Final Year Project Report

Full Unit - Final Report

Comparison of Image Classification Models with Transfer Learning

James Arnott

A report submitted in part fulfilment of the degree of

BSc (Hons) in Computer Science

Supervisor: Li Zhang

Department of Computer Science
Royal Holloway, University of London

March 26, 2023

Declaration

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count: XXXX

Student Name: James Arnott

Date of Submission: March 26, 2023

Video Link:

Signature:

Table of Contents

Abstract

Transfer learning is a common, efficient method for training deep learning models. It involves using a pre-trained model to extract features from a dataset, and then training a new model on top of the extracted features. This project report gives a comparison of the performance of several image classification models when using transfer learning and evaluates their performance on a dataset with MRI scans of Alzheimer's patients. The models compared are MobileNetv2, MobileNetv3-Small, MobileNetv3-Large, InceptionV3, ResNet101 and VGG19.

Project Specification

Aims: To implement and compare on benchmark data sets various machine learning algorithms Background: Machine learning allows us to write computer programs to solve many complex problems: instead of solving a problem directly, the program can learn to solve a class of problems given a training set produced by a teacher. This project will involve implementing a range of machine learning algorithms, from the simplest to sophisticated, and studying their empirical performance using methods such as cross-validation and ROC analysis. In this project you will learn valuable skills prized by employers using data mining. Early Deliverables

- You will implement simple machine learning algorithms such as nearest neighbours and decision trees.
- They will be tested using simple artificial data sets.
- Report: a description of 1-nearest neighbour and k-nearest neighbours algorithms, with different strategies for breaking the ties;
- Report: a description of decision trees using different measures of uniformity.

Final Deliverables

- May include nearest neighbours using kernels and multi-class support vector machines.
- The algorithms will be studied empirically on benchmark data sets such as those available from the UCI data repository and the Delve repository.
- For many of these data set judicious preprocessing (including normalisation of attributes or examples) will be essential.
- The performance of all algorithms will be explored using a hold-out test set and cross-validation.
- The overall program will have a full object-oriented design, with a full implementation life cycle using modern software engineering principles.
- Ideally, it will have a graphical user interface.
- The report will describe: the theory behind the algorithms.
- The report will describe: the implementation issues necessary to apply the theory.
- The report will describe: the software engineering process involved in generating your software.
- The report will describe: computational experiments with different data sets and parameters.

Suggested Extensions

- Modifications of known algorithms.

- Dealing with machine learning problems with asymmetrical errors (such as spam detection) and ROC analysis.
- Nontrivial adaptations of known algorithms to applications in a specific area, such as medical diagnosis or option pricing.
- Exploring the cross-validation procedure, in particular the leave-one out procedure. What is the optional number of folds?
- Comparative study of different strategies of reducing multi-class classifiers to binary classifiers (such as one-against-one, one-against-the-rest, coding-based).

Reading

- Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The Elements of Statistical Learning. Second edition. Springer, New York, 2009.
- Tom M. Mitchell. Machine Learning. McGrow-Hill, New York, 1997.
- Vladimir N. Vapnik. The Nature of Statistical Learning Theory. Second edition. Springer, New York, 2000.
- Vladimir N. Vapnik. Statistical Learning Theory. Wiley, New York, 1998.
- Seth Hettich and Steven D. Bay. The UCI KDD Archive. University of California, Department of Information and Computer Science, Irvine, CA, 1999, http://kdd.ics.uci.edu.
- Delve archive. http://www.cs.toronto.edu/ delve.

Addendum

As I was not keen on doing the pre-assigned generic project, I had asked for permission to do a project of my own choosing. I was given permission to do so, and I chose to do a project on comparison of image classification models using transfer learning. I chose this project as I was interested in transfer learning as I had used it in a previous project and wanted to learn more about it and it seemed far more interesting than the generic project and I believe this is generally a more applicable problem to solve in the real world.

Chapter 1: **Introduction**

My project is about the comparison of image classification models with transfer learning. In this report I shall cover the background of transfer learning, the models I have chosen to compare, the datasets I have chosen to use, the pre the results of my experiments and the conclusions I have drawn from them.

The importance of transfer learning cannot be overstated, it makes training models for anyone with a computer and an internet connection possible, It opens up possibilities for people who would otherwise not be able to train models as far less data and computing power is required and where properly applied, they can be more accurate than models trained from scratch.

Chapter 2: **Rationale**

2.1 Background

Alzheimer's disease is a neurodegenerative disease that affects the brain. It is the most common cause of dementia, and 1 in 3 people born in the UK this year will develop dementia in their lifetime.[3]

The symptoms of Alzheimer's disease are memory loss, confusion, mood swings, and difficulty with everyday tasks, it can also eventually lead to death, however the slow progression disease causes many prople to live with it for years and slowly lose their independence, and forget who they are.

There have recently been developments in drug discovery and development that have been able to slow the progression of AD, but are only effective in the early stages of the disease, and are not able to reverse the damage that has already been done. This means that early diagnosis is essential to ensure that the drugs are as effective as possible.[10]

Transfer learning can be a useful tool to create predictions of the progression of AD, as it can be used to train models on a smaller dataset than would otherwise be required, as MRI scans are expensive and time consuming to obtain.

Chapter 3: **Neural Networks**

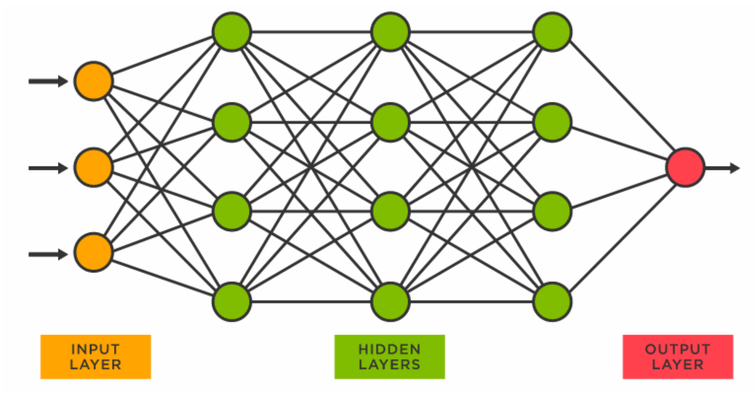


Figure 3.1: A diagram of a neural network [22]

3.1 What is a Neural Network

A neural network is a machine learning model that is inspired by the way the human brain works. It is made up of a series of layers, each layer is made up of a number of neurons. Each neuron is connected to every neuron in the next layer, and each connection has a weight associated with it. The weights are used to determine how much each neuron in the next layer is affected by the current neuron. The weights are updated during training, and the process of updating the weights is known as backpropagation.

3.2 Why use a Neural Network

The use of neural networks has increased dramatically in recent years, and they are now used in a variety of different applications. The reason for this is that they are very good at learning complex patterns in data, and they are able to learn these patterns without being explicitly programmed to do so. This saves a lot of time and effort when compared to other machine learning techniques.

They are far more flexible than many of the traditional strategies for image classification, due to the fact they can adapt to new data and are resilient to noise in the data or changes such as lighting or rotation.

3.3 What are the alternatives to Neural Networks in Image Classification

This is not an exhaustive list for image classification, but it is a list of some of the most common alternatives to neural networks that produce the best results.

3.3.1 Support Vector Machines

There are a number of different alternatives to neural networks, one common alternative is Support Vector Machines. Support Vector Machines are a type of supervised machine learning model that can be used for both classification and regression. They work by finding a hyperplane that separates the data into different classes, and then classifying new data based on which side of the hyperplane it is on. They are very good at classifying data that is linearly separable, otherwise they are not as good, and they are also very sensitive to noise in the data.

In one study using SVMs to classify Alzheimer's patients, the researchers managed to get an accuracy of 62.64% using MRI scans of the brain. They managed to get between 83% and 90% accuracy when using SVMs with clinical parameters, however combining the two methods in this study did not improve the accuracy. [38]

3.3.2 Content Based Image Retrieval

Content Based Image Retrieval (CBIR) is a type of image retrieval system that uses the actual content of an image as the basis for retrieving similar images from a large database. It works by extracting the features from the image and then using those features to compare to other images in the database. The features are usually based on a combination of color, texture, shape, and other attributes. The system then produces a list of images that are similar to the one used for the search query. CBIR systems are particularly useful for searching for images that can't be easily described using traditional keywords.

CBIR can be used to identify images that are not in the database by analyzing the content of the images. By extracting features from the image such as color, texture, and shape, CBIR can be used to compare the content of the query image with the content of the images in the database. It can then return images that are similar in content to the query image even if they are not present in the database.

For example, if the target image was a blue sky with white clouds, then the algorithm would return images that had similar features such as a blue sky and white clouds to the target image.

Although this method of image classification would not seem to be very useful for classifying images of Alzheimer's patients, a study was done using this method to classify Alzheimer's patients, and they had managed to get an accuracy of 87% using MRI scans[1] and getting feedback from the physicians who were treating the patients.

For classifying images of flowers it could also be very useful, as it would allow the user to search for images of a specific flower. This could help identify flowers by recognizing the different characteristics of each flower, such as the shape of the petals, the color of the petals, and the overall structure of the flower.

3.4 The different optimizers

In deep learning, optimisers are algorithms used to adjust the parameters of a model in order to minimise a loss function. Optimisers are used to improve the accuracy of a model by reducing the error rate. Common optimisers used in deep learning include Stochastic Gradient Descent (SGD), Adaptive Moment Estimation (Adam), and Root Mean Square Propagation (RMSProp). There are many different optimizers that can be used to train a model, and the one that is used depends on the type of problem that is being solved.

The impact using different optimisers can have is dependent on the type of problem being solved. For example, SGD is often used for linear regression problems, while Adam is better suited for deep learning problems. Additionally, the choice of optimiser can also affect the speed of convergence and the accuracy of the model.

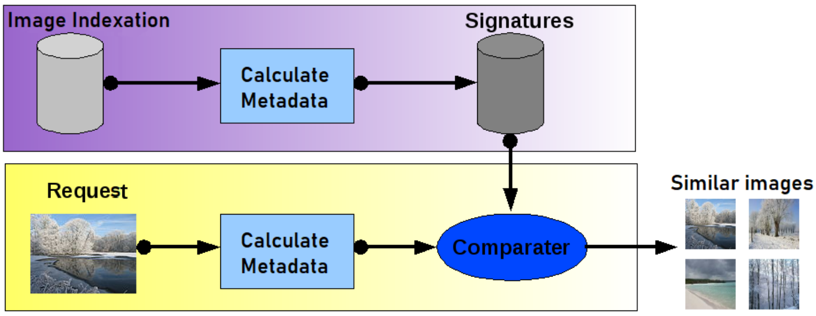


Figure 3.2: A diagram of Content Based Image Retrieval [8]

3.5 Classification Vs Regression

Classification is a type of supervised machine learning model that is used to predict the class of a given data point. It is used to predict the class of the given data, whereas regression is used to predict the value of a given data point. For example, classification would be best suited to classifying images for dogs or cats, whereas regression would be best suited to predicting the price of a house.

Alzheimer's disease is not a binary classification problem, as it is not a case of either having Alzheimer's or not having Alzheimer's, it is more of a spectrum of different stages of the disease.

The OASIS-1 dataset contains a CDR score for each patient, which is a measure of the severity of the disease. This means that the problem is more of a regression problem, as it is trying to predict the severity of the disease, rather than trying to classify the patient as having Alzheimer's or not, however as there are only 4 different scores in the dataset, classification may be more suited to this problem, for this specific dataset.

3.6 What is a Convolutional Neural Network

A convolutional neural network (CNN) is a type of neural network that is used for image classification. It is made up of a series of convolutional layers, pooling layers, and fully connected layers. The convolutional layers are used to extract features from the image, and the pooling layers are used to reduce the size of the image. The fully connected layers are used to classify the image. They are often much more efficient than other types of neural networks, and are commonly used for image classification.

3.7 Types Of Layers

- Convolutional Layer: This layer extracts features from an input image and creates a feature map.
- Pooling Layer: This layer reduces the dimensionality of a feature map while preserving its most important features.
- Dropout Layer: This layer randomly ignores nodes during training to reduce overfitting.
- Fully Connected Layer: This layer connects all the neurons of the previous layer to every neuron in the next layer. It helps in mapping input to output.

3.8 Using an LSTM for classification with MRI scans

An LSTM (Long Short Term Memory) network is a type of recurrent neural network that is used for time series prediction, it is typically not used with transfer learning as it is not as efficient as other types of neural networks. The specific LSTM structure allows for multiple MRI scans to be used as input to the model, and the model, so it would be able to classify the Alzheimer's CDR score more accurately, as it could take into account more scans, using the raw scan information, instead of the pre-processed data that I would otherwise have to use with regular CNN model that I could use with transfer learning.

