--besvar/hud-har-och-naglar/fodelsemarken-och-hudforandringar/godartade-hudforandringar/

[25] T. M. LeLeux, "Pathology of benign melanocytic nevi," accessed: 2022-05-12. [Online]. Available: https://emedicine.medscape.com/article/1056742-overview

[26] IBM, "What is Deep Learning?" accessed: 2022-03-23. [Online]. Available: https://www.ibm.com/cloud/learn/deep-learning

[27] C. M. Bishop, "Neural networks and their applications," *Review of scientific instruments*, vol. 65, no. 6, pp. 1803–1832, 1994.

[28] IBM, "What are Neural Networks?" accessed: 2022-03-23. [Online]. Available: https://www.ibm.com/cloud/learn/neural-networks

[29] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[30] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on knowledge and data engineering*, vol. 22, no. 10, pp. 1345–1359, 2009. [Online]. Available: https://ieeexplore.ieee.org/document/5288526/authors#authors

[31] I. Kandel and M. Castelli, "Transfer learning with convolutional neural networks for diabetic retinopathy image classification. A review," *Applied Sciences*, vol. 10, no. 6, p. 2021, 2020. [Online]. Available: https://www.mdpi.com/666280

[32] Y. Gao and K. M. Mosalam, "Deep transfer learning for image-based structural damage recognition," *Computer-Aided Civil and Infrastructure Engineering*, vol. 33, no. 9, pp. 748–768, 2018.

[33] E. Deniz, A. Şengür, Z. Kadiroğlu, Y. Guo, V. Bajaj, and Ü. Budak, "Transfer learning based histopathologic image classification for breast cancer detection," *Health information science and systems*, vol. 6, no. 1, pp. 1–7, 2018.

[34] R. Barman, S. Deshpande, S. Agarwal, U. Inamdar, M. Devare, and A. Patil, "Transfer learning for small dataset," in *Proceedings of the National Conference on Machine Learning, Mumbai, India*, vol. 26, 2019.

[35] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[36] S. V. Lab, "ImageNet," publisher: Stanford Vision Lab. [Online]. Available: https://www.image-net.org/index.php

[37] T. Kaur and T. K. Gandhi, "Automated brain image classification based on VGG-16 and transfer learning," in *2019 International Conference on Information Technology (ICIT)*. IEEE, 2019, pp. 94–98.

[38] S. S. Yadav and S. M. Jadhav, "Deep convolutional neural network based medical image classification for disease diagnosis," *Journal of Big Data*, vol. 6, no. 1, pp. 1–18, 2019.

[39] C. Sitaula and M. B. Hossain, "Attention-based VGG-16 model for COVID-19 chest X-ray image classification," *Applied Intelligence*, vol. 51, no. 5, pp. 2850–2863, 2021.

[40] R. Rokhana, W. Herulambang, and R. Indraswari, "Multi-Class Image Classification Based on MobileNetV2 for Detecting the Proper Use of Face Mask," in *2021 International Electronics Symposium (IES)*. IEEE, 2021, pp. 636–641.

[41] C. Buiu, V.-R. Dănăilă, and C. N. Răduţă, "MobileNetV2 ensemble for cervical precancerous lesions classification," *Processes*, vol. 8, no. 5, p. 595, 2020. [Online]. Available: https://www.mdpi.com/717830

[42] ISIC, "Archive," accessed: 2022-04-27. [Online]. Available: https://www.isic-archive.com/#!/topWithHeader/tightContentTop/about/aboutIsicOverview

[43] IBM, "What is Overfitting?" accessed: 2022-04-28. [Online]. Available: https://www.ibm.com/cloud/learn/overfitting

[44] *MACHINE LEARNING A First Course for Engineers and Scientists*, New ed ed., Sweden.

[45] C. P. Chai, "The importance of data cleaning: Three visualization examples," *Chance*, vol. 33, no. 1, pp. 4–9, 2020. [Online]. Available: https://www.mdpi.com/717830

[46] Google, "Google Colaboratory," accessed: 2022-05-14. [Online]. Available: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/images/classification.ipynb#scrollTo=dC40sRITBSsQ
[47] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.
[48] V. Yadav, "Add firebase to flutter," accessed: 2022-05-09. [Online]. Available: https://viveky259259.medium.com/add-firebase-to-flutter-6bc9e2755284
[49] Flutter, "Flutter guide," accessed: 2022-05-02. [Online]. Available: https://docs.flutter.dev/get-started/install/macos
[50] G. Developers, "Firebase, make your app the best it can be," accessed: 2022-05-23. [Online]. Available: https://firebase.google.com
[51] D. o. E. United Nations and S. Affairs, "The 17 goals," accessed: 2022-05-13. [Online]. Available: https://sdgs.un.org/goals

A

The MobileNetV2 Model

Layer (type)	**Output Shape**	**Parameter**
InputLayer	160, 160, 3	0
Conv2D	80, 80, 32	864
BatchNormalization	80, 80, 32	128
ReLU	80, 80, 32	0
Depth	80, 80, 32	288
BatchNormalization	80, 80, 32	128
ReLU	80, 80, 32	0
Conv2D	80, 80, 16	512
BatchNormalization	80, 80, 16	64
Conv2D	80, 80, 96	1536
BatchNormalization	80, 80, 96	384
ReLU	80, 80, 96	0
ZeroPadding2D	81, 81, 96	0
DepthwiseConv2D	40, 40, 96	864
BatchNormalization	40, 40, 96	384
ReLU	40, 40, 96	0
Conv2D	40, 40, 24	2304
BatchNormalization	40, 40, 24	96
Conv2D	40, 40, 144	3456
BatchNormalization	40, 40, 144	576
ReLU	40, 40, 144	0
DepthwiseConv2D	40, 40, 144	1296
BatchNormalization	40, 40, 144	576
ReLU	40, 40, 144	0
Conv2D	40, 40, 24	3456
BatchNormalization	40, 40, 24	96
Add	40, 40, 24	0
Conv2D	40, 40, 144	3456
BatchNormalization	40, 40, 144	576
ReLU	40, 40, 144	0
ZeroPadding2D	41, 41, 144	0
DepthwiseConv2D	20, 20, 144	1296
BatchNormalization	20, 20, 144	576
ReLU	20, 20, 144	0

Conv2D	20, 20, 32	4608
BatchNormalization	20, 20, 32	128
Conv2D	20, 20, 192	6144
BatchNormalization	20, 20, 192	768
ReLU	20, 20, 192	0
DepthwiseConv2D	20, 20, 192	1728
BatchNormalization	20, 20, 192	768
ReLU	20, 20, 192	0
Conv2D	20, 20, 32	6144
BatchNormalizatio	20, 20, 32	128
Add	20, 20, 32	0
Conv2D	20, 20, 192	6144
BatchNormalization	20, 20, 192	768
ReLU	20, 20, 192	0
DepthwiseConv2D	20, 20, 192	1728
BatchNormalization	20, 20, 192	768
ReLU	20, 20, 192	0
Conv2D	20, 20, 32	6144
BatchNormalization	20, 20, 32	128
Add	20, 20, 32	0
Conv2D	20, 20, 192	6144
BatchNormalization	20, 20, 192	768
ReLU	20, 20, 192	0
ZeroPadding2D	21, 21, 192	0
DepthwiseConv2D	10, 10, 192	1728
BatchNormalization	10, 10, 192	768
ReLU	10, 10, 192	0
Conv2D	10, 10, 64	12288
BatchNormalization	10, 10, 64	256
Conv2D	10, 10, 384	24576
BatchNormalization	10, 10, 384	1536
ReLU	10, 10, 384	0
DepthwiseConv2D	10, 10, 384	3456
MatchNormalization	10, 10, 384	1536
ReLU	10, 10, 384	0
Conv2D	10, 10, 64	24576
BatchNormalization	10, 10, 64	256
Add	10, 10, 64	0
Conv2D	10, 10, 384	24576
BatchNormalization	10, 10, 384	1536
ReLU	10, 10, 384	0
DepthwiseConv2D	10, 10, 384	3456
BatchNormalization	10, 10, 384	1536
ReLU	10, 10, 384	0

Conv2D	10, 10, 64	24576
BatchNormalization	10, 10, 64	256
Add	10, 10, 64	0
Conv2D	10, 10, 384	24576
BatchNormalization	10, 10, 384	1536
ReLU	10, 10, 384	0
DepthwiseConv2D	10, 10, 384	3456
BatchNormalization	10, 10, 384	1536
ReLU	10, 10, 384	0
Conv2D	10, 10, 64	24576
BatchNormalization	10, 10, 64	256
Add	10, 10, 64	0
Conv2D	10, 10, 384	24576
BatchNormalization	10, 10, 384	1536
ReLU	10, 10, 384	0
DepthwiseConv2D	10, 10, 384	3456
BatchNormalization	10, 10, 384	1536
ReLU	10, 10, 384	0
Conv2D	10, 10, 96	36864
BatchNormalization	10, 10, 96	384
Conv2D	10, 10, 576	55296
BatchNormalization	10, 10, 576	2304
ReLU	10, 10, 576	0
DepthwiseConv2D	10, 10, 576	5184
BatchNormalization	10, 10, 576	2304
ReLU	10, 10, 576	0
Conv2D	10, 10, 96	55296
BatchNormalization	10, 10, 96	384
Add	10, 10, 96	0
Conv2D	10, 10, 576	55296
BatchNormalization	10, 10, 576	2304
ReLU	10, 10, 576	0
DepthwiseConv2D	10, 10, 576	5184
BatchNormalization	10, 10, 576	2304
ReLU	10, 10, 576	0
Conv2D	10, 10, 96	55296
BatchNormalization	10, 10, 96	384
Add	10, 10, 96	0
Conv2D	10, 10, 576	55296
BatchNormalization	10, 10, 576	2304
ReLU	10, 10, 576	0
ZeroPadding2D	11, 11, 576	0
DepthwiseConv2D	5, 5, 576	5184
BatchNormalization	5, 5, 576	2304

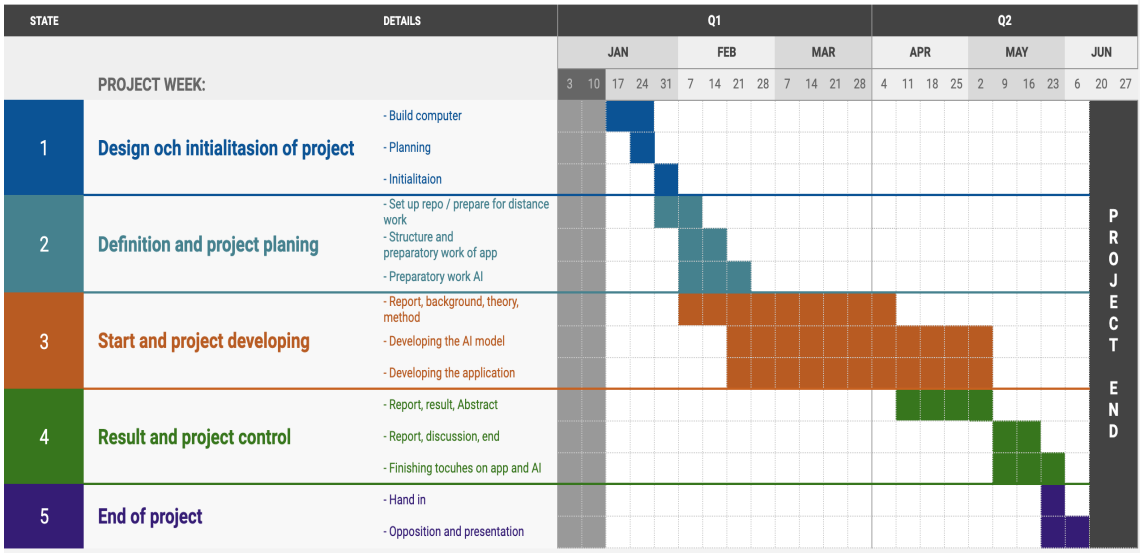
ReLU	5, 5, 576	0
Conv2D	5, 5, 160	92160
BatchNormalization	5, 5, 160	640
Conv2D	5, 5, 960	153600
BatchNormalization	5, 5, 960	3840
ReLU	5, 5, 960	0
DepthwiseConv2D	5, 5, 960	8640
BatchNormalization	5, 5, 960	3840
ReLU	5, 5, 960	0
Conv2D	5, 5, 160	153600
BatchNormalization	5, 5, 160	640
Add	5, 5, 160	0
Conv2D	5, 5, 960	153600
BatchNormalization	5, 5, 960	3840
ReLU	5, 5, 960	0
DepthwiseConv2D	5, 5, 960	8640
BatchNormalization	5, 5, 960	3840
ReLU	5, 5, 960	0
Conv2D	5, 5, 160	153600
BatchNormalization	5, 5, 160	640
Add	5, 5, 160	0
BatchNormalization	5, 5, 960	3840
ReLU	5, 5, 960	0
DepthwiseConv2D	5, 5, 960	8640
BatchNormalization	5, 5, 960	3840
ReLU	5, 5, 960	0
Conv2D	5, 5, 320	307200
BatchNormalization	5, 5, 320	1280
Conv2D	5, 5, 1280	409600
BatchNormalization	5, 5, 1280	5120
ReLU	5, 5, 1280	0
GlobalAveragePooling2D	1280	0
Dropout	1280	0
Dense	1	1281

B

Project GANTT schedule

Project Timeline

PROJECT NAME	Detection of malignant melanomas using neural networks	COMPANY	Broccoli Engineering AB
PROJECT LEADERS	Simon Länsberg and Anna Manfredsson	DATE	18-03-12



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY / UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden
www.chalmers.se

CHALMERS