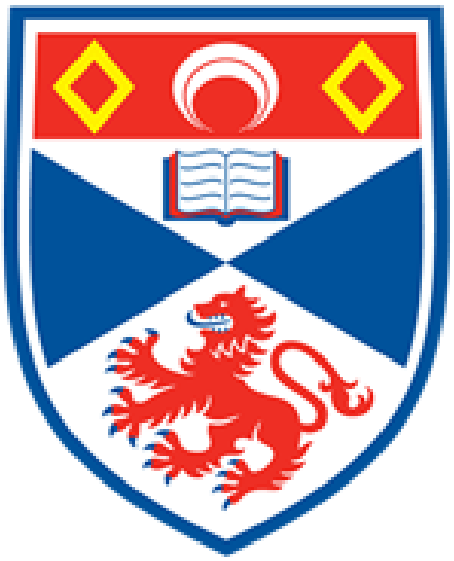
Combining Many Datasets to Classify Medical Images by Modality

Craig Macfadyen

160010069



University of St Andrews

January 8, 2021

Abstract

The aim of this dissertation is to use deep learning to predict the modality of a medical imaging scan. Current deep learning research is focused on creating the optimal model for a classification task on a single disease using one dataset. This project aims to show that datasets from many sources across many locations can be combined and used to train a deep learning model. The modality of an image is always known so this was chosen as a target for the model. 102 medical imaging datasets were combined to create a dataset with a total of 4.5 million images. These images cover at least 14 different locations of the body.

This dataset was used to train a deep convolutional neural network which achieved a classification accuracy of 96%. Multiple deep networks were compared on the dataset and different regularisation strategies were attempted. Even with the care taken to ensure a balanced spread of datasets in the train-validate-test split the models suffered greatly from overfitting. The research shows it is possible to combine data from many sources and use it to train a deep learning model. The research also identified many pitfalls when working with such a large dataset and suggests ways of mitigating these problems for future researchers.

Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main text of this project report is 15,832 words long, including project specification and plan. In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

Contents

Chapter 1

Introduction

Current research in deep learning for medical imaging is highly specific. In most papers, a model is trained to recognise a single disease in one area of the body, only using one kind of medical image. This has proved effective and many models have shown the same level of performance as clinicians in the tasks they were trained on. The problem is it would be very difficult to integrate all of these different models into a clinician's workflow, because there needs to be one model for every disease for every kind of medical image. The modality of a scan is the imaging method used to capture that scan. Figure 1.1 has some examples of images in different modalities. The modality of a scan is always known, so a system that classifies images by their modality would not be useful to radiologists. However, if a model was trained on multiple medical imaging datasets in multiple modalities then the model might be able to predict the presence of disease across multiple organs, disease types and modalities. A system like this would be approaching the capabilities of a radiologist. In recent years many open source medical imaging datasets have become available, but there been very little research that has uses a combination of datasets to train a model. Predicting the modality of the scan is the simplest task to solve with a large combination of datasets as this label is always known. Training with many datasets is desirable because:

1. Many tasks in radiology are related so isolating them into small individual problems doesn't make sense.
2. It would be much easier for a radiologist to use a system that could take any scan and make a prediction about the presence of disease.

This project will serve as a demonstration that networks can be trained over multiple datasets and will hopefully lay the foundation for future research to create systems that would be useful to a radiologist in the real world.

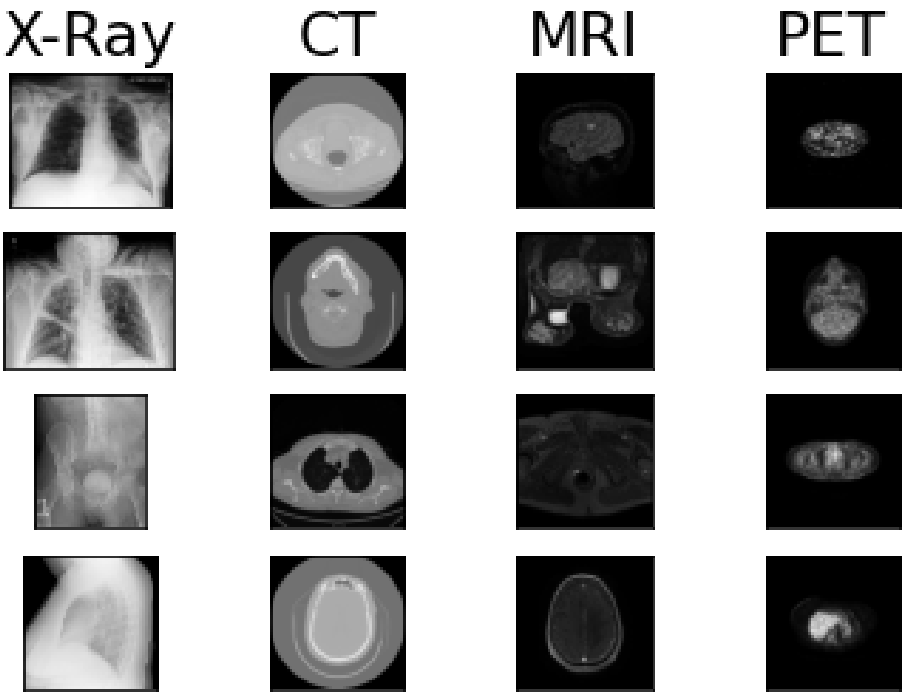


Figure 1.1: Visualisation of a spread of images from different locations in different modalities. Different modalities use different kinds of radiation, and these are absorbed to varying degrees by tissue in the human body. This leads to the same tissue looking different in each modality. Sources (left to right, top to bottom): [1, 2, 3, 4] [5, 6, 7, 3] [8, 9, 10, 11] [12, 3, 13, 14]

1.1 Objectives

The main objective of this project is to combine many medical imaging datasets into one large dataset, and then to use this dataset to train a deep learning model. Before doing these tasks a literature review needs to be created to motivate the project and to allow discussion of where the project fits into current research.

1.1.1 Primary Objectives

1. Write a literature review to understand previous research in the area of deep learning for medical imaging.

2. Use the knowledge gained writing the literature review to select and train the best

model for this task.

3. Document and evaluate the performance of the model on unseen data.

1.1.2 Secondary Objectives

1. Experiment with different models and architectures to find a model that can achieve high accuracy.
2. Experiment with different hyperparameters to increase the performance of a selected model.

1.1.3 Tertiary Objectives

If the primary objectives are completed quickly it would be possible to use the large dataset for other tasks.

1. Classifying what organ/area of the body is present in a medical scan of any modality.

1.2 Objectives Completed

All the primary objectives and secondary objectives were completed. 102 datasets were combined to create a dataset with 4.5 million images and a deep convolutional neural network was trained and evaluated on this dataset. Multiple models were experimented with and one model was selected for further tuning to increase performance, which was unsuccessful. The time it took to train a model using this many images meant that development took a long time and the tertiary objectives were not attempted.

Chapter 2

Context Survey

2.1 Introduction

Medical imaging is the process of capturing and creating visualisations of the interior of the body. These visualisations can be used by a doctor to gain an understanding of what is happening in the patient's body to aid with diagnosis and treatment. A radiologist can tell the modality of a scan is simply by inspecting the scan because tissue in the human body is presented differently for each scan modality. These visual differences could be used by a deep learning model to predict the modality of a medical imaging scan. Deep learning models have been shown to be effective in the field of image classification but this has not been applied to predicting the modality of a scan. Such a model would be interesting for two reasons:

1. To see if a deep learning model can predict the modality of a scan.
2. To demonstrate training a model across a combination of many independent medical imaging datasets.

The following review of literature will discuss deep learning, how deep learning is relevant to medical imaging and where this project will fit into current research.

2.2 Medical Imaging

Medical images allow radiologists to get a visualisation of the patients body. These can be used to inform a diagnosis about the presence of disease, and the treatment of the patient. Unfortunately, there is a growing shortage of radiologists, especially in countries with developing healthcare systems [15]. The shortage is due to the fact that radiologists need many years of training. In addition, interpreting medical images is difficult and takes time. Recent advancements in deep learning techniques show progress towards creating systems that can ease the workload of radiologists which can lessen the effect of this shortage.

2.3 Deep Learning

In the last decade there has been a large improvement in the performance of computer vision models using deep learning. Deep learning tasks in the field of computer vision are often classification tasks. A dataset is created where every image has a label that specifies the class that image belongs to. The model is trained on this dataset and learns the features of an image that indicate what class that image belongs to. When a trained deep learning model is shown new data, it is able to make a prediction about what class this previously unseen image belongs to. The following sections will outline how deep neural networks learn to classify images, and what applications deep networks have for medical imaging.

2.3.1 Supervised Learning

Deep learning is a form of supervised learning which means it requires labelled input data to train the model. Supervised learning is the process of taking labelled input data and fitting a model to predict the label based on the input. More formally, it is the process of learning a function:

$$\hat{Y} = f(X, \theta) \tag{2.1}$$

$$Y = f(X, \theta) + \epsilon \tag{2.2}$$

where X is the labelled input sample, θ is the model parameters, $\hat{Y}$ is the predicted label for the sample, Y is the actual label and ϵ is the error. The error is way of quantifying how 'wrong' a prediction is. The parameters of the model, θ, can be changed to reduce the size of the error the model generates. The learning process is finding the set of parameters

that leads to the smallest error value over the training data. These parameters are learned using an optimisation strategy called gradient descent (or variations of it).

2.3.2 Gradient Descent

Gradient descent is an iterative optimisation algorithm to find the minima of a differentiable function. In the case of deep learning the goal is to minimise the amount of error, so a differentiable function can be used to quantify the error. This function is called the loss function. Gradient descent can be used to search for the minima of the loss function, which will lead to a model that makes correct predictions more often (provided the loss function is sensible).

A common analogy for gradient descent is descending a hill blindfolded. A strategy to quickly descend would be to feel what part of the slope is the steepest then take one step in that direction. When the ground under your feet is flat you have reached a local minimum. Gradient descent optimisation is the process of finding the derivative (gradient) of a loss function relative to some parameters (θ), then tweaking the parameters to take a step down the "slope" in the direction of decreasing gradient. For every iteration of gradient descent, the model is given a sample to predict on, the loss is calculated from the predicted and actual labels, then for every parameter θ_i in θ, the partial derivative of the loss function with respect to θ_i is found. The model's parameters can then be updated in the opposite direction of the partial derivative, which is taking one step down the slope.

$$\theta_i \leftarrow \theta_i - \alpha \frac{\delta}{\delta \theta_i} L(\theta) \tag{2.3}$$

The "$\leftarrow$" operator is intended to mean assignment in this case, so θ_i is updated. α is called the *learning rate*, and it dictates the size of the update to the parameters. The learning rate is set before training. If the learning rate is too high, the model will not converge because it keeps stepping over the minima. If the learning rate is too low, the model will converge eventually but it could take a very long time to do so. There is no way of knowing the correct learning rate for a problem before training so generally this takes some manual optimisation to get right. Another option is to use an adaptive learning rate. If the loss plateaus after in the training process, the learning rate is likely too high. Reducing the learning rate at this point allows the optimiser to descend into the local minimum and the model can converge. Parameters of the model that affect the learning process in this way but are not learned themselves are called *hyperparameters* and there are other important hyperparameters that will be touched on later in this review. Gradient descent is "finished" when the gradient is very close to zero. At this stage the parameters of the model are the ones that led to the least amount of error on the training set.

A common addition to gradient descent is *momentum*. The idea behind momentum is to keep the gradient moving in roughly the same direction. For the intuition behind this, picture a ball rolling down a hill. The ball builds momentum as it travels down the hill and this allows it to move over flat sections on the descent until eventually it reaches a hill it cannot climb over and it slowly rolls backwards and forwards until it comes to rest. In this analogy the ball rolling down the hill is the gradient descent optimiser and momentum is simulated by adding a fraction of the last weight update to the current weight update. Momentum can allow the optimiser to travel over local minima because even if the gradient at a point is zero, there will still be some momentum left from the previous weight update.

The benefit of gradient descent is that for optimisation problems with many input features the calculations can be performed in a reasonable amount of time. However, there are some problems associated with gradient descent. Complex problems can have non-convex loss functions, which means there will be local minima for the algorithm to get stuck in. The loss function has to be differentiable which means not all loss functions can be used.

2.3.3 Biological Neural Networks

Animal brains are made up of many neurons. These neurons have connections from other neurons and can receive electrical signals along these connections. If a neuron receives enough of these signals in a short period it will send out a signal to the neurons that are connected to it. While this behaviour is quite simple, complex computations can be performed by signals propagating through these neural networks. The exact operation of biological neural networks is still being researched but this simplified model is enough to discuss artificial neural networks. Figure 2.1 shows the structure of a biological neuron and highlights the connections from and to other neurons.

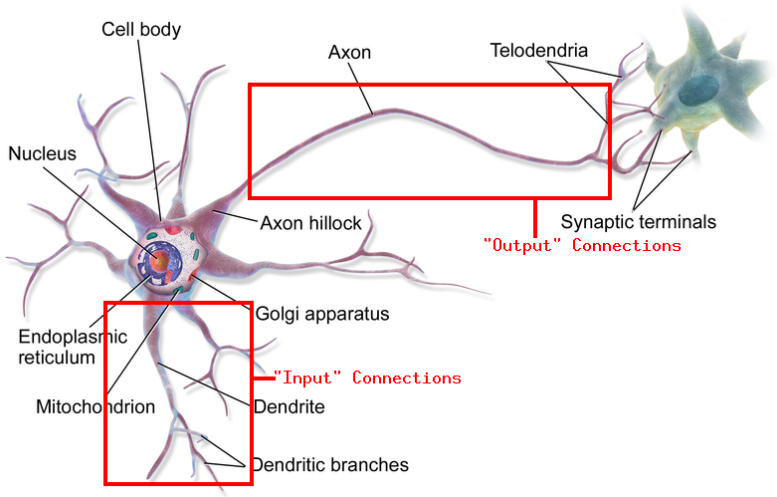


Figure 2.1: Diagram showing structure of a biological neuron with annotations showing the "input" connections, which receive impulses from other neurons. The impulses excite the neuron, causing it to send a signal to its "output" connections, which send that signal to other neurons [16], annotations added by the author of this project.

2.3.4 Artificial Neural Networks

An artificial neural network is a simplification of the complex biological structure. In a biological neural network, any neuron can fire at any time. This would be extremely complex to model computationally so instead, neurons are arranged into connected layers. For each layer, the activations of the artificial neurons are calculated and then these are used to calculate the activations of the next layer. This continues for every layer in the network until the activations of the final layer are generated. The activations of the final layer are the output values of the model. For a classification task the output values are the predicted class for the input data. For many years it was not known how to optimise an artificial neural network with multiple layers until an algorithm called backpropagation was created. Figure 2.2 shows a visualisation of a network with a single artificial neuron and Figure 2.3 shows how a multi-layered neural network is structured.

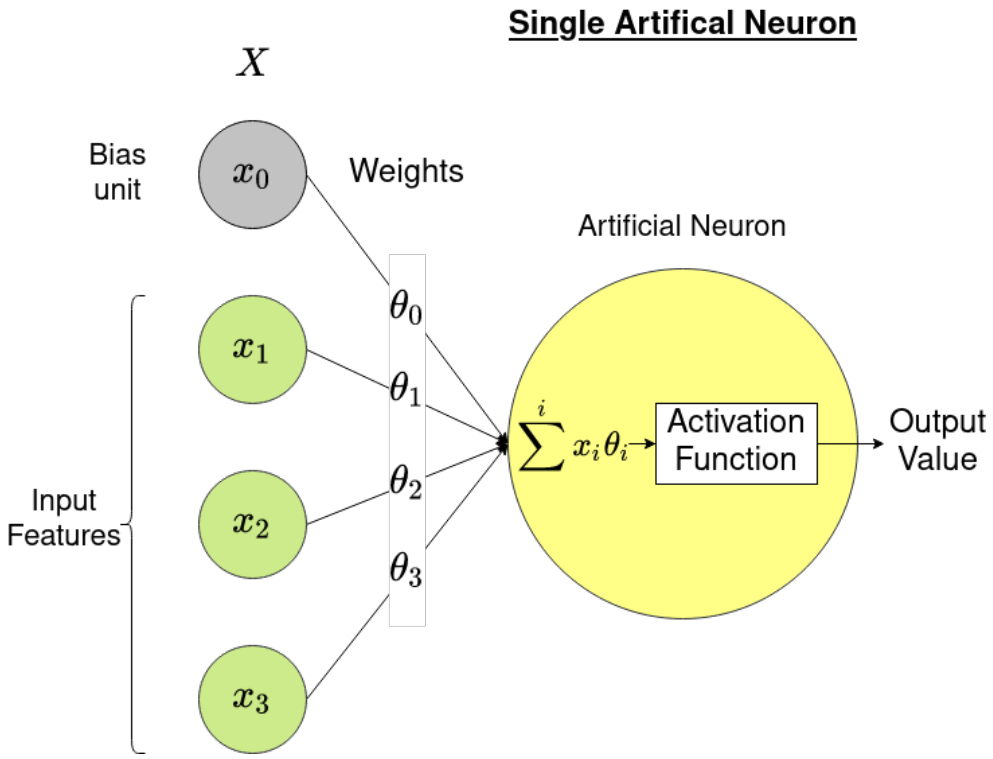


Figure 2.2: Visualisation of a model with a single artificial neuron. The neuron has takes an input vector X, computes the weighted sum of X and passes this value through an activation function to determine the output signal it will send. X in this case is the input features and the weights are the learned parameters θ. In a neural network with more layers this output value would be sent to every neuron in the following layer via a weighted connection.

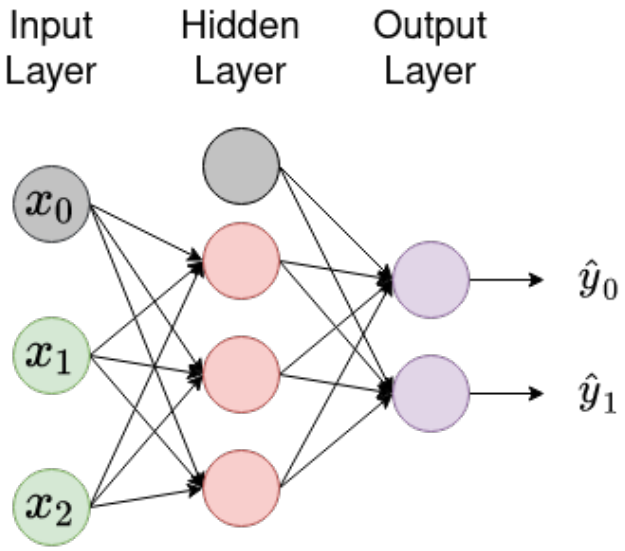


Figure 2.3: An artificial neural with multiple layers. In this architecture, the output from every neuron is connected to the input to every neuron in the next layer. This is called a feedforward, fully connected neural network. For a classification task, the output values would be the predicted probability that the sample is in each class.

2.3.5 Backpropagation

Backpropagation is an algorithm for optimising neural networks by finding the gradient of the loss function with respect to every parameter in the model. It consists of three main steps, the forward pass, the backward pass and the weight update. The forward pass is essentially the same process as when the model makes a prediction. An input vector is passed to the model, the activations for the first layer are found, these are passed to the second layer and so on, until the model outputs a prediction. All the intermediate values need to be preserved to be used by the backwards pass. The error in the prediction is found by passing the predicted and actual labels into a loss function. In the backwards pass the algorithm steps backwards layer by layer and calculates how much each connection contributed to the loss. Once the algorithm has made its way through the whole network and reached the input layers, the amount that each parameter contributed to the loss is known. From this point a gradient descent step can be performed to update all the weights in the model. The error is propagated backwards through the network, hence the name.

2.3.6 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) were developed following research into the structure of the visual cortex, which is the part of the brain that receives information from the eyes [17]. It was observed that neurons receiving information directly from the eyes (essentially the input layer) only receive information about a small area of what is being seen, which was called their "local receptive field". These observations led to the theory that a layered structure is present, with neurons lower in the network extracting features of different areas of the image, with the higher level neurons combining these features to detect more complex patterns, allowing brains to detect objects, identify symbols etc.

Convolutional neural networks have two main components, *convolutional layers* and *pooling layers*. Neurons in a convolutional layer are only connected to a few neurons from the previous layer, mimicking a local receptive field. The weights of a neuron in a convolutional layer are called a filter, or a convolutional kernel. These filters can be thought of as a small image the size of the receptive field. The closer the pixels in a neuron's local receptive field are to its filter, the more the neuron will be activated. The filters in the earlier layers of the network extract lower level features, such as straight lines. As the data moves through the network these lower level features are combined to create higher level features. Generally the last few layers of a CNN will be fully connected layers. These final fully connected layers combine all the high level features to output a prediction about the sample. Training a CNN is finding the filters that lead to the best classification results. Figure 2.4 shows how convolutional neural networks have receptive fields.

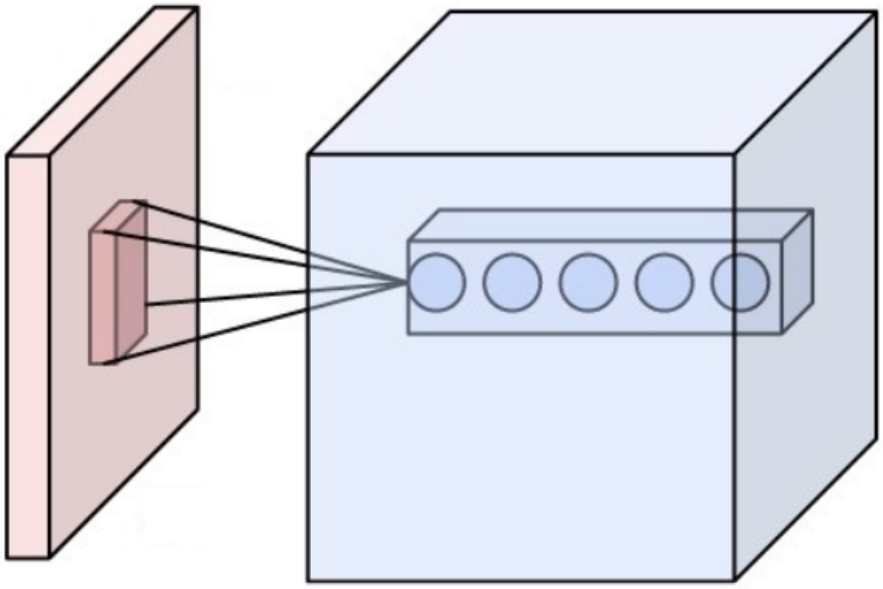


Figure 2.4: Visualisation of a convolutional layer, each neuron has a small local receptive field. Neurons can have the same receptive field but a different filter. Taken from [18]

The other important layers in CNNs are pooling layers. Pooling layers are a simple way of reducing the number of parameters in the network without losing too much of the meaning of the features. Pooling neurons also have a small receptive field, but they simply return the maximum value within their receptive field. Figure 2.5 shows a visualisation of a pooling layer. If a pooling layer contains 2x2 kernels it reduces the size of the input by 2 in each dimension. It still preserves most of the meaning of the image because the highest value in a small area is the pixel that "stands out" the most. There are other options instead of choosing the max like average pooling, which returns the mean value in its receptive field. In practice max pooling is normally used because it is "good enough" and very fast.

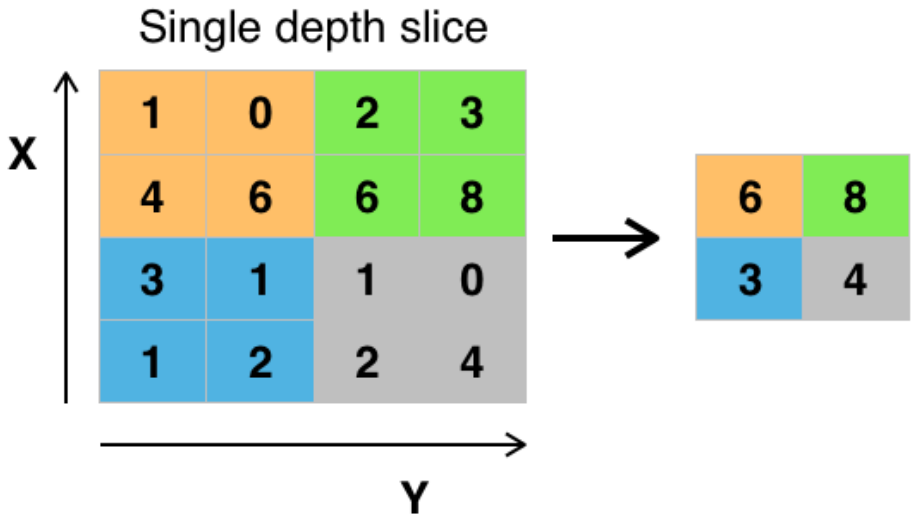


Figure 2.5: How a max pooling layer downsamples a feature map. The output image is only the maximum value from each 2×2 square in the input image. This reduces the number of features by 75%. Taken from [19]

2.4 Deep Learning in Medical Imaging

The previous section discussed the improvements made to deep learning models over the last decade and how effective deep convolutional networks are at completing computer vision tasks. Medical images are a very important part of modern medicine and inform many of the decisions made by medical professionals. Some tasks in medical imaging can be viewed as simple classification tasks. For example, does a brain MRI contain a tumour? Does a chest X-ray contain Covid? By labelling medical images in this way deep networks can be trained to classify images, effectively diagnosing disease.

Deep networks have been shown to reach clinician level accuracy across many diseases in different locations of the body. Networks have matched the performance of clinicians in diagnosing pneumonia [20], brain haemorrhages [21], hip fractures [22], lung cancer [23] and many other diseases [24, 25, 26, 27]. However, there are some significant drawbacks of deep networks that prevent them from being to diagnose disease with no input from doctors. Neural networks offer no explanations about the factors that led to their prediction, because we have no way of reasoning about a neural network's weights. This means if the neural network's predictions are incorrect there is no way of knowing why. A lack of explainability would be undesirable in a clinical environment, where mistakes can cost lives. Deep networks will not replace radiologists any time soon, but they can be used to help radiologists.

2.4.1 Computer Aided Diagnosis (CAD)

Deep models can achieve human level performance in medical imaging tasks, but they cannot replace radiologists. Instead, deep networks can be trained in a way that helps radiologists come to the correct diagnosis more often, while keeping the doctor in control of the decisions. Using deep learning in this way is called computer-aided diagnosis (CAD). Radiology is often extremely tedious and repetitive and it is possible for radiologists to lose focus which can lead to missed diagnosis. After a long shift at work radiologists have been shown to have reduced ability to focus and show signs of ocular strain which can lead to errors [28]. Computer aided diagnosis is a possible way of reducing the strain on radiologists, by bringing parts of the image that may be important to the attention of the clinician.

The neural networks directly predicting disease mentioned in the last stage could be applied directly to computer aided diagnosis, if the radiologists were to use the model's diagnosis as a second opinion about an image. Classification models applied in this way have been used to improve diagnostic performance in fracture detection [29] and differentiating between different breast tumours [30]. Many papers propose systems like this and generally report high accuracies and high recall, but do not test their systems with radiologists so it is hard to say if they are actually effective for computer aided diagnosis, e.g. [31, 32].

The lack of explainability of classification models means there is no reasoning behind the predictions they make, which limits the usefulness of such a system to a radiologist. What is more useful is a model that can show where in the image it believes disease is present because that could draw the radiologists attention to something they may have missed. It is possible to adapt classification models to display a "heatmap" showing what area contributed most to the classification prediction. If the model makes predictions on many small sections within the image, the radiologist can see what sections of the image the model predicts are more likely to contain disease. Using a classification model in this way has been shown to reduce error rate by 85% when combined with a radiologist for breast cancer diagnosis [33]. Figure 2.6 shows a visualisation of this process.

An image segmentation model predicts a class for every pixel in an image. Segmentation models are perhaps the most obvious choice for computer aided diagnosis because they output where they think disease is by default, as opposed to classification which only predicts a label for the image, see figure. While there have been papers published that propose CAD systems using segmentation [34, 35, 36, 37], there seems to be very little research investigating whether these systems actually improve the performance of clinicians. Segmentation models are also harder to train because annotations are needed for every pixel in an image which is very time-consuming, so much less data is available.

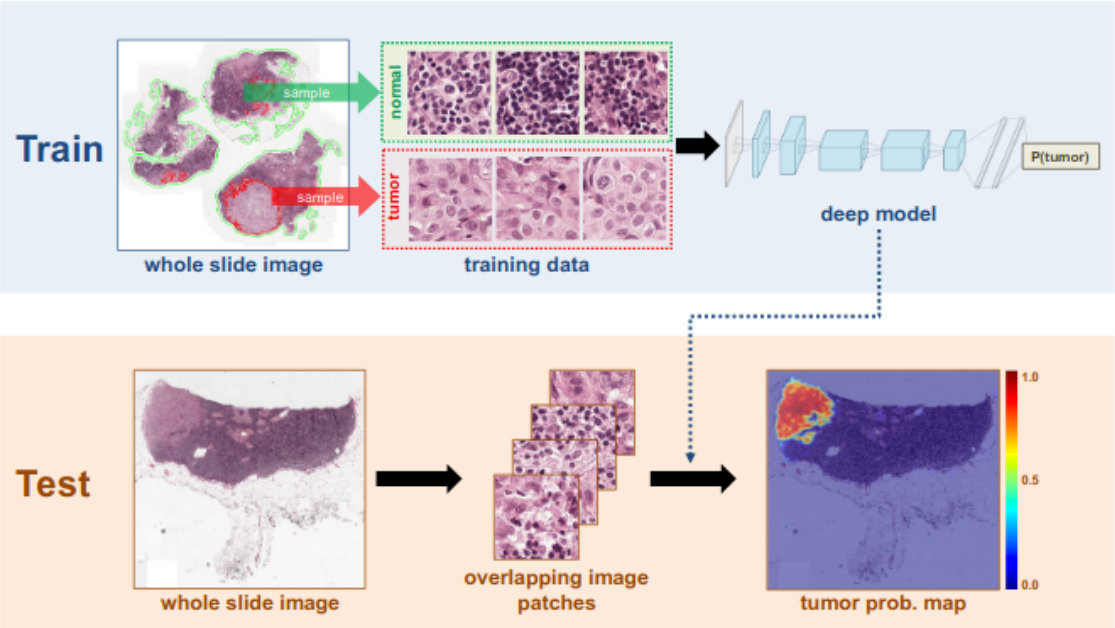


Figure 2.6: Image showing how a classification model can be used to localise disease by predicting on small "patches" of an image. The model makes predictions on many small subsections of the image, and the predictions are used to create a heat map showing where the model predicted the presence of disease [33].

Computer aided diagnosis has been shown to reduce human error rate, but there are some problems with the current research. Most papers that propose CAD models do not investigate if the system actually improves the performance of radiologists. Furthermore, it doesn't seem like there has been any research into the impact on the radiologist of working with these systems. For example do they work faster, are they less tired after their shift, do they feel comfortable using such a system? These questions are important to answer because without them its possible these models will not be aiding radiologists at all.

2.4.2 Publicly Available Data

The increasing focus on deep learning research in medical imaging has led to the release of large publicly available datasets to train these models on. While previously researchers needed access to a hospital to create datasets now anyone can train deep networks on freely available data. What this also means is that much research is done by people without access to doctors to compare their results with. Without a clinician to compare the results with it is hard to know how well the network is modelling the problem.

The Cancer Imaging Archive [38] hosts over 100 open-access datasets which anyone can download and use. Stanford ML Group (`https://stanfordmlgroup.github.io/`) hosts several datasets, the largest of which contains 262,000 chest X-ray images. Competitions are another area where a large amount of data is available. Kaggle (`https://www.kaggle.com/`) host labelled datasets and withhold a test set so people can compete to get the best score. At the time of writing there is a competition on Kaggle with a $30,000 prize for the highest score. Another example is the MICCAI BraTS challenge which has a top prize of $5,000. The increase in publicly available data should lead to an increase in deep learning model performance. However, most research still only trains and tests on a single dataset which makes it less clear if the model can generalise to different data sources, i.e. different image capturing machines from different hospitals.

2.5 Training Across Multiple Modalities

Generally, a deep learning paper will take one dataset and use this dataset to train and test a model. In medical imaging a dataset is normally collected from one location and in one modality. It has already been mentioned that there is a difference between modalities but there is also a difference in the images generated by different machines of the same modality. Training across data captured from multiple locations would help the model generalise, making the results more robust and making the network more effective on unseen data.

2.5.1 Combining Datasets

Training using multiple datasets from different sources is a way of showing that the model is able to generalise to scans from other sources. The other benefit of combining datasets is that there is more data to train the model on. Lack of data is often cited as a problem in medical imaging as datasets can be quite small when compared to datasets for other tasks. While there has not been much research explicitly discussing performance changes when combining datasets, research has shown that training and testing on two breast cancer datasets resulted in higher accuracy that a model trained and tested on a single dataset [5].

These results show that combining datasets for with the same labels achieves better results than training and testing them only on one dataset. This suggests that the gain from having more data outweighs any possible confusion from the noise introduced by using images from different machines/hospitals. Taking multiple datasets and combining them is useful, but all the datasets combined so far have had the same classification labels.

It would be interesting to see if networks could be trained not just across datasets, but also across classification tasks. Current deep learning models are mainly trained for one specific task, but radiologists are capable of understanding many tasks. It is possible that tasks in medical imaging are not completely independent and there could be features that a CNN could learn across tasks.

One example is classification of brain MRI scans as having Alzheimer's disease or being cognitively normal. A different but related disease is mild cognitive impairment (MCI). MCI often progresses to Alzheimer's and predicting whether a case will progress to Alzheimer's is useful. These problems have the same input (brain MRIs) but different classification outputs. Because these tasks are similar it may be possible to combine them and train a model that can classify Alzheimer's disease, MCI or cognitively normal brains. It has been shown that this is possible and an increase in accuracy was found over baseline models that could only complete the binary tasks [39]. This suggests any potential errors between classes from different tasks are offset by the gain in performance from having more data to train on.

2.5.2 Combining Modalities

Doctors often order multiple scans of the same patient in different modalities because no one modality can capture all the important information [40]. For example CT images show bones and other dense tissue very clearly, but do not show much distinction between softer tissues. MR scans show greater contrast between different soft tissues but it would be difficult to use an MR scan to detect a fractured bone. PET and CT scans are often taken at the same time for cancer treatment because CT scans display the anatomical structure of the body well and PET show areas of the body that consume more energy, which is indicative of a tumour. These extra scans provide radiologists with more information and this has led to more research looking at how deep learning models can leverage this information. Figure 2.7 shows the growth in Google Scholar search results for multimodal deep learning papers, taken from [40]

Figure 2.8 shows different ways that multi-modal data can be combined. The data can be fused at the input-level, before any feature extraction has been done. It can be combined at the feature level, after feature extraction has been done on the individual images but before a prediction has been made. Or it can be done at the output level, where each image has an independent prediction which is then combined to create a final prediction. Input and output level fusion are the most common, feature level fusion is more difficult to implement.

The top performing segmentation model in every BraTS challenge from 2013 to 2019 has used a multimodal combination of MRI scans [42, 43, 44, 45, 46, 47]. In this challenge

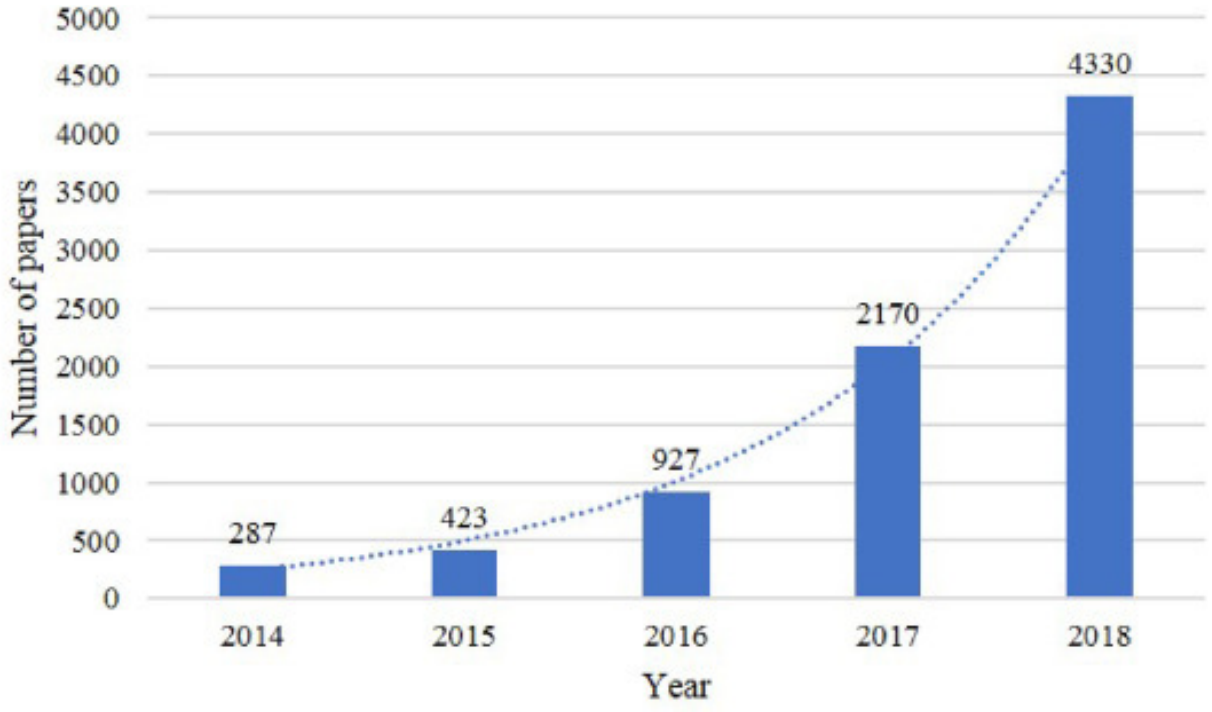


Figure 2.7: The number of results from the Google Scholar search "deep learning medical image segmentation multi modality", taken from [40]

four different MRI modalities for the same patient are provided and the task is to segment the tumour regions. These results suggest that combining multimodal data can lead to better results than any single modality.

While these multimodal networks are more powerful than networks trained on a single modality they suffer from inflexibility due to needing many modalities of scans at once to be used. This would make them more difficult to use in a clinical setting as all of these scans would need to be ordered to get a prediction. Output level fusion does not suffer so much from this problem as only the voting method needs to be changed if a scan from one modality is missing, instead of the whole deep network.

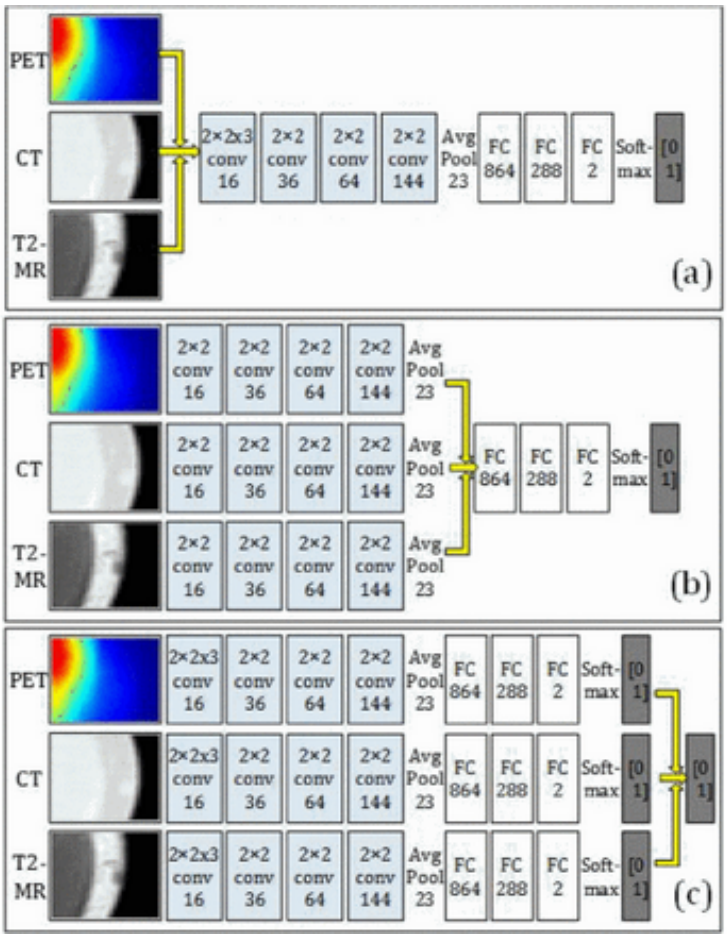


Figure 2.8: A diagram from [41] showing different ways of combining modalities. a) shows treating each image as a single channel in a higher dimensional image and passing that into a 3 channel CNN. b) shows 3 different CNNs all doing feature extraction seperately then their outputs being combined by one fully connected network. c) shows 3 indpendent neural networks making a prediction then combining these independent predictions into one final prediction. This is sometimes called "voting".

2.5.3 Combining Datasets across Modalities

While there has been some research into combining tasks in the same modality there has been almost none into combining tasks across modalities. The exception is [48] which trained a deep network to perform segmentation on three different kinds of scan. The scans were brain MRI, breast MRI and heart CT. The results showed that a deep learning model can be trained across multiple modalities and multiple locations. Interestingly, most of the errors were within the same task. What this means is it rarely confused heart matter with brain matter, or vice-versa ($< 0.0005\%$). The mistakes were mainly within the task, e.g. wrong kind of brain matter. The paper reports that the results are similar to previous published results for networks only trained on the brain MRIs, but it might have been better if the researchers re-created a network themselves to directly compare results on their dataset. It was still an interesting study that shows it is possible to learn across modalities.

2.5.4 Conclusion

Deep networks are an effective tool in medical imaging and show promise in being able to lighten the workload of medical imaging professionals. However, these models are for the most part getting increasingly complex and specialised which will make it more difficult to integrate them into a clinical workflow. The literature on training across datasets and modalities is lacking but it is possible that such a system could leverage the fact that tasks in radiology are related. Such a model could perform comparably to these highly specialised models, but across a variety of tasks. This project aims to create the most basic version of such a system in the hopes that future research will focus more on this area.

There are also some problems with the reporting used in medical imaging papers. Papers often don't compare their results directly with doctors on the same test set as the network was tested on. Papers also often don't make it clear if they used out-of-sample validation at all. This problem is discussed at length in [49].

Chapter 3

Ethics

This project will be working with medical data so special consideration must be given to some ethical concerns. The main concern when using medical data is the possibility of any images being traced back to the original patient. To ensure this is not possible, this project will only use data that is fully anonymised and intended to be used for research. The 102 datasets to be used in the project were identified and a list was sent to the University Teaching and Research Ethics Committee to request approval to continue with this project. Ethical approval was granted and the letter of approval can be seen in Appendix A. A table containing references to every dataset used in this project can be found in Appendix C.

Chapter 4

Project Overview

To achieve the goal of classifying medical images by modality across many datasets, a dataset and a model are required. A broad overview of the steps required to complete this project can be seen in Figure 4.1.

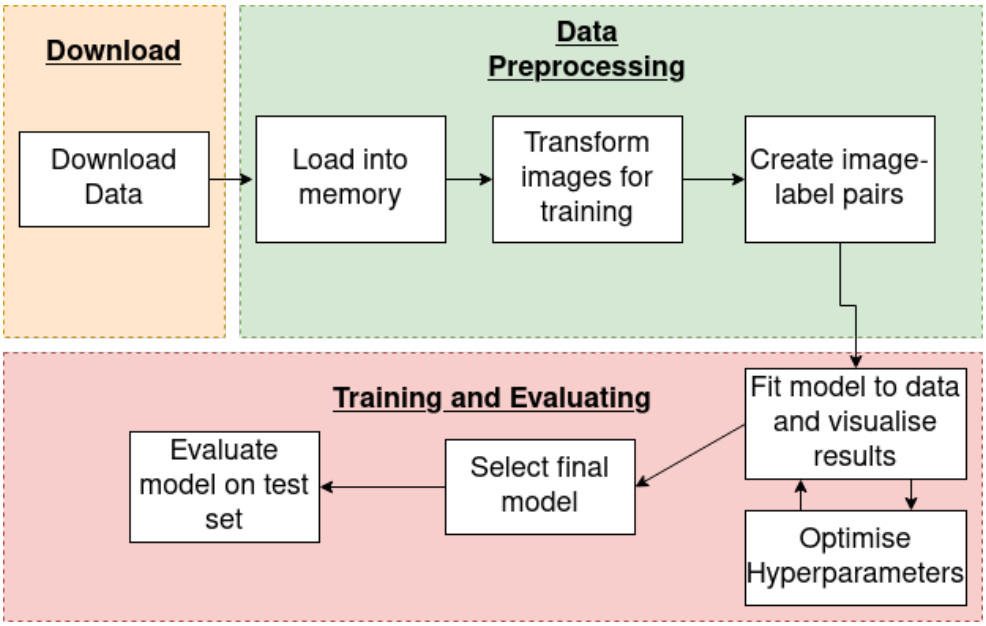


Figure 4.1: Flow diagram showing the overview of the project, from data download to evaluating the model.

4.1 Downloading the Data

Most of the datasets will be sourced from The Cancer Imaging Archive (TCIA). The Cancer Imaging Archive provides a REST API for programmatically downloading open access medical imaging datasets. This allows for a script to be written to download more data than would be feasible if every dataset had to be found individually. Because the Cancer Imaging Archive mostly hosts datasets related to cancer, seven datasets not hosted by TCIA were also identified to get a better spread of data sources.

4.2 Data Preprocessing

Before the downloaded data can be used to train deep models there is some processing required. The images need to be the same size and the pixel values need to be in the same range. To achieve this, the images will be resized and the pixel values rescaled.

4.3 Model Selection

As discussed in the literature review, deep Convolutional Neural Networks (CNNs) are the state-of-the-art for image classification but many different architectures are available. This project will adapt several state-of-the-art models to predict on greyscale images instead of 3-channel colour images. This will allow each model to be trained on the training set and evaluated on the validate set to find the model that best fits the classification task.

4.3.1 Training and Hyperparameters

Hyperparameters are all the variables involved in the learning process that are set before training, e.g. number of layers, optimisation strategy, learning rate etc. The model that achieves the highest performance in the training process will be optimised further by tuning hyperparameters manually. Training deep models is an iterative process where changes are made to these hyperparameters and then the model is evaluated to see what effect the changes had on the model's performance. Figure 4.2 shows a visualisation of this process.

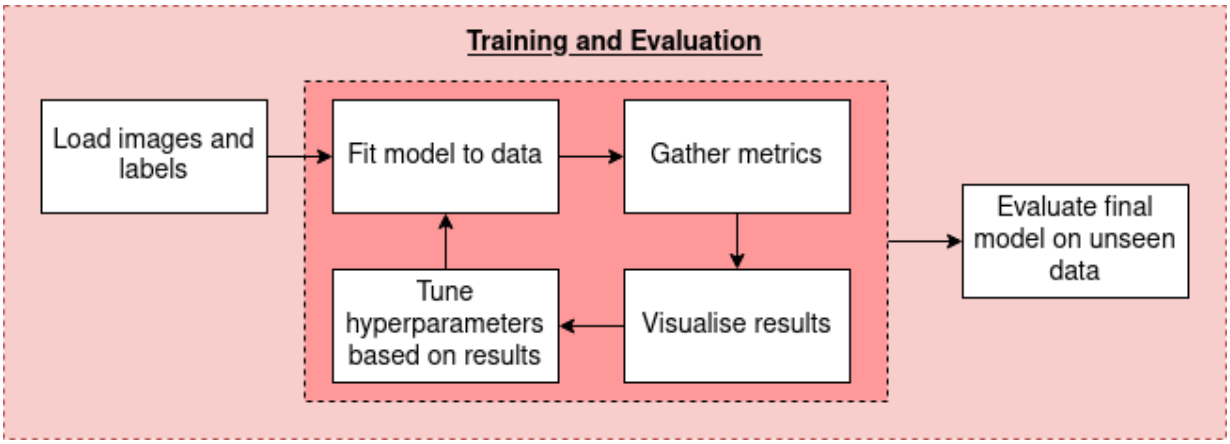


Figure 4.2: Flow diagram outlining the process of fitting the model to data and tuning hyperparameters.

4.3.2 Model Evaluation

Once the training is complete the models will be evaluated on unseen data. It is important this data has not been seen in the training process as the purpose is to simulate performance "in the wild", on real world data that the model has not seen before.

Chapter 5

Data

This project requires many datasets because it needs to be clear that the model can generalise across different areas of the body for each modality. The increasing number of open access medical imaging datasets makes a project like this possible. For this project to work, there are some important properties that the data collected must have. There must be roughly the same number of images for each modality, because unbalanced datasets make the learning process more difficult. For every modality, there must be images from several locations. Multiple locations are important otherwise it wouldn't be clear if the model was learning features specific to the location of the body or the modality. Finally, there can be no data leakage between the data that is used to train the model and the data that is used to test the model. One way this could happen is if images from the same patient were in the training and testing sets of images. The best way to avoid this is to make sure that images from one dataset are contained entirely in the training or testing set. With all these requirements, it is clear that many datasets will be needed. The following chapter outlines how the data needed was acquired and transformed into a machine learning friendly format.

5.1 Acquisition

One of the largest hosts of open access medical imaging datasets is the Cancer Imaging Archive (TCIA) [38]. The Cancer Imaging Archive hosts over 100 open-access datasets with the purpose of furthering cancer research. As such almost all the datasets contain cancer/non-cancer images, although recently some Covid datasets have been added. Being intended for cancer research means that the datasets are generally CT, MR and PET scans as these are the most common scans used in cancer screening and diagnosis. To ensure

a good spread of locations and modalities, some other large datasets were sourced. This section will discuss the process of sourcing all the data needed for this project.

5.1.1 The Cancer Imaging Archive

The Cancer Imaging Archive provides a REST API for programmatically downloading the datasets that they host [50]. A REST API allows downloading data through with a script as opposed to through a graphical interface. The REST API makes it easier to download many datasets at once which is useful for this project. Figure 5.1 shows an overview of how this download was implemented.

The REST API was used to download the metadata that accompanies each of these datasets. The Cancer Imaging Archive also provides example code to simplify this process. The code is hosted on GitHub but was written in Python 2, which is no longer supported. As part of this project the repository was forked and updated to Python 3 so it could be used for development.

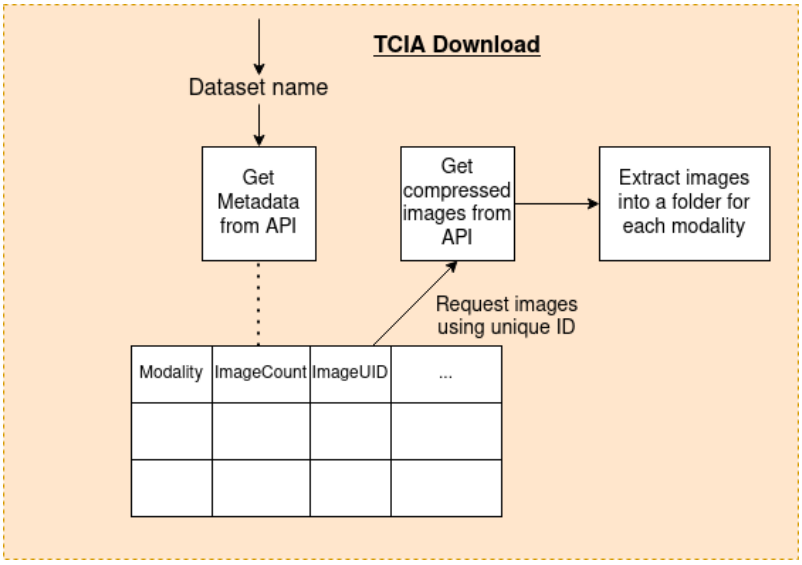


Figure 5.1: Flow diagram outlining the process for downloading images from the TCIA REST API.

A Jupyter Notebook was used for the analysis described in this section, this decision was made both for ease of development and for reproducibility. In the analysis 22 different options were found for image modalities across all datasets hosted on TCIA. The Cancer Imaging Archive provides a table which contains the meaning of these options [51]. 13 of the 22 options were discounted for not being image formats. After this step the image types left in the analysis were MR, CT, PT, CR, DX, NM, XA, US and MG. The meaning

of these abbreviations is in Table 5.2. It is important to note here that whenever an image is referred to as being 3D it means that the image is made of many 2D images stacked on top of each other. These 2D images are greyscale. That is, the images only have one colour channel. This note is important because colour images are also technically 3D as each pixel has a channel for red, green and blue. Once these modalities were established

Abbreviation	Meaning
MR	Magnetic Resonance Imaging scan (3D)
CT	Computed Tomography scan (3D)
PT	Positron Emission Tomography scan (3D)
CR	Computed Radiography, a kind of X-ray (2D)
DX	Digital Radiography, a kind of X-ray (2D)
NM	Nuclear Medicine, broad term covering many medical images including PT
XA	X-ray Angiography, a specific kind of X-ray for viewing the heart
US	Ultrasound (3D)
MG	Mammography, a breast scan that is almost always a kind of X-ray (2D)

Table 5.2: The 9 imaging modalities present across all TCIA datasets.

code was written to count how many images of each modality were available from TCIA. One important note is that for 3D scans, (MRI, CT, PT, US), an image is one 2D slice within that scan. This means the number of images for these scans is going to be higher because one CT scan could generate 100-200 images and one X-ray generates one image. The image counts can be seen in Figure 5.3. As can be seen there are several orders of magnitude more MR, CT and PET scans than X-rays (CR and DX).

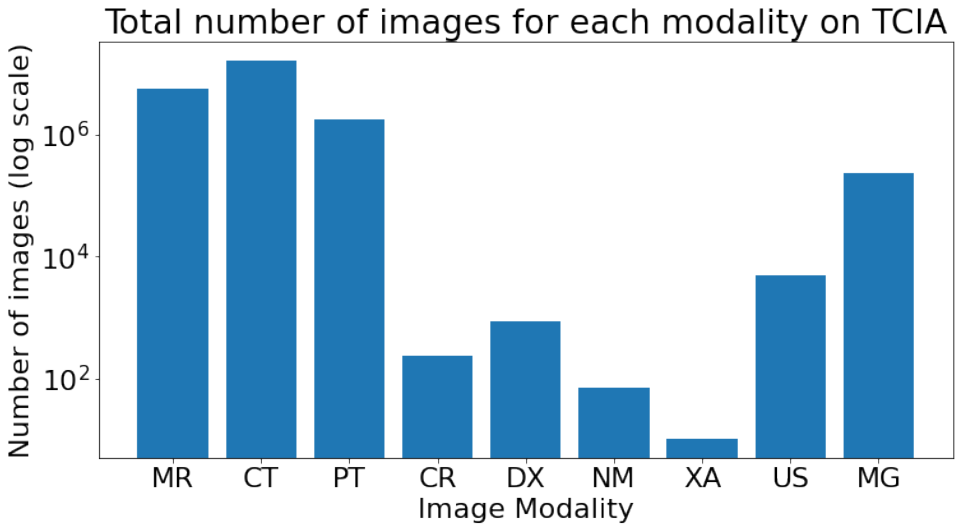


Figure 5.3: The number of images for each modality available on TCIA. Note the log scale. The meaning of each of these abbreviations is shown in Table 5.2.

With this visualisation, X-ray angiography (XA) was discounted from further analysis because there were too few images available (10). Nuclear medicine (NM) was also discounted as it is not a specific imaging modality, it is an umbrella term that includes PET scans, and there were only 71 images. While there were 5000 ultrasound images on TCIA, the vast majority of them were only in one dataset. Therefore, ultrasound (US) also had to be discounted from further analysis.

The 3 X-ray modalities (DX, CR and MG) were combined into one class, XR. This decision was made because there were too few of any one of them to create a good split. Combining them like this is valid as they are used for the same purpose, but some hospitals have DX machines and some have CR. It also isn't clear which category (CR or DX) the mammographies would be placed in to. MG should not be its own category because then the network might learn to identify images of breasts as MG, instead of learning general features identifying X-ray images. After this step, the final modalities were MR, CT, PT and XR.

Download

Once the split had been identified a script was written to iterate over the selected datasets and download the image files.

5.2 Other Datasets

Seven other datasets were also downloaded. Three were provided by the Stanford ML Group, these were CheXpert [1], a dataset of 200000+ chest x-rays, MRNET [52], around 100000 MRI images and MURA [53], a collection of 40000 bone x-rays. Also downloaded were BraTS20, a brain MR dataset, the NIHCC chest X-ray collection, and X-rays from the NDA Osteoarthritis Initiative. These datasets were all accessed and downloaded through the websites hosting the datasets. Before this data could be used for training a train-validate-test split needed to be created.

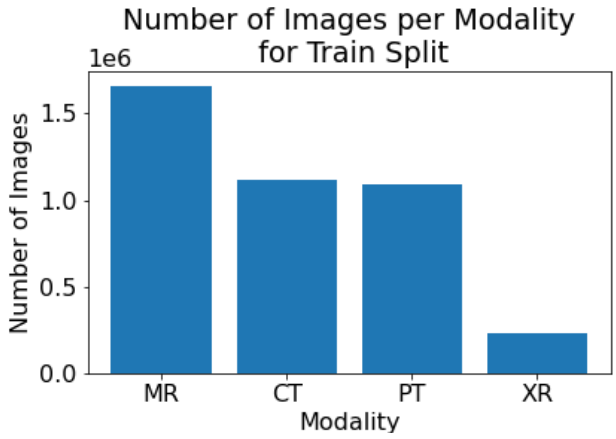
5.3 Train-Validate-Test Split

The terms train, validate and test are not used in exactly the same way in research papers so their exact meaning should be defined here. The training set is used to perform the training of the model. It is available to the research at all times to train and fine-tune the

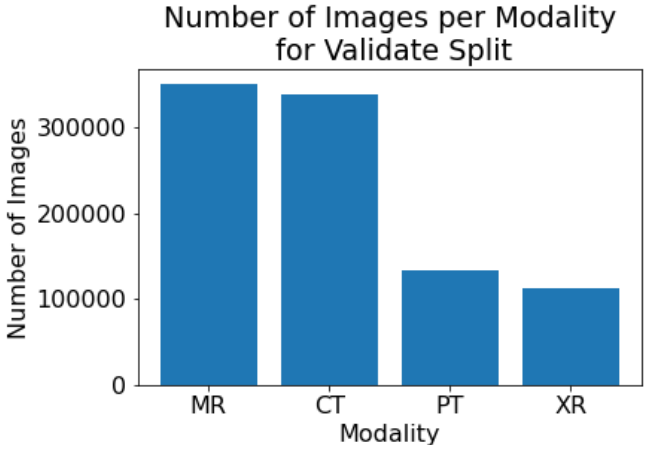
model. The test set is used only for the final evaluation, and nothing else. It is not touched at all during the training process. The validate set is used to simulate the test set. The validate set can be used in the training process to judge how the model would perform on unseen data. However, the validate set cannot be used to actually evaluate the final model as it is not unseen data, it has been used during the training process. In some papers the words validate and test are used interchangeably, and some papers refer to a test set as an out-of-sample validation set.

The selected split is important because a poor split may allow the model to "cheat". For example, if all the MRI images in the train and validate set are of the brain, then the model may recognise features of a brain as being an MRI scan. Then if the test set has CT scans of the brain the model might predict all of these as MRI images incorrectly. It is important to create the splits at the dataset level to prevent data-leakage. This is because scans of the same patient in the same modality are likely to be similar, so if there is an image of the same patient in the train and test set then the test set does not contain completely unseen data. Putting each dataset into one of train, validate or test prevents this data leakage. Splitting the datasets like this also helps achieve the goal of demonstrating generalisation across datasets.

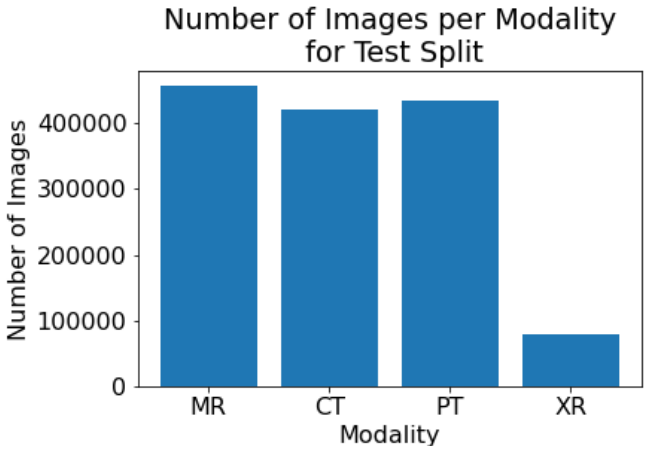
The train-validate-test split was created using an iterative process of moving datasets between the different splits and visualising the number of images for each modality. Special care was taken to ensure a spread of locations in each of the splits. Each dataset was annotated with the location of the body it contained. The manual split ensured that there are at least two locations for each modality in each of the train, validate and test split. The main difficulty for this was X-rays, which are mostly of breasts in TCIA. This meant the non-TCIA datasets had to be carefully split. The final breakdown of the train-validate-test split can be seen in Figures 5.4a, 5.4b and 5.4c. A maximum of 100000 MR images were taken from each dataset, and a maximum of 50000 CT images were taken from each dataset. This was to prevent the very large CT and MR datasets from creating a very unbalanced final dataset. There are 74 datasets in the train set, 13 in the validate set and 16 in the test set. The list of datasets in Appendix C has every dataset used in the project and the split it was placed in to.



(a) Number of images for each modality in the created train set.



(b) Number of images for each modality in the created validate set.



(c) Number of images for each modality in the created test set.

Figure 5.4: Figures showing the number of images for each modality in the created splits: a) train, b) validate and c) test. Note that each graph has a different scale, the purpose is to show the ratios of each class are similar. There are 74 datasets in the train set, 13 in the validate set and 16 in the test set.

5.4 Preprocessing

Before the downloaded data can be used to train a model there are some preprocessing steps that need to be applied to the images. The three preprocessing steps done in this project are resizing, rescaling and standardisation. It transpired that standardisation was not feasible across this many images and that will be discussed at the end of the section. Figure 5.5 shows an overview of the preprocessing implemented in this project and the purpose of these preprocessing steps.

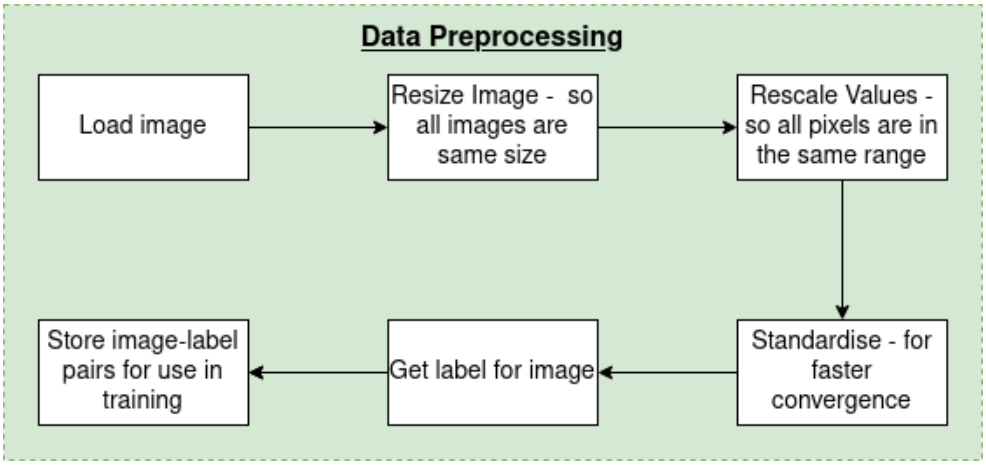


Figure 5.5: Flow diagram outlining the steps taken in the preprocessing pipeline.

5.4.1 Resizing

To be used to train a deep learning model all images should be the same size. This is because the inputs to a model need to be the same shape because a model learns patterns in the data. If images are different shape then the patterns will not be the same. Furthermore, generally the images are too large to be used to train a deep model. Decreasing the size of the images makes the calculations easier, which makes the training process faster and means the model can have more parameters.

5.4.2 Feature Scaling

With the images all being the same size the next issue is making sure the pixel values are all in the same range. This process is called feature scaling. Features as inputs for deep learning models should all be in the same range because these models do not have context for these values. For example, one image could have a range of possible pixel values from 0

to 1024, and another could have a range of -32768 to 32767. In the first image, a completely white pixel would have a value of 1024. In the second image a completely white pixel would have a value of 32767. While both of these pixels have the same meaning (fully white), the actual numbers are different so the model would treat them differently. Feature scaling is also important in gradient descent because the step size in gradient descent is a factor of the input features. So, if features are in different ranges the step sizes will vary and the algorithm will not converge smoothly.

To rescale the features a desired range of values is required. The range chosen for this step is not especially important, a minimum of 0 and a maximum of 1 are often chosen because many activation function saturate at these values. The formula for this transformation is:

$$p = \frac{(p - max(p))}{max(p) - min(p)} \tag{5.1}$$

It is important to note that $max(p)$ and $min(p)$ are defined as the maximum and minimum **possible** values, and not the maximum and minimum pixel value within the image. If these values were calculated based on the maximum value within the image, then the highest value in an image would be treated as white after rescaling, even if it was possible for a pixel to be much brighter in the image.

Feature Scaling Issues

The problem with rescaling medical images is that not all medical imaging modalities have a set range of valid pixel values. This is because each machine is different, and can store data in a different format or capture a different range of data. CT scans use a scale called the Hounsefield scale, and this means all CT scan values fall between -1000 and 30000. However, MRI scans have no such specified range so there is no way of knowing what the maximum and minimum possible values could be for a given scan. PET scans also do not have a standard pixel range to conform to. This means it is not known what the maximum and minimum possible pixel values could be for any one image.

While the maximum and minimum values for many of the images cannot be known, the images are still saved into a binary format at some point. The image files will have metadata that contains the number of bits each pixel uses and whether the data is signed. This information can be used to obtain a theoretical maximum and minimum value for the pixels. This works on the assumption that data types chosen by the image capturing machines are sensible, and only use as many bits as they need. If the data type has a much larger possible range than the actual captured values then the rescaling will push all the data in the image close to zero, effectively undoing the purpose of rescaling.

The other option would be to rescale each image using the maximum pixel value within

the image as $max(p)$, and the minimum value within the image for $min(p)$. This would make the assumption that values very close to the maximum and minimum are present in every image, which may or may not be true. For example, the maximum possible value for Hounsefield units is observed when gold is present in the scan, which is almost never going to be the case. Neither option for rescaling is perfect but finding the maximum and minimum based on image data type will be used for this project. To achieve this, rescaling functionality was added to the open-source package MedPicPy.

After the data had been resized and scaled it was saved in separate files to be used in the neural network training process. The files were saved at this point because the preprocessing takes a long time and it allows the neural network code to be completely separate from the preprocessing code.

5.4.3 Problems

Another useful preprocessing step that is often performed when worked with image data is standardisation. Standardisation is ensuring that across the whole dataset, the pixels have a mean of zero and a standard deviation of one. A mean of zero is desirable because all weight updates are a factor of