3.9 What transfer learning is

Transfer learning is a common, efficient method for training deep learning models. There are many different ways to implement transfer learning and here I hope to compare the performance of a variety of different methods with different pre-trained models.

The applications for transfer learning are vast, and it is a common method for training deep learning models, This is because it is a very efficient method for training models as most of the work is done by the pre-trained model. The pre-trained model is used to extract features from the dataset, and then a new model is trained on top of the extracted features. The pre-trained model is usually frozen so that it does not change during training, and only the new model is trained.

After the new model is trained on top, a process known as fine-tuning can be used to further improve the performance of the model. Fine-tuning involves training the new model on the dataset, but with a lower learning rate than the initial training. This is also where the model is usually unfrozen, so that the pre-trained model can be trained as well, which helps to improve the overall performance of the model.

Chapter 4: **The Datasets I will be using**

The dataset I have chosen to primarily use for this project is the OASIS-1[24] dataset. This dataset contains 416 patients, ageing from 18 to 96 years old. It is primarily composed of patients without Alzheimer's disease.

I have also chosen to include another dataset which should be easier to classify it's used in the Intelligent skin cancer diagnosis using improved particle swarm optimization and deep learning models[36]. It's a combination of 2 datasets, the Edinburgh Research and Innovation (Dermofit)[4], and the Dermatology Service of Hospital Pedro Hispano[21]. There are 484 images in the dataset, with 2 different classes, benign (270) and malignant (214). All the images are 200x200 pixels, with 3 channels (RGB), which is not optimal as they require upscaling to be used with the models I will be using for transfer learning.

In the paper from the dataset creators, they used KNN, SVM and a CNN model to classify the images in the dataset. With the optimal hyperparameters, they achieved an accuracy of 99.54% using the SVM and RCPSO which stands for Real-Coded Particle Swarm Optimization, this is a nature-inspired optimization technique used to search for the best solution in a given problem space, which was used for optimising the hyperparameters of the SVM model.

4.1 The Contents of OASIS-1

This information is from the fact sheet, associated with the dataset. [25] The OASIS-1 dataset contains a few different associated parameters, as listed here from the CSV:

'ID,M/F,Hand,Age,Educ,SES,MMSE,CDR,eTIV,nWBV,ASF,Delay'

- ID: The ID of the patient
- 'M/F': The gender of the patient
- 'Hand': The primary hand of the patient
- 'Age': The age of the patient
- 'Educ': The education level of the patient 1: less than high school grad., 2:High school grad., 3: some college, 4: college grad., 5: beyond college
- 'SES': The socioeconomic status of the patient
- 'MMSE': The Mini-Mental State Examination score of the patient
- 'CDR': The Clinical Dementia Rating score of the patient
- 'eTIV': The Estimated Total Intracranial Volume of the patient
- 'nWBV': The normalized whole brain volume of the patient
- 'ASF': The Atlas Scaling Factor of the patient

The augmentations I can use for this dataset cannot be as extensive as the skin cancer dataset, as the scans are pretty consistent in size and shape, along with being in greyscale, so I will only be using the following augmentations:

- Random horizontal flipping
- Random vertical flipping
- Random rotation (20 degrees)
- Random zoom

4.2 The Contents of the Skin Cancer Dataset

The skin cancer dataset contains 2 different classes, benign and malignant, and 484 images in total. This is all the information that is available about the dataset, as it is not a publicly available dataset, this requires me to augment the data as much as possible to get good results.

The augmentations I will be using are:

- Random horizontal flipping
- Random vertical flipping
- Random rotation
- Random zoom (359 degrees)
- Random brightness
- Random offset

Chapter 5: **Transfer Learning**

5.1 Why use Transfer Learning

Transfer learning can enable a model to be trained much more efficiently than if it was trained from scratch, as most of the work is done by the pre-trained model. Aside from the extra time it takes to train a model from scratch, it often requires more training data, which is not always available and can be expensive to obtain. Transfer learning can often get better results than training a model from scratch, as the pre-trained model has already learned about different features from a large dataset, and the transfer learning model can then use these features to learn about the new dataset.

5.2 Drawbacks of Transfer Learning

The main drawback of transfer learning is that it is not always possible to use it, as the pre-trained model may not be able to extract the features from the new dataset. With image classification, the pre-trained model can typically extract features from any image, but it may not be able to extract the features that are needed for the new dataset. Transfer learning can also be less efficient than training a model from scratch, as the pre-trained model will have all the weights from the original dataset, which may not be needed for the new dataset. This can cause the model to be less efficient, as it will have to train on weights that are not needed but will still affect the performance of the model.

5.3 The code behind the transfer learning

The code for the transfer learning is shown below.

```
# This imports the model from the keras library
MobileNetV3Small = keras.applications.MobileNetV3Small(
    input_shape=(224, 224, 3),
    include_top=False,
    weights='imagenet')

# Create the model
model = keras.Sequential()

# Freezing all but 10 layers of the pre-trained model
for layer in MobileNetV3Small.layers[:-10]:
    layer.trainable = False

# Adding the pre-trained model to the model, along with
# a dropout layer and a fully connected layer
model.add(MobileNetV3Small)
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(512, activation='relu'))
model.add(keras.layers.Dropout(0.6))
model.add(keras.layers.Dense(512, activation='relu'))
model.add(keras.layers.Dropout(0.6))
model.add(keras.layers.Dense(4, activation='softmax'))
```

The code above shows how to create a model using transfer learning, creating the model structure consists of adding the pre-trained model to the model, and then adding a few layers on top of it. The Flatten layer is used to flatten the output of the pre-trained model as that is the input for the fully connected layers. The fully connected layers are used to classify the images.

The dropout layers are used to prevent overfitting, and the dense layers are used to classify the images. The dense layers are potentially the most important part of the model, as they are the ones that are actually learning about the dataset. They get trained on the features extracted by the pre-trained model, and the weights of the dense layers are what is used to classify the images.

The final layer of the model is a Softmax layer, which is used to classify the images. It is used to classify the images into one of the classes in the dataset, in this example, the 4 levels of Alzheimer's disease that are in the Alzheimer's MRI Dataset.

5.4 How to implement Transfer Learning

5.4.1 Extracting features

Feature extraction is the process of extracting features from the images in the dataset, these features in the images could be anything, such as the colour, shape, or the texture of the image. All of these features are used to classify the images, and having these features already extracted can make the training process much faster. Most of the pre-trained model is usually frozen so that it does not change during the initial stages training, and only the new model is trained, and on top of the extracted features from the base model.

5.4.2 Fine-tuning

This is the process of training the new model on the dataset, but with a lower learning rate than the initial training and the model is usually unfrozen, so that the pre-trained model can be trained as well, which helps to improve the overall performance of the model. This was where I managed to get the best results for my models.

5.5 How to fine-tune a model

Fine-tuning a model is a very simple process, and can be done in just a few lines of code. The first step is to unfreeze the pre-trained model, which can be done by setting the trainable attribute of the layers to True. The next step is to set the learning rate to a lower value, which can be done by setting the learning rate attribute of the optimizer. The final step is to recompile the model, which can be done by calling the compile method of the model. Here is an example of how I fine-tuned a model using Keras:

```
base_model.trainable = True
optimizer.learning_rate = 0.0001
model.compile(optimizer=optimizer, loss=loss, metrics=metrics)
```


```
base_model.trainable = True
optimizer.learning_rate = 0.0001
model.compile(optimizer=optimizer, loss=loss, metrics=metrics)
```


5.6 My training setup

For this project, I have set up my gaming laptop as a server, with Manjaro[19] as the operating system. I've chosen my laptop to do the training on due to it's powerful GPU, which is a NVIDIA GeForce GTX1070[23]. This can be used with Nvidia CUDA[9] to speed up the training process, which is what I have done. Training the models on my laptop has been very useful, as it has allowed me to train the models much easier as I can leave it to train overnight without having to constantly check on it.

I have recently upgraded the GPU in my computer to an RTX 3080[31], which is a much more powerful GPU than the GTX1070, as it has more CUDA cores, a higher core clock, 2gb more of VRAM, and a much higher memory bandwidth. Alongside this, it has "RT Cores", these are designed to be used for ray tracing, which is a technique used to render realistic images, however they also can accelerate the training process.

Chapter 6: **Data Importing and Preprocessing**

6.1 Formatting and converting the data

Clinica[32] is a python library that can be used to convert the MRI scans in the different formats into the BIDS format[11]. This significantly speeds up development as the standardised BIDS format lets me use the same code for the different datasets, as well as allowing me to use pre existing tools to import the data and preprocess it, ready for training.

MRI data can be complicated and before the BIDS standardisation, it was significantly more difficult for researchers to share data with each other. It contains information such as the manufacturer of the scanner, the model of the scanner, the software associated with the scanner and version, then other parameters about the setup of the scanner, such as the field strength and the number of channels.[5]

Whilst I have not used the BIDS standard for all the datasets in this project, I feel that it is very useful and I will be using it for all the future datasets that I use.

6.2 Dataset Augmentation

When training a deep learning model, it is important to have a large dataset, as this will allow the model to learn from a wide variety of data. Unfortunately I do not have as many MRI scans as I would like, so I have used data augmentation to increase the size of the dataset. Using the ImageDataGenerator class from Keras[17], I have been able to generate new the new images. This helps training the model as it allows the model to learn from a wider variety of data, and it also helps to prevent overfitting. It is not optimal to be required to use data augmentation, however it's significantly better than not having enough data to train the model. If these models were to be used in a real world scenario, augmentation helps to make the model more resilient to the quality of the scans.

6.3 Batch sizes

The batch size is the number of images that are passed to the model at a time. A larger batch size will cause the model to learn quicker, however it will also use more memory whereas having a smaller batch size will cause the model to learn slower, however it will use less memory. When I started off, I did not understand the concept of batch sizes, and I was using a batch size of 1000, this lead to my training times being very long and I had to wait days for a model to train. I have found that a batch size of 32 works well for the models that I have trained and my GTX1070 8 GB GPU[23] doesn't often run out of memory.

6.4 Feature Normalisation

Normalisation is the process of scaling the data so that it is typically between 0 and 1. In the case of age, this would be the highest age in the dataset, divided by the age of the patient. This is done to make the training process more efficient, as it helps with convergence, as it can avoid the model from getting stuck in a local minimum. This can help reduce the impact of outliers by scaling the input to a smaller range. Although it is not required, it is generally recommended to always normalize the data.

6.5 Keras Data Generators

When passing the batched images and the other data to the model, I had to create my own superclass for the Keras Sequence class. This allowed me to pass the age, gender, MMSE score, and the image to the model, instead of just the image. I needed to create this as I did not have enough ram to load all of the images into memory, so using batches was required. This still allowed me to use the ImageDataGenerator class to augment the images.

Chapter 7: **Evaluating the performance**

7.1 The different metrics

Accuracy is typically the metric used to compare models and is defined as the number of correct predictions divided by the total number of predictions. It is a very simple metric to understand, and is therefore often used to compare the performance of different models. There is accuracy for both the training and validation sets, and the accuracy of the validation set is used to determine how well the model generalises to new data, and a validation accuracy that is much lower than the training accuracy is a sign that the model is overfitting. However, it is not always the best metric to use, as it can be misleading in some cases.

Loss is a very common and useful method to evaluate the performance of a model, it is calculated by taking the difference between the predicted value and the actual value. There are however many different types of loss functions, and the one that is used depends on the type of problem that is being solved. These could be binary cross entropy, categorical cross entropy, or mean squared error. For my project I used binary cross entropy due to it's simplicity and the fact that it is the most common loss function used for binary classification problems.

The validation loss and accuracy are used to determine how well the model generalises to new data, and a validation loss that is much higher than the training loss is a sign that the model is overfitting. For the validation set, I have assigned 20% of the data to be used.

7.2 Data Augmentation

Data augmentation is the process of artificially increasing the size of a dataset by applying random transformations or filters to the images. This can help to improve the performance of a model, as it can help to reduce overfitting, and can also help to improve the generalisation of the model. The transformations that are applied to the images are usually random, and can include things like rotating the image, flipping the image, or changing the contrast/saturation of the image.

In the case of the Alzheimer's MRI Dataset, it is extremely useful to have more data to train the model on, as there are a limited number of brain scans available. MRI scans are also very expensive, so using data augmentation can significantly reduce the cost of training the model.

In the Alzheimer's MRI Dataset, the majority of the archive consists of augmented images already however this caused me problems which I will go on to talk about. With the flower dataset however, I did apply data augmentation to the images, as there were not enough images per class to train the model on.

7.3 Problems With The Datasets and How I Fixed Them

7.3.1 The Problems

The Alzheimer's MRI dataset that I had chosen early on in the project was far from ideal as I found out later on. The pre-augmentation that was applied to the dataset caused problems with the training/validation split, as the images were already augmented, and the validation set was made up of images that were in the training set, but with different transformations applied to them. This gave me amazing results on the validation set, but when I tested the model on new data, it performed very poorly. Here the image shows the last 30 epochs of the training process, and the validation accuracy is very high, higher than what other models from other research papers have achieved.

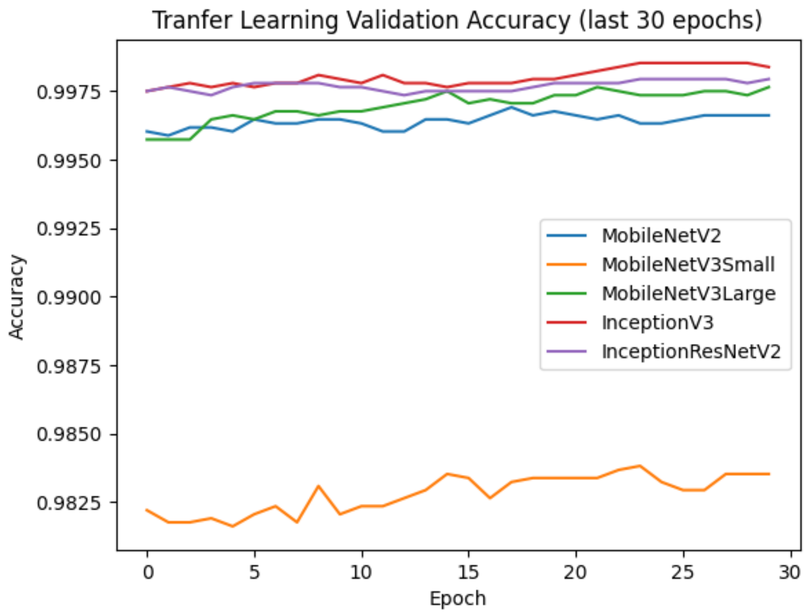


Figure 7.1: The training results from the various models (Last 30 epochs)

7.3.2 The solution

The solution to this problem was to use the original images, and not the pre-augmented images. This meant that I had to re-download the dataset, and re-split the dataset into training and validation sets, and then apply data augmentation to the images. This gave me much better results, and the model was able to generalise to new data much better. Here is the image of the training results from the various models, and the validation accuracy is much lower than before.

Here's the snippet of code that I used to augment the images:

```
train_datagen = ImageDataGenerator(rescale=1./255,
    validation_split=0.2,
    featurewise_center=True,
    featurewise_std_normalization=True,
    rotation_range=360,
    width_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=True,
)
```

This however did not solve all my problems, as the model was still overfitting. This ended up being a problem with the dataset, as even with the "original images" that were provided, there were still images that were repeated, and aside from the overfitting problems this caused, it also meant that my validation set had images that were in the training set, and this gave me very high validation accuracy.

This was then solved by going directly with the OASIS-1 dataset, and using the images that were provided there. I was hesitant to do this at first, as there was more pre-processing required to format the data in the way that I wanted it, but it has now allowed me to train my models without getting false results.

By using the OASIS-1 dataset directly, that presented a new set of challenges, as the images were given in a different format, as I had a CSV file with the labels and the image IDs which identified the images. This lead me to learn how to use the Pandas[26] library in Python, which allowed me to read the CSV file and extract the image IDs and the labels from it. Then I had to create my own custom data generator, which allowed me to load the images from the dataset, and apply the transformations to them. This was a very useful learning experience, as I was able to learn important skills relating to data preprocessing. The data generator, which I had talked about earlier, helps batch and augment the data that is being fed into the model during training.

7.4 Evaluating the given metrics

The metrics of accuracy and loss generally tell us how well the model is performing, but they do not tell us how well the model is performing for each class. They do also not tell us how well the model would work in a real world scenario, as they are only calculated on the training and validation sets, of which we have a limited amount of data.

The validation dataset is also from the same distribution as the training dataset, so it is not a good representation of how well the model would work in a real world scenario as the scans could be of a different quality.

I have used the augmented dataset for the training and validation sets as this should give a better representation of how well the model would work in a real world scenario.

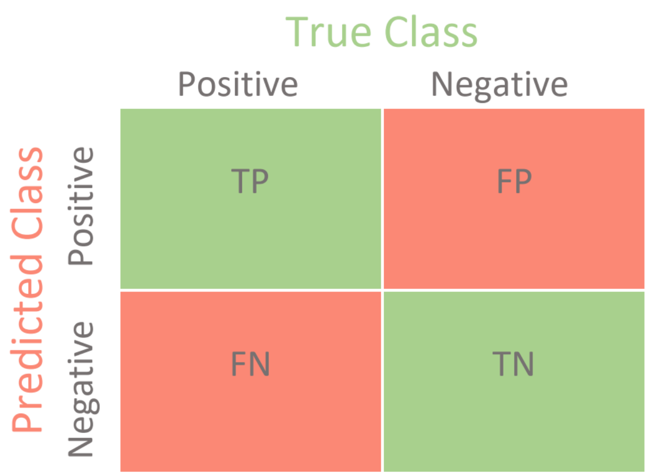


Figure 7.2: A confusion Matrix[7]

For the final report I intend on using confusion matrices to evaluate the performance of the models, as they can show which classes the model has the best or worst performance on. They can also represent nicely in a table, all the different hyperparameter combinations that I will try to use to get the best possible results.

7.5 The different datasets

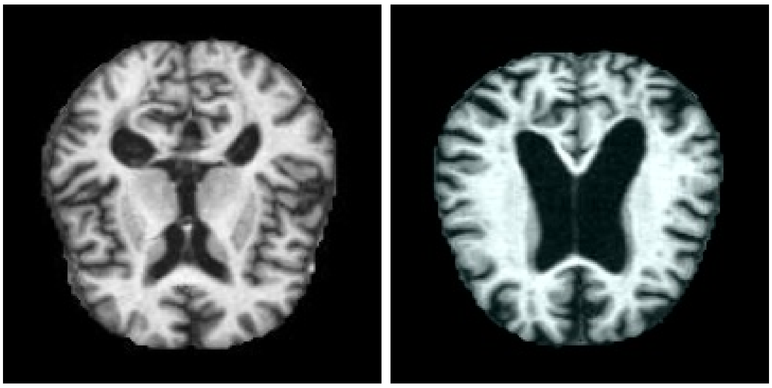


Figure 7.3: Samples from the Alzheimer's MRI Dataset[2]

Chapter 8: The Different Models That Are Being Compared

All the pre-trained models are from the Keras library[17] and were trained on the ImageNet dataset[15].

8.1 Mobilenetv3

As described in the "Searching for Mobilenetv3" paper[13], this model is designed for mobile devices, and is a successor to the Mobilenetv2 model. It is a comparatively small model as it is optimised for mobile devices, and is also very fast.

The resulting models from this paper are:

- Mobilenetv3-Large
- Mobilenetv3-Small

The idea to have a large and small model is to specialise the model for different use cases and different hardware, increasing the speed or accuracy of the model. The large model is designed for high accuracy, and the small model is designed for high speed, there may also be cases where the smaller model is the only viable option due to hardware constraints.

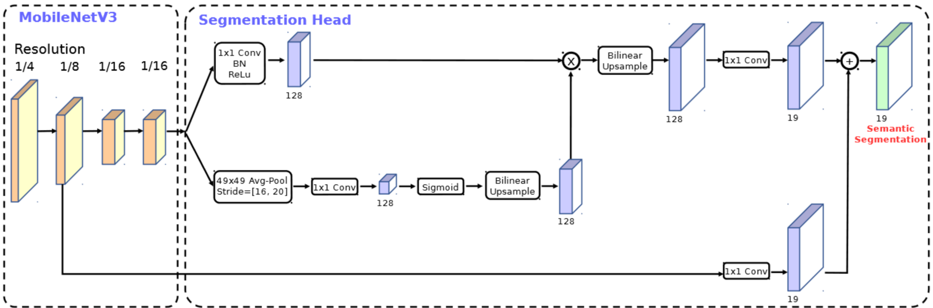


Figure 8.1: The proposed segmentation head for MobileNetV3[13]

8.2 InceptionV3

InceptionV3[34], it's one of the most popular models for image classification as it is comparatively fast and accurate. It's feature extraction capabilities are much more accurate than traditional methods such as hand-crafted feature extraction. It is a deep learning model that uses a combination of convolutional layers, pooling layers, and fully-connected layers. It uses auxiliary classifiers, these are additional classifiers used in a neural network to predict the output of the main classifier. Auxiliary classifiers are used to improve the accuracy of the main classifier by providing additional input data. They can also help to reduce the amount of overfitting that can occur when using a single classifier.

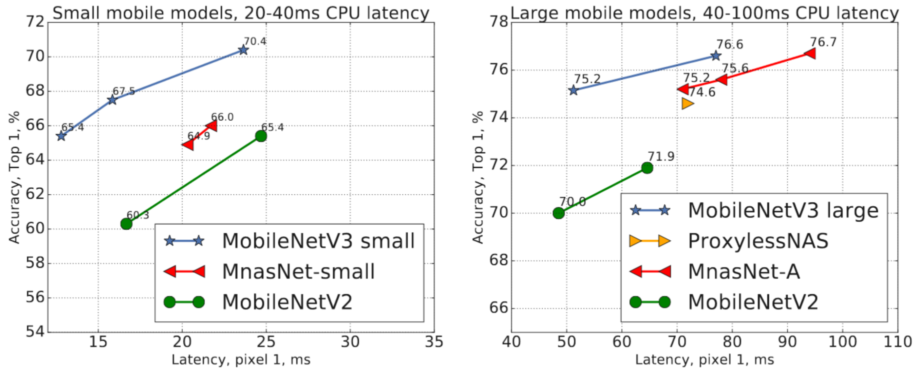


Figure 8.2: A comparison of latency and accuracy[13]

8.3 ResNet101

Resnet101[12] is another popular model, it's the largest model that is being compared and different due to the amount of layers that it has, which helps with the accuracy of the model, however it is also much slower than the other models to train and to run inference on.

8.4 VGG19

VGG stands for Visual Geometry Group, it is a model that was developed by the Visual Geometry Group at Oxford University. VGG19[33] is a model which has 19 convolutional layers

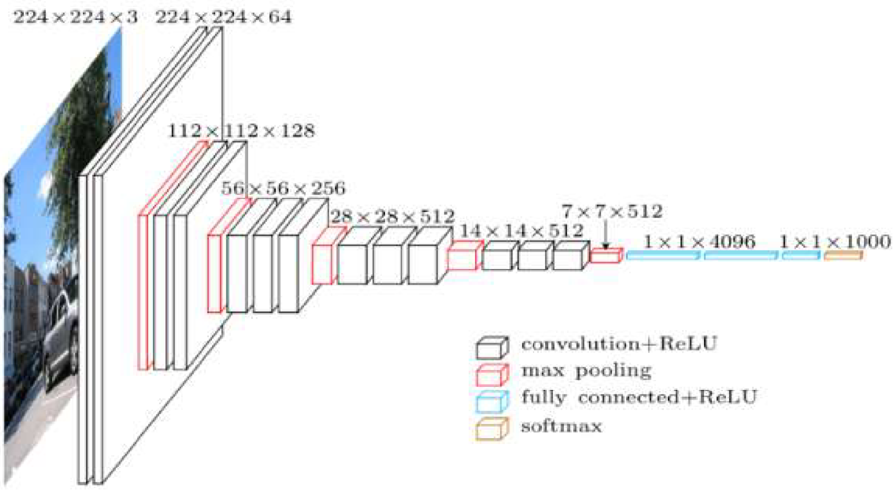


Figure 8.3: The structure of VGG19[33]

8.5 EfficientNet

[35]

Chapter 9: **Software Engineering**

9.1 The code

The way I have structured the code is by having a main.py file that contains the code to train the model, and a Main.ipynb file that contains the code to test setting up the model.

9.2 My Development Environment

I have used the following tools to develop this project:

- Python 3.8.5[28] - To run the code
- Jupyter Notebook[16] - To help me develop the code and generate the graphs
- Visual Studio Code[39] - As my IDE to develop in, I have used the SSH feature of VsCode to SSH into my Linux server and modify the code there quickly
- GitLab[30] - To store the code and to track the changes
- tmux[37] - To run the code in the background
- CUDA[9] - To train the models on the GPU

9.3 Using Git

I have used Git to manage the code, and have used the RHUL GitLab server to host the code. This was not the first time I have used Git, however it was the first time I have used it with GitLab. I used separate branches for the different stages of the project, and then merged them into the master branch when they were complete. The branches were made to create the different planning papers, and then to set up the development environment. I also used branches to create the different models, and then to test them too.

All of my commits were in the format of Commitlint[6] to make it easier to read the commit messages, make them more consistent, and to make it easier to go through the history and find a specific change or commit to cherry pick or revert.

9.4 Testing with Jupyter Notebook

I have used Jupyter Notebook to test the code, and to generate the graphs. This was the most useful tool for me as it allowed me to test the code quickly and easily, I had many issues with setting up the code as this was the first time I have trained any model.

9.5 Using Test Driven Development

I used PyTest[27] to test the code, and to make sure that everything was working as expected. This helped me quickly test and debug my code, I would write a test to define how I wanted my code to work, and then write the code to make the test pass. This sped up development as I could quickly test the code and make sure that it was working as expected.

The other tests I wrote were making the graphs that I generated, as I was testing the code by generating the graphs using Jupyter Notebook[16] and matplotlib[20], and then checking that the graphs were looking correct. This was really the only way I could test the training code, as the models take 10+ hours to train, and I didn't have the time to train the models multiple times to see if unit tests that I had written were correct.

In the future I would like to use unit tests as I could set up an automated pipeline to train the models and then run the unit tests, and then if the unit tests fail, then the pipeline would fail, and I would be notified. This would allow me to test the code more easily, and to make sure that the code is working correctly. I would also be able to test the code on a smaller dataset, which would allow me to train the models quicker, and to test the code quicker.

Aside from testing the models, I could set up unit tests for an API that I could create for anyone to upload their brain scans to and get a prediction of whether they have Alzheimer's or not. This would give me an opportunity to write unit tests for the API, and to test the code that I would write for the API.

9.6 Different learning rates

When training a model, the learning rate is the rate at which the model learns from the data. A low learning rate will cause the model to learn slower, and a high learning rate will cause the model to learn quicker but it can also cause the model to get stuck in a local minimum. A combination of a high learning rate whilst the base model is frozen, and a low learning rate whilst the base model is unfrozen is a good way to train a model as it allows the model to learn quickly at the start, and then fine-tune the model at a slower rate.

A local minimum is when the model can't decrease its loss any further by adjusting its parameters. It is stuck at a 'valley' in the loss landscape with the parameters that it currently has. A global minimum is the same concept, but it is the absolute lowest the model can take its loss. It is the lowest point in the entire loss landscape no matter what parameters the model has.

Chapter 10: **The User Interface**

10.1 The Front End

For the front end I have used React[29] as it is a popular framework for creating web applications, and I've got experience with it. It specifically uses React Typescript, which is a combination of React and Typescript. Typescript is a superset of Javascript, and it adds types to Javascript. This allows for better code completion, and it makes it easier to debug the code, as it will tell you if you've made a mistake sooner, whereas javascript would give you an error at runtime, if you end up with a type error. Typescript helps counter many of the issues with JS however it's not perfect and there are still issues where there can be scope for errors.

I do have the option of using tensorflow-JS, which can process all of the data locally, this has benefits and drawbacks, the main benefit is that the user doesn't have to upload their data to a server, which can make the handling more secure as everything is processed locally, so there would be far less red tape with data protection laws. The main drawback is that the model would have to be downloaded to the user's device, which could be a problem if the model is large, and it would also mean that the user would have to wait for the model to be downloaded before they could use the application, then they would have to wait for the model to make a prediction after this.

10.2 The Back End

I've decided to use Python for the API backend, as it's the main server side language I have used so far. I can use Flask to host the api and return the results when they have been processed by the model.

Chapter 11: Identifying and Mitigating Bias

11.1 The OASIS-1 Dataset

The OASIS-1 dataset[24] that I used for this project contains 416 patients with 434 scans.

11.1.1 nWBV vs Age

Giving the age of the patient to the machine learning algorithm may allow for bias, this is because there are very few patients who are young and have Alzheimer's disease, in the dataset. This also means it would likely be problematic to give the normalized whole brain volume to the machine learning algorithm, as this is likely to be highly correlated with age, and will therefore allow for bias.

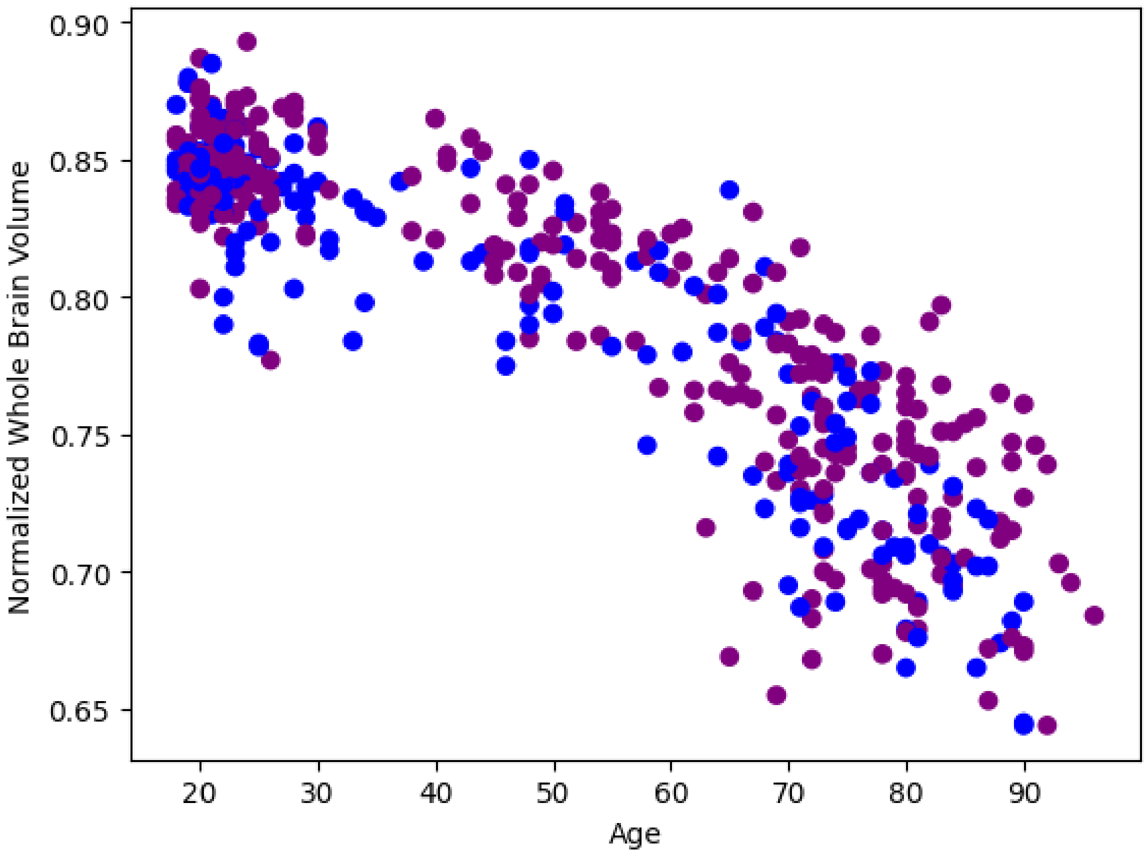


Figure 11.1: The nWBV of the patients in the dataset

11.1.2 eTIV vs Age

This scatter graph is significantly more random than the one for age and whole brain volume, which means it is likely better to give to the machine learning algorithm, as it shouldn't give any strong indication of the age of the patient, and it should be useful for the machine learning algorithm. One potential issue with this parameter is that women are more likely to have Alzheimer's disease, and this graph shows how the men and women are separated, which may allow for bias based off of the gender of the patient.

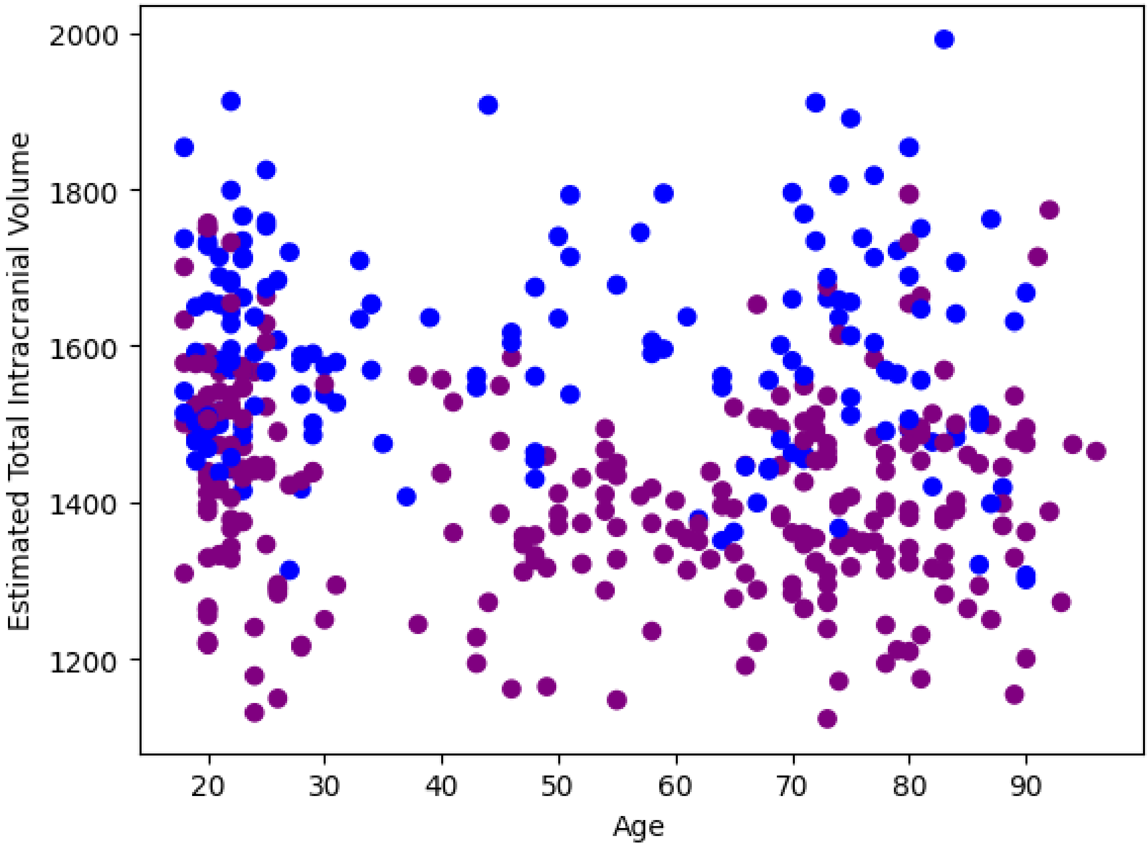


Figure 11.2: The eTIV of the patients in the dataset

11.2 The Skin Cancer Dataset

The Skin cancer dataset that I was provided for this project contains images of benign and malignant skin cancer.

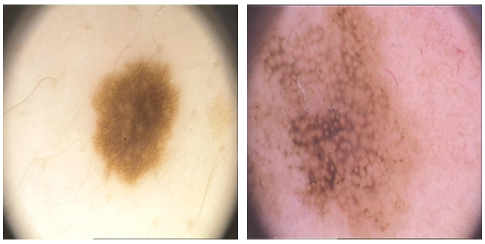


Figure 11.3: Benign and malignant skin cancer image

These are some example images of what the training and test data looks like. The images are all 200x200 pixels and are in RGB format. Unfortunately they appear to only be pictures of caucasian skin, which means that the model may not be able to predict skin cancer on people with other ethnic backgrounds. There does however appear to be multiple types of skin cancer in the dataset, which means that the model should be able to predict a variety of skin cancers. With this particular dataset, I don't have the ability to do much to mitigate bias, as the dataset is already biased towards caucasian skin, and my points of reference for model accuracy are with this dataset. This dataset is also relatively small, with only 484 images all together. Although this is a small number, the results from training with this dataset are still very good, and the model is able to predict skin cancer with a high accuracy, it just have slightly more bias than it would if it was trained with a larger and more diverse dataset.

Chapter 12: **My Initial Faulty Training Results**

12.0.1 The Comparison Of The Different Models

Here is a graph of all the different models that I have trained properly with the old and faulty Alzheimer's dataset from kaggle. The graph shows the accuracy of the models on the validation set along with the accuracy on the training set.

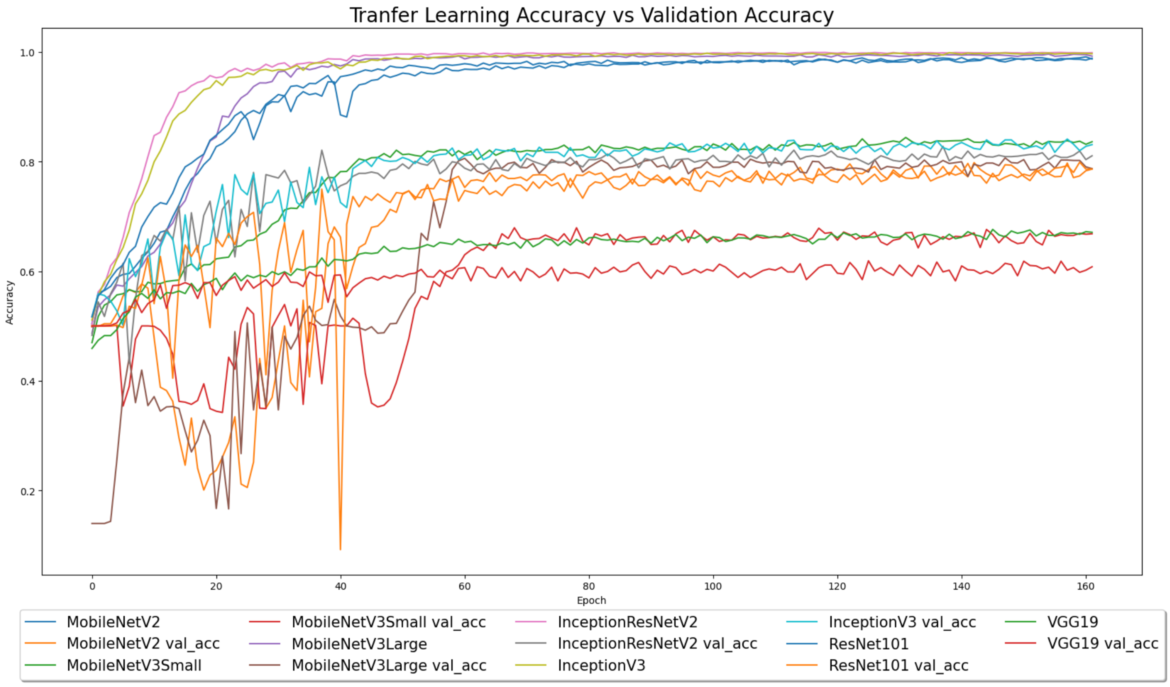


Figure 12.1: The accuracy of the different models

12.0.2 InceptionV3

For my training results, when everything was working correctly, I was able to get the highest validation accuracy of 84% and a loss lowest of 0.30.

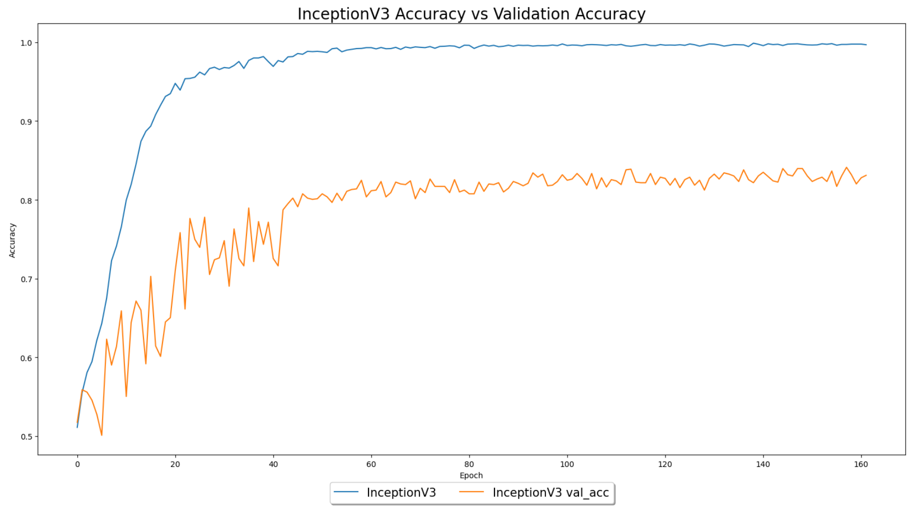


Figure 12.2: The training results using InceptionV3[34] as the base model

This graph clearly indicates that there was overfitting, as the validation accuracy is significantly lower than the training accuracy. This is due to the fact that the model is learning from the training data, and it is not generalising well to the validation data. To help mitigate overfitting, I did use data augmentation and dropout layers, however I think that the base model it's self is overfitting to the data.

I was planning on using L2 and L1 regularisation techniques work by adding a penalty to the weights of the model, which helps to reduce overfitting, however I didn't have to implement as I changed from this dataset to the OASIS-1 dataset. Hyperparameter tuning is a process of optimising the hyperparameters of a model, such as learning rate, number of layers, etc. This helps to improve the accuracy of the model and reduce overfitting.

12.0.3 MobileNetV3 Small

MobilenetV3 Small is the smallest model in size and is consistently the least accurate model and has the highest loss.

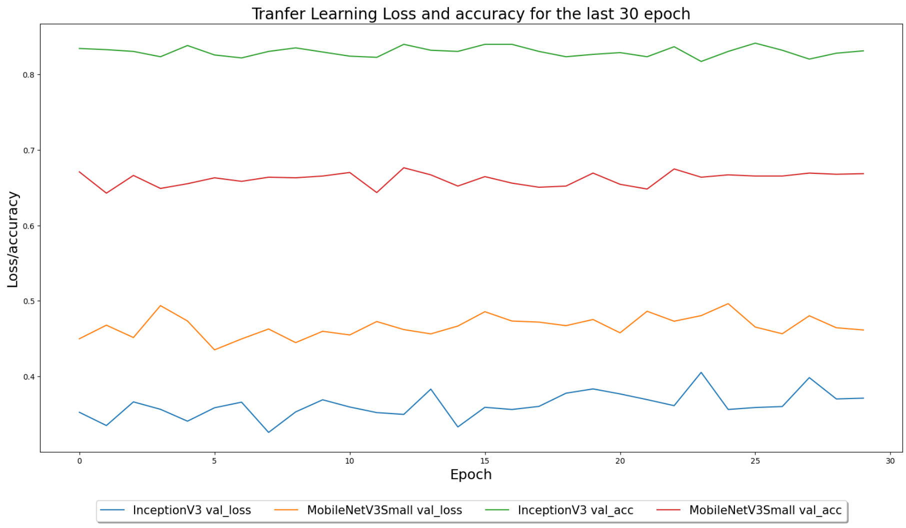


Figure 12.3: The last 30 epochs of the training results

Here's another graph that clearly shows the difference in performance between Inceptionv3 and MobilenetV3 Small. Although InceptionV3 is slightly over 2x the size of MobilenetV3 Small, the model is likely not overfitting as much as the other models are, however I can't be sure as this was trained with the old dataset.

12.0.4 The model sizes

Here's a bar chart comparing the sizes of the models in megabytes. They are uncompressed and include the full model, after transfer learning.

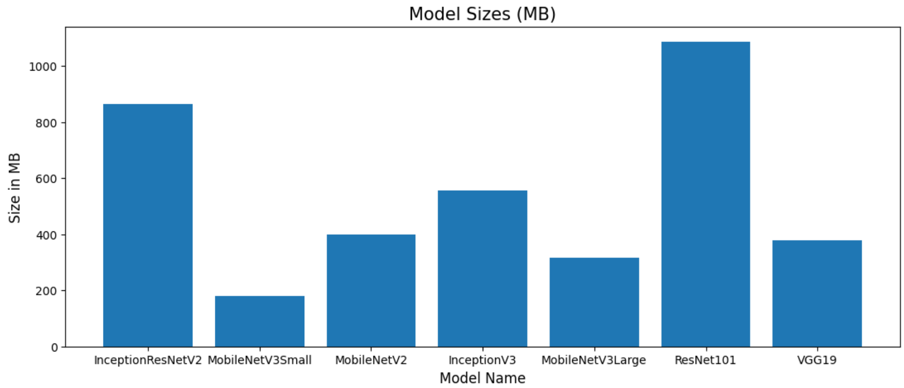


Figure 12.4: The different model sizes in MB

From this graph, it's clear to see how large some of the models are in comparison to each other.

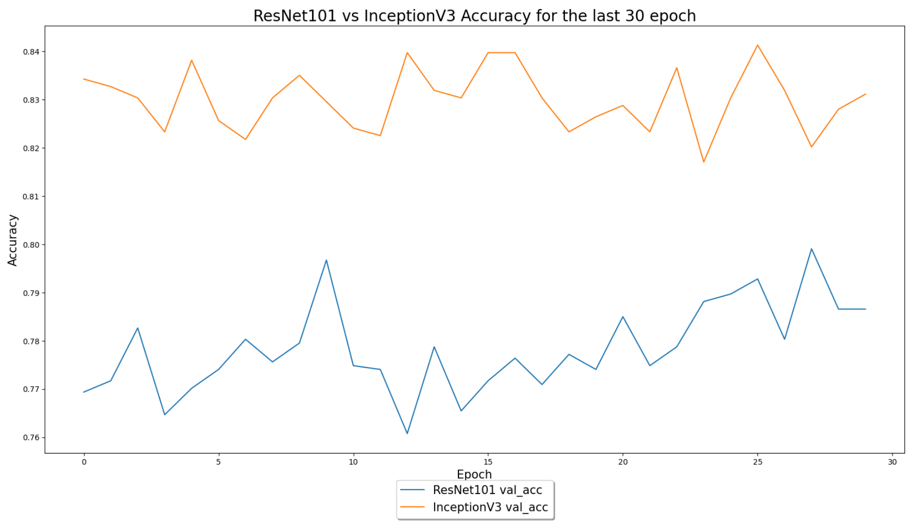


Figure 12.5: Here's it's performance in comparison to InceptionV3

The ResNet101[12] model was created by Microsoft and is a 101 layer deep neural network. It was created to compete in the ImageNet Large Scale Visual Recognition Challenge[14] in 2015. Inceptionv3 is a 48 layer deep neural network, and was created by Google to compete in the same challenge in 2015.

12.1 Why These Results Looked Promising

The results from the faulty dataset looked so promising as there was consistent progress in the accuracy of the models, and the loss was decreasing over time. This was because the model was learning from some of the validation data (but not all) so it's accuracy was never perfect (100%) on the validation data, but it was high enough to give the illusion that the model was working well.

My lack of experience with machine learning meant that I didn't know what to trust when it came to the results, I just assumed that the model was working well. Thankfully I did realise the issue before I had finished my project, and I was able to fix it with the upcoming results.

Chapter 13: The new training results

13.1 The OASIS-1 Dataset

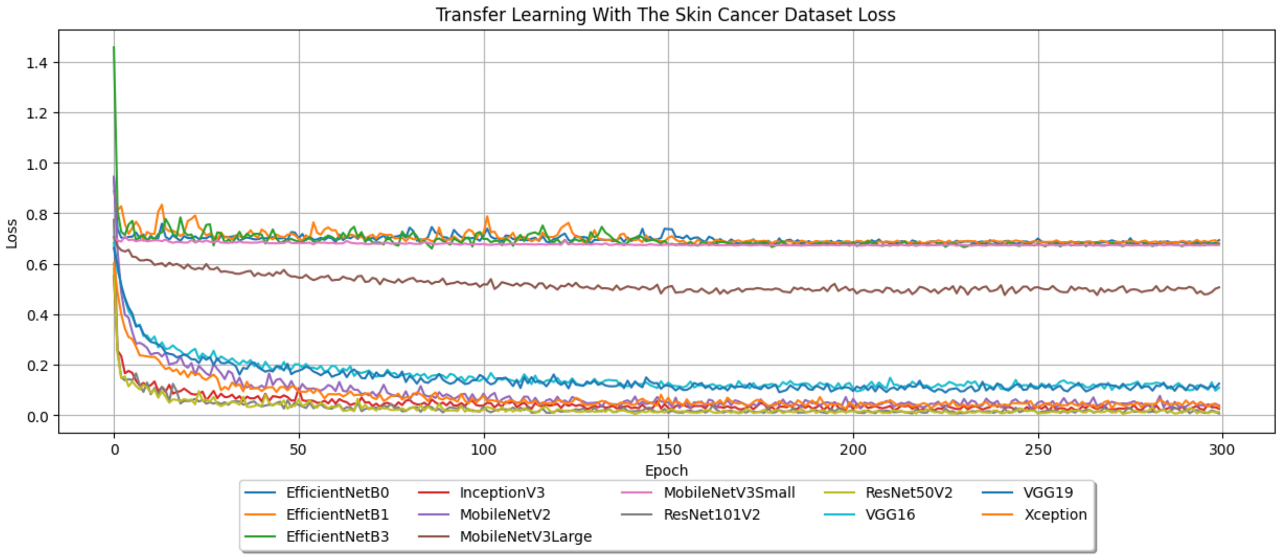
The OASIS-1 dataset accuracy is not particularly high without any optimisation, the 2D preprocessed scan images don't provide enough information for the transfer learned models to learn from and make accurate predictions.

13.1.1 Training with only the MRI scans

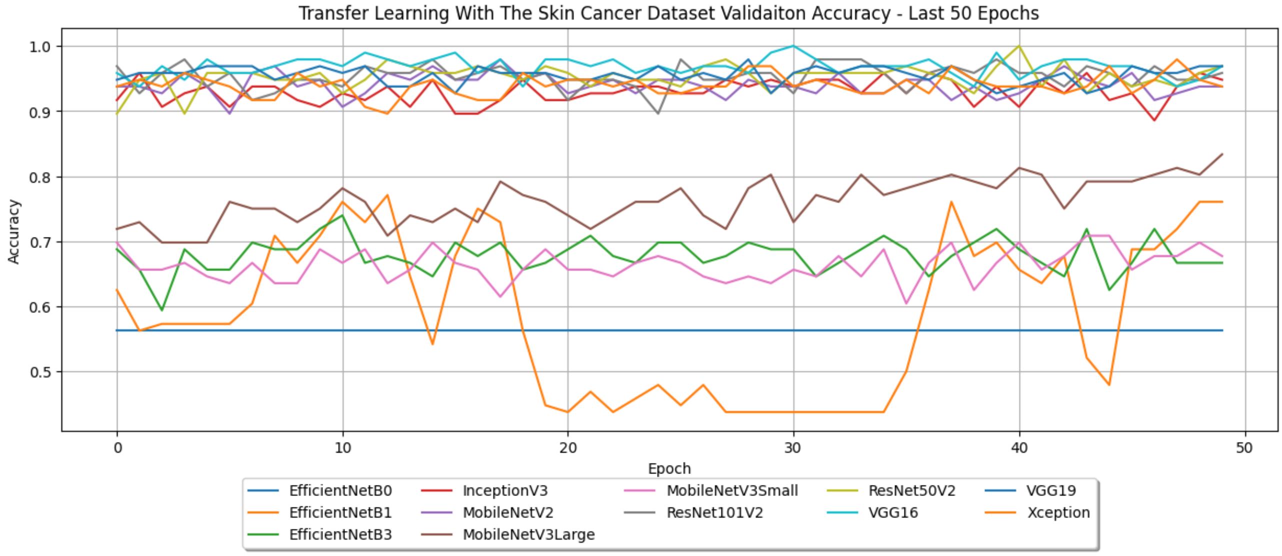
13.1.2 Training with the MRI scans and the other features

13.2 The Skin Cancer Dataset

Training the models with the skin cancer dataset was a lot more successful than training with the other dataset. This is likely due to the fact that skin cancer is visually easier to identify than Alzheimer's disease, as the skin cancer images are much more clearly defined.

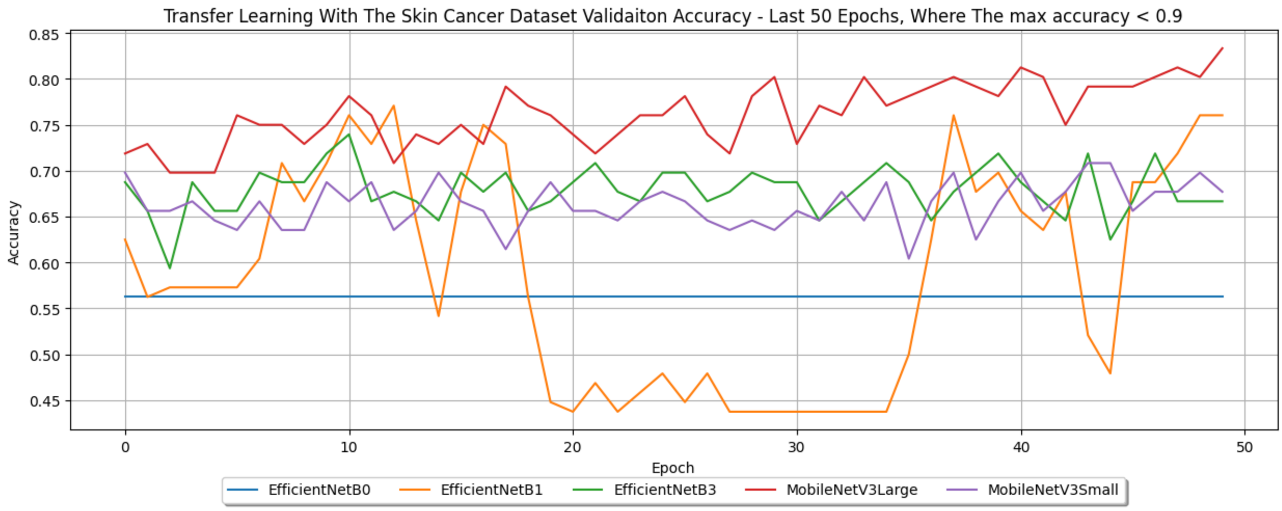


As this graph shows, there are lots of models that did well, and a handful that did poorly. This is only the loss from the training data, and not the validation data, so it's only really a good indication of what models obviously performed poorly, such as the EfficientNetB3 and the MobilenetV3 models.



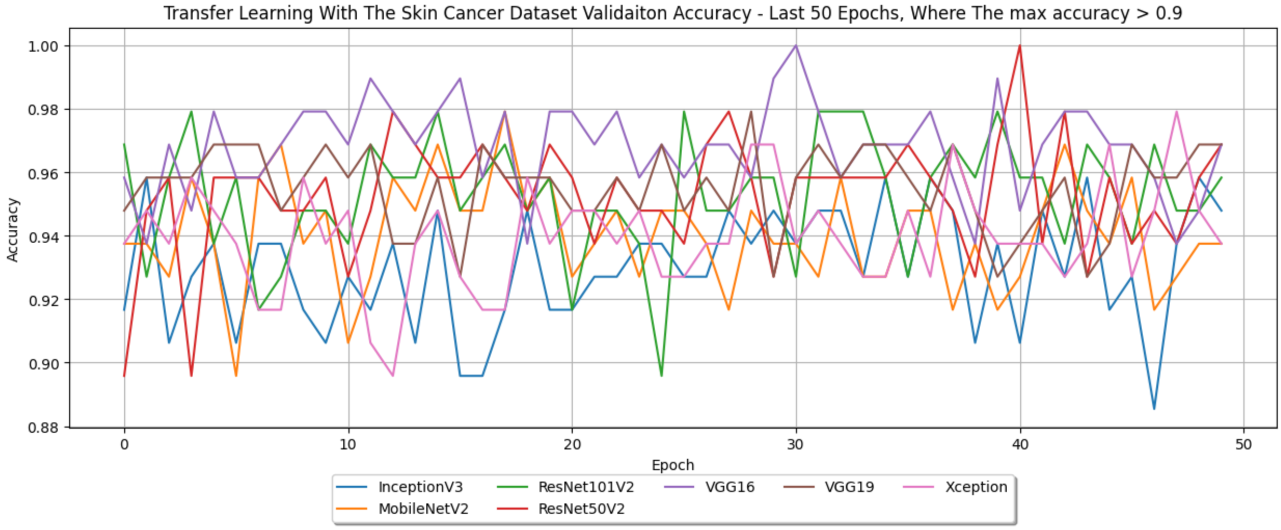
This graph shows the last 50 epochs of the training process for the skin cancer dataset, with the validation accuracy. This shows the most clear results, as the models with a high accuracy here are the ones that have learned the best from the training data. This was graph was also created with the full augmentation performed on the validation data. This may have changed the results to make the models appear to perform worse than they actually did, however lots of the augmentation I had done was reasonable and within the scope of what it would be like in the real world.

There appears to be a clearly defined line between the models that performed well and the poorly performing models.

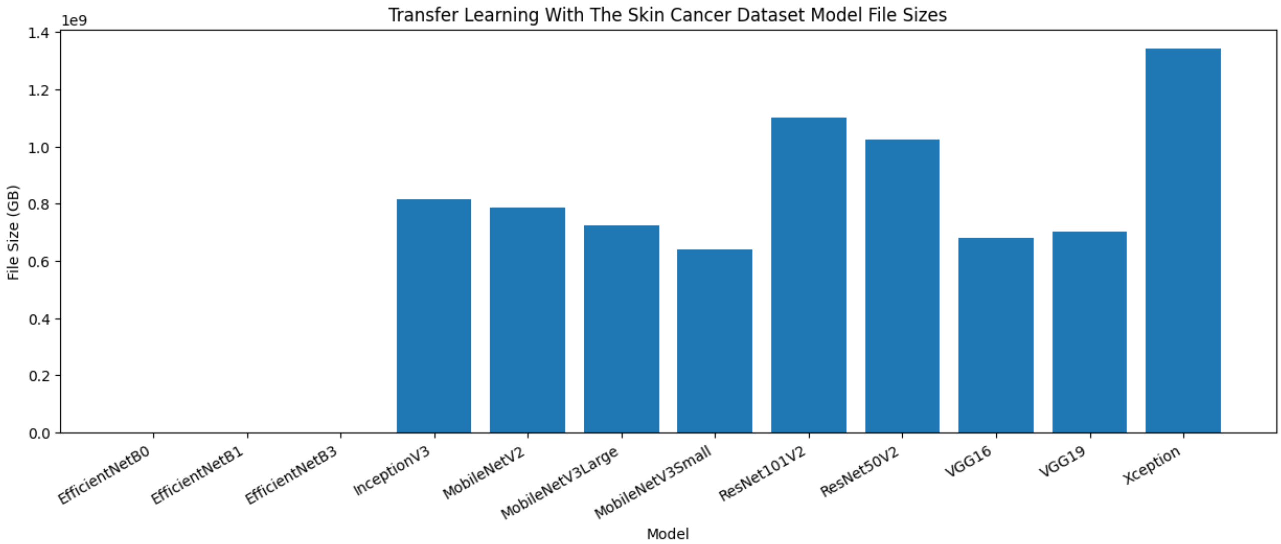


These models are the lower performers in the skin cancer dataset, none of them ever scored above 90% accuracy on the validation data, they all also were clearly distinguished from the other models in their training loss. Mobilenetv3 Large has typically been a good performer with the Alzheimer's dataset, however here it's performing comparatively poorly.

The efficient net models are also performing poorly, this is likely due to the fact that the efficient net models are comparatively very small, and may not have enough capacity to effectively capture the more complex features of the skin cancer images, as opposed to classifying images of hotdogs or not hotdogs, for example. None of these models are specifically designed to classify smaller details in images, however many of them have the capacity to do so, it just may have been pruned too much for it to be effective at this task.



These models with a maximum validation accuracy of over 90%, typically all perform very well, not dipping below the 90% mark often. As this graph shows too, many models achived a consistent accuracy of over 96% on the validation data, which is very good for the amount of time it took to train them. VGG16 typically performed the best with this dataset, and it's the smallest model that performed well. This could have been a coincidence, however it's undenyable that VGG16 is a very good model for this application.



Chapter 14: **How to run the code**

14.1 Prerequisites

Go to the directory in your terminal (with python 3.7 ¿ installed) and run the following command:

```
pip install -r requirements.txt
```

14.2 Running the code

To run the code in the notebook, follow the markdown instructions in the notebook. To run the training program, run automated-testing-training/main.py (it will likely take a long time to run).

To run the tests for the AutomatedTestingLibary run the following command:

```
python -m pytest
```

Run main.py in the AutomatedTestingLibary directory to run the training program.

The flower training notebook does not work as I am having issues with Keras and the dataset (specifically the way it is loaded).

Chapter 15: **Professional Issues**

15.1 Usability

As AI-based systems such as transfer learning models are increasingly employed for Alzheimer's detection, it is crucial to ensure that they are accessible, user-friendly and transparent in their operation. The algorithms should be focused on designing systems that complement human expertise rather than replacing it, facilitating seamless integration between healthcare professionals and AI tools. Appropriate training and guidelines for the use of AI in medical imaging should be implemented to ensure that healthcare providers are comfortable and proficient with the technology. This would mean that my web interface should be intuitive and easily accessible to healthcare professionals, and should be able to be used by them without any training.

15.2 Plagiarism

To prevent unethical practices and promote collaboration, it is essential to establish guidelines for the appropriate use, citation, and acknowledgment of research materials and software. When using code or datasets from other sources, I have quoted the source and provided a link or citation to the original work.

15.3 Safety and Reliability

The validity and reliability of AI models are critical for maintaining trust in the technology, especially when dealing with sensitive healthcare data. Rigorous testing and validation methods should be employed to ensure the accuracy of the model. Furthermore, healthcare providers and patients should be made aware of any "as is" clauses or other limitations associated with the software to prevent misuse or over-reliance on the technology. As the accuracy of the models is not guaranteed, I will need to ensure that the output is clearly displayed from the model and the confidence scores for each class are also displayed, to help the healthcare professional make a decision. The models used in this project would most likely be better suited for screening patients, rather than making a final diagnosis.

15.4 Privacy and Legal Issues

The use of machine learning for Alzheimer's diagnosis raises significant privacy concerns, particularly with regard to the handling and storage of sensitive patient data. Data protection laws and regulations, such as HIPAA and GDPR, must be adhered to in order to guarantee patient confidentiality and to secure informed consent for the collection of medical data. Data anonymization techniques should be used to protect patients' identities when training and utilizing AI models. For the OASIS datasets, the data is already anonymized, but there are also other terms and conditions that must be followed, for example, with MRI scans, you can generate an image of the patient's face and use that to identify the patient and that was explicitly forbidden in the terms and conditions.

Machine learning is also used in other areas of healthcare, such as drug discovery and clinical trials, for example, a drug may get killed off in it's early stages if an algorithm determines that it could be carcinogenic. This could be a problem if the algorithm is wrong, as the drug could have been a significant breakthrough treatment, however it's likely to have a better impact to use the prediction algorithm to screen drugs and get the problematic ones removed quicker, as more resources can be put into the drugs that are more likely to be successful.[18]

15.5 Monopoly and Competition

The AI healthcare market should remain competitive and accessible to various stakeholders, preventing the dominance of a few companies or technologies. The avoidance of proprietary formats, DRM measures, and forced tie-ins is crucial for maintaining a fair marketplace and providing healthcare professionals with multiple options for AI-assisted Alzheimer's diagnosis and treatment. Specifically, the BIDS format should be used to store and interpret the data, as it is an open standard for neuroimaging data. In terms of competition, Keras, Tensorflow and the pre-trained models are all open source, so anyone can use them, and there are many other open source libraries that can be used by anyone, so the competition is not a huge issue.

15.6 Management and Stakeholder Consultation

There are various stakeholders who would be interested in AI prediction of Alzheimer's, for example the patients could be interested in knowing if they have Alzheimer's quicker than they otherwise would have, however the patients having direct access to the web interface would be problematic, as the results may be inaccurate and could potentially cause un-needed stress for the patient.

The healthcare professionals would be interested in the web interface, as it would allow them to quickly and easily get a prediction of whether the patient has Alzheimer's or not, and it would also allow them to see the confidence scores for each class, so they can make a more informed decision, this interface being available 24/7 and being easy to use would benefit them greatly. The usability to the healthcare professionals would depend on the accuracy of the model too, so it would be important to ensure that the model is as accurate as possible.

15.7 Biases in the data

The biases in the OASIS-1 dataset are very prevalent, I have decided to dedicate a seperate chapter to this issue. Giving the machine learning model the age of the patient as a feature could potentially be a bad idea, as the biases in the data could potentially lead to the model predicting whether or not the patient has AD, based off of their age. If I don't give the model the age of the patient, there is a chance that it could predict the age it's self, which would defeat the purpose of the model. I could try predicting the age of the patient to see how well it can guess the age of a patient based off of their MRI scan, to see if it has the ability to predict the age of the patient, I could also see how changing the given age of the patient affects the accuracy of the model, accross the different classes.

15.8 Collaboration and Communication Barriers

If the model's accuracy is not explicitly stated in the web interface, doctors may be more likely to rely on the models predictions, rather than using their own judgement, which could lead to a misdiagnosis. The model's accuracy should be clearly stated in the web interface, so that the healthcare professionals can make an informed decision, and the interface should have a disclaimer stating that the model's accuracy is not guaranteed, and that the healthcare professional should use their own judgement when making a diagnosis, along with multilingual support, so that the interface can be used by healthcare professionals from all over the world, without any potential problems with language barriers.

15.9 Bias in the Skin Cancer Dataset

The skin cancer dataset has a variety of different photos of benign and malignant blemishes on skin, however all of the images are of caucasian people, with there appearing to be no variety in terms of skin colour. This means that this model would only benefit anyone of caucasian descent, and it would not be reliable with any other skin colour. This is a problem as the dataset is not representative of the world's population, which could lead to non-caucasian people getting a lower quality of healthcare, as they would not be able to use the model to get a diagnosis of their skin cancer. On the other hand, prediction of skin cancer would be a lot more accurate for caucasian people as opposed to people with darker skin, as the melanin in their skin would make it harder to see the blemishes and make out what they are. This specific problem would be beyond the scope of this report, however it would be something to consider for anyone using this dataset for other projects.

Chapter 16: **Diary**

19/10/2022

Today I have been deciding the specific classes of images I want to use for my project. I had initially intended on using only 1 class, however I have decided to use 2 classes instead as I think it will be more interesting to see the results of the transfer learning on 2 classes rather than 1, It will also demonstrate better the different strengths and weaknesses of the individual transfer learning methods.

I have decided to use the following sets of images:

- Flowers Dataset This dataset is likely to be an easier dataset for the models to get high accuracy on, as the different flowers are quite distinct from each other.
- Alzheimer's Dataset This dataset is going to be more difficult for the models as the changes are more subtle between the different scan images, and the images are not as distinct from each other.

I am going to follow the tutorial here which should help me to get a good understanding of how to use the datasets and how to train the models to start with. I have no prior experience training models so this will be a good starting point for me. I will then need to use parts of the code from the tutorial to transfer run the transfer learning.

26/10/2022

Today I have been working on the transfer learning tutorial, I've made more progress through it and I've managed to get Mobilenetv3 to run on the flowers dataset, however I'm having some issues with the alzheimer's dataset. This is significant progress as this was a major blocker for me as I was not sure how to go forward with the prediction.

01/11/2022

Today I've worked more on the transfer learning tutorial and it can now identify the different flowers in the flowers dataset, with varying degrees of accuracy, around 80% was the best I got however this was only after 5 epochs and was the first time I have tried transfer learning.

09/11/2022

I had lost some of my git history for my diary, so I have had to rewrite some diary entries. I have found more flower dataset images here and I've decided to change over to this as soon as possible; https://www.kaggle.com/competitions/tpu-getting-started/overview

14/11/2022

I am training the model "Mobilenetv3 Small" with the Alzheimer's dataset, I have been able to get it to run and it is currently training. The power required to train the models is making it difficult to use trial and error for the parameters, so I am going to try and find a way to use the GPU on my laptop to train the models as soon as possible. I'm also going to optimise the way I am training the models as to not use more power than necessary. This means I've got to get a better understanding of the models I am training with and how they work. I have found this paper Searching For MobileNetV3[13] which I think will be useful for me to read through and understand the structure of Mobilenetv3.

15/11/2022

I've had trouble getting the Alzheimer's classification to work with transfer learning, so I decided to try and run other people's code to see if I could get it to work. I have found that my transfer learning could be significantly more efficient and accurate if I use the existing code, so I'll have to try and identify where I went wrong with my code and make improvements.

21/11/2022

Today I managed to get the accuracy up to 90% on the alzheimer's dataset, this is a significant improvement from the 60% I was getting before. My model appears to still be overfitting however this progress is good. This was done through transfer learning on the Mobilenetv3 small model.

Output from the model during fine-tuning:

```
loss: 0.0314 - accuracy: 0.9883 - val_loss: 0.1337 - val_accuracy: 0.9036
```

22/11/2022

Here's some graphs of the model's predictions on the alzheimer's dataset: This graph was made using the MobilenetV3Small model and transfer learning on the Alzheimer's dataset, with all but 3 layers of the mobilenetv3 model frozen. `![image](/images/better-output.png)`

Here is a graph of where I was fine-tuning the model however I had overfitted the model and it was not generalising well to the test data:

`![image](./images/maxing-out.png)`

I have also setup my laptop properly using Manjaro Linux to train the models on the GPU, this should make it much easier to train the models and to try different parameters and should allow me to create a confusion matrix for different parameters.

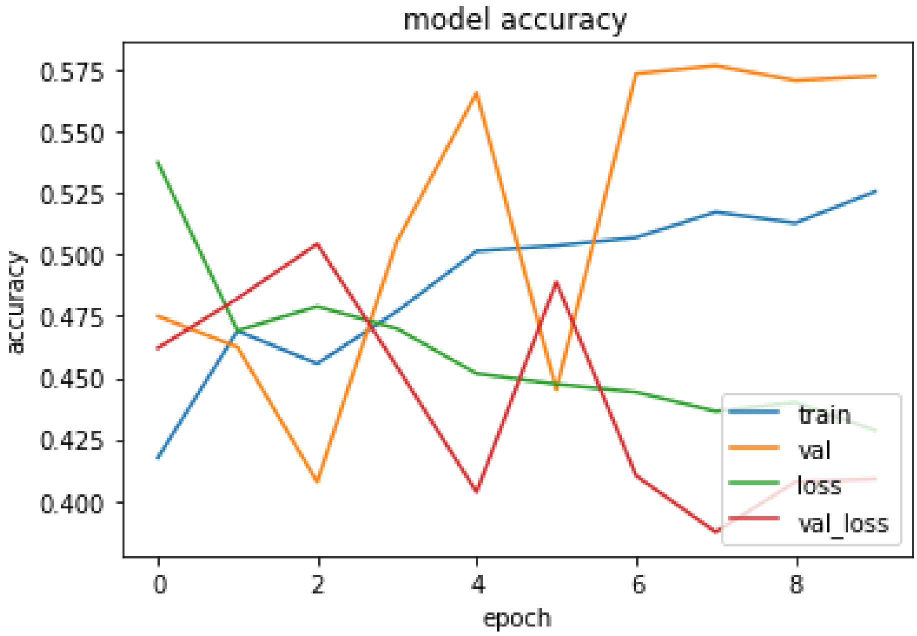


Figure 16.1: Graph of the model's predictions on the alzheimer's dataset

23/11/2022

I have achieved 99% accuracy on my model identifying Alzheimer's (validation dataset). This was unfortunatley not saved however as it crashed, however I believe I can get it to that level again. I have written a scritpt to use different base models for transfer learning and to output the results of training to a JSON file so I can analyse the results later, it also saves the model to a file so I can use it later. This should make it a lot easier to use different models, datasets and parameters to train the models and to analyse the results and make the confusion matrix.

27/11/2022

I have been working lots on the interim report and getting it ready to submit, during this time I have also been setting my laptop off to train the models as I work. I discovered that my '99%' accuracy was from the model learning the validation set through the augmented images, so I have had to retrain all the models with images I have augmented myself.

Soon I'll finish the Alzheimer's dataset training and I'll start working on the flowers dataset.

28/11/2022

Today I have had trouble with the training, my GPU kept running out of memory due to me not releasing the ram after the keras model was trained. This meant I had to keep on checking and restarting the training, every few models. This is now fixed and I have been able to train the models without any issues. They are currently training and there is a lot of progress to be made.

```
MobileNetV3Small = keras.applications.MobileNetV3Large(
    input_shape=(224, 224, 3),
    include_top=False,
    weights='imagenet')

# Create the model

model = keras.Sequential()

for layer in MobileNetV3Small.layers:
    layer.trainable = False

model.add(MobileNetV3Small)
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(1024, activation='relu'))
model.add(keras.layers.Dropout(0.2))
model.add(keras.layers.Dense(4, activation='sigmoid'))

✓ 1.9s

# Compile the model

model.compile(
    optimizer=keras.optimizers.Adam(learning_rate=0.001),
    metrics=['accuracy'],
    loss='binary_crossentropy',
)

# Train the model
history = model.fit(
    train_ds,
    epochs=10,
    validation_data=test_ds,
)

✓ 11m 32.7s
```

Figure 16.2: Code used to generate the graph

29/11/2022

Today I have decided to start work on the training and testing library which I can use to speed up training and increase reliability of my training, through unit tests and consistency. This library will allow me to adjust parameters for fitting and compilation to let me easily do hyperparameter tuning.

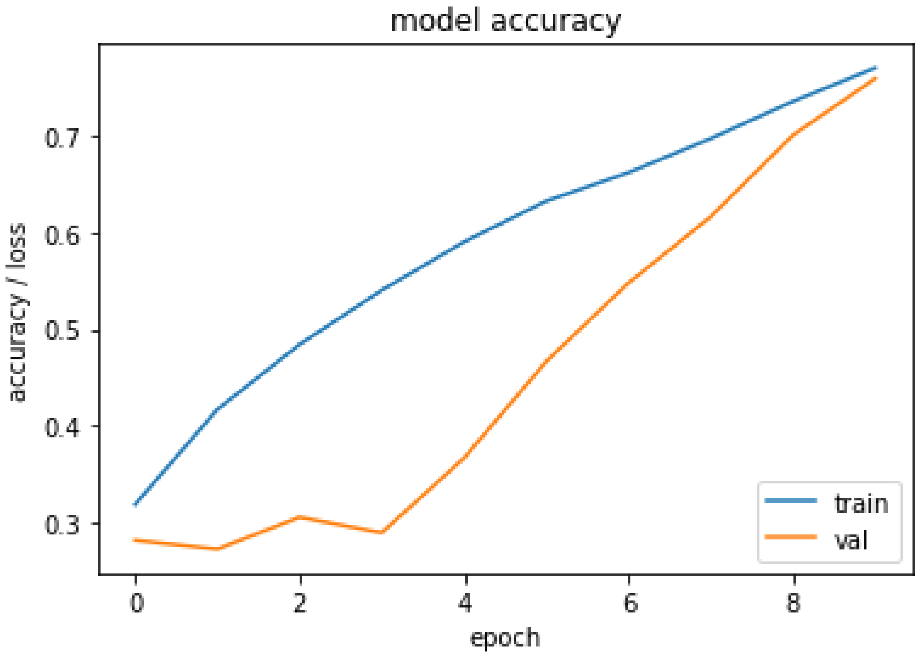


Figure 16.3: Positive progress from unfreezing all the layers of the model and ”fine-tuning” the model

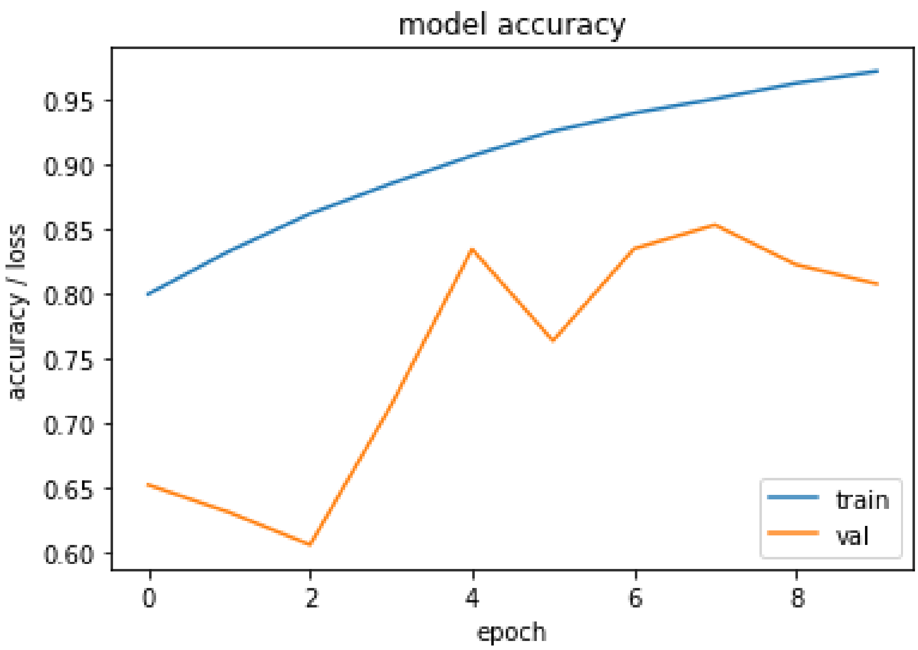


Figure 16.4: Graph of where I was fine-tuning the model however I had overfitted the model and it was not generalising well to the test data

1/12/2022

I have been working lots on my report and the training library, this has taken a while to get all the information I feel I need in my report, however I have now mostly finished it and will submit it soon.

I have made lots of graphs from the history of the models in training, this made it a lot easier

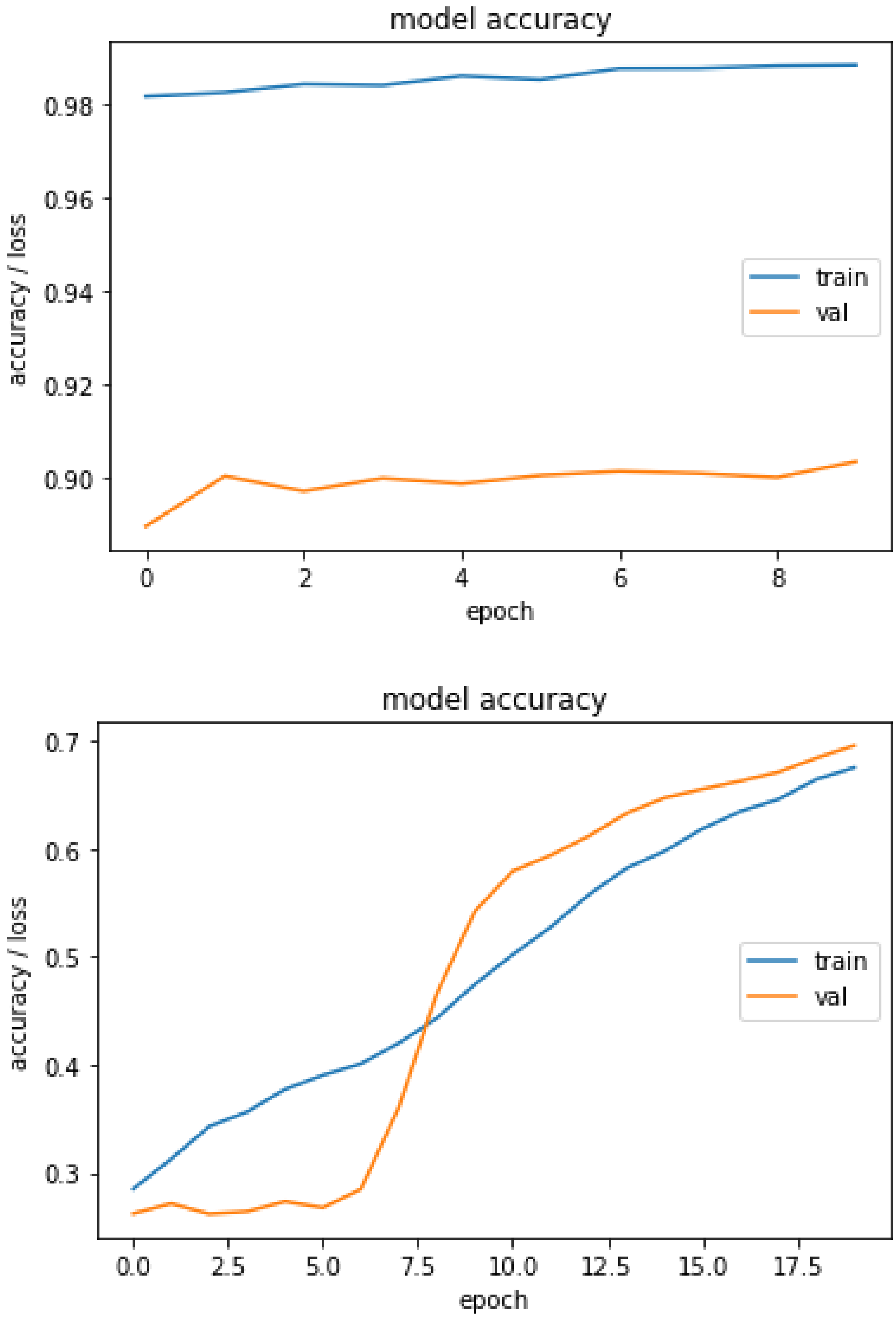


Figure 16.5: First attempt at training the mobilenetV3Large model after unfreezing all the layers

to talk about the different strengths and weaknesses of the different base models.

Chapter 17: **Conclusion**

Bibliography

[1] Mayank Agarwal and Javed Mostafa. "Content-based image retrieval for Alzheimer's disease detection". In: *2011 9th International Workshop on Content-Based Multimedia Indexing (CBMI)*. 2011, pp. 13–18. DOI: 10.1109/CBMI.2011.5972513.

[2] *Alzheimer's Dataset*. URL: https://www.kaggle.com/datasets/uraninjo/augmented-alzheimer-mri-dataset.

[3] *Alzheimer's Disease Statistics*. URL: https://dementiastatistics.org/.

[4] Lucia Ballerini et al. "A Color and Texture Based Hierarchical K-NN Approach to the Classification of Non-melanoma Skin Lesions". In: *Color Medical Image Analysis*. Ed. by M. Emre Celebi and Gerald Schaefer. Dordrecht: Springer Netherlands, 2013, pp. 63–86. ISBN: 978-94-007-5389-1. DOI: 10.1007/978-94-007-5389-1_4. URL: https://doi.org/10.1007/978-94-007-5389-1_4.

[5] *BIDS MRI Metadata*. URL: https://bids-specification.readthedocs.io/en/stable/04-modality-specific-files/01-magnetic-resonance-imaging-data.html.

[6] *CommitLint*. URL: https://github.com/conventional-changelog/commitlint.

[7] *Confusion Matrix*. URL: https://towardsdatascience.com/confusion-matrix-for-your-multi-class-machine-learning-model-ff9aa3bf7826.

[8] *Content-Based Image Retrieval Diagram*. URL: https://en.wikipedia.org/wiki/Content-based_image_retrieval.

[9] *CUDA Toolkit Documentation*. URL: https://docs.nvidia.com/cuda/.

[10] Christopher H van Dyck et al. "Lecanemab in Early Alzheimer's Disease". en. In: *N Engl J Med* 388.1 (Nov. 2022), pp. 9–21.

[11] Krzysztof J. Gorgolewski et al. "The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments". In: *Scientific Data* 3.1 (2016), p. 160044. ISSN: 2052-4463. DOI: 10.1038/sdata.2016.44. URL: https://doi.org/10.1038/sdata.2016.44.

[12] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[13] Andrew Howard et al. "Searching for MobileNetV3". In: *CoRR* abs/1905.02244 (2019). arXiv: 1905.02244. URL: http://arxiv.org/abs/1905.02244.

[14] *ILSVRC*. URL: http://www.image-net.org/challenges/LSVRC/.

[15] *ImageNet*. URL: http://www.image-net.org/.

[16] *JupyterLab is a web-based interactive development environment for Jupyter notebooks, code, and data.* URL: https://jupyter.org/.

[17] *Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano.* URL: https://keras.io/.

[18] Ting Li et al. "DeepCarc: Deep Learning-Powered Carcinogenicity Prediction Using Model-Level Representation". In: *Frontiers in Artificial Intelligence* 4 (2021). ISSN: 2624-8212. DOI: 10.3389/frai.2021.757780. URL: https://www.frontiersin.org/articles/10.3389/frai.2021.757780.

[19] *Manjaro Linux*. URL: https://manjaro.org/.

[20] *Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python.* URL: https://matplotlib.org/.

[21] Teresa Mendonça et al. "PH2 - A dermoscopic image database for research and benchmarking". In: *2013 35th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2013, pp. 5437–5440. DOI: 10.1109/EMBC.2013.6610779.

[22] *Neural Network Diagram*. URL: https://www.tibco.com/reference-center/what-is-a-neural-network.

[23] *NVIDIA GeForce GTX 1070*. URL: https://www.nvidia.com/en-gb/geforce/products/10series/geforce-gtx-1070/.

[24] *OASIS*. URL: https://oasis-brains.org/#data.

[25] *OASIS-1 Fact Sheet*. URL: https://oasis-brains.org/files/oasis_cross-sectional_facts.pdf.

[26] *Pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.* URL: https://pandas.pydata.org/.

[27] *PyTest*. URL: https://docs.pytest.org/en/stable/.

[28] *Python*. URL: https://www.python.org/.

[29] *React: A JavaScript library for building user interfaces*. URL: https://reactjs.org/.

[30] *RHUL GitLab*. URL: https://gitlab.cim.rhul.ac.uk/.

[31] *RTX 3080*. URL: https://www.nvidia.com/en-us/geforce/graphics-cards/30-series/rtx-3080/.

[32] Jorge Samper-González et al. "Reproducible evaluation of classification methods in Alzheimer's disease: Framework and application to MRI and PET data". In: *NeuroImage* 183 (2018), pp. 504–521. ISSN: 1053-8119. DOI: https://doi.org/10.1016/j.neuroimage.2018.08.042. URL: https://www.sciencedirect.com/science/article/pii/S1053811918307407.

[33] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *International Conference on Learning Representations*. 2015.

[34] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *CoRR* abs/1512.00567 (2015). arXiv: 1512.00567. URL: http://arxiv.org/abs/1512.00567.

[35] Mingxing Tan and Quoc V. Le. *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*. 2020. arXiv: 1905.11946 [cs.LG].

[36] Teck Yan Tan, Li Zhang, and Chee Peng Lim. "Intelligent skin cancer diagnosis using improved particle swarm optimization and deep learning models". In: *Applied Soft Computing* 84 (2019), p. 105725. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2019.105725. URL: https://www.sciencedirect.com/science/article/pii/S156849461930506X.

[37] *tmux*. URL: https://github.com/tmux/tmux/wiki.

[38] Yudthaphon Vichianin et al. "Accuracy of Support-Vector Machines for Diagnosis of Alzheimer's Disease, Using Volume of Brain Obtained by Structural MRI at Siriraj Hospital". In: *Frontiers in Neurology* 12 (2021). ISSN: 1664-2295. DOI: 10.3389/fneur.2021.640696. URL: https://www.frontiersin.org/articles/10.3389/fneur.2021.640696.

[39] *VsCode*. URL: https://en.wikipedia.org/wiki/Visual_Studio_Code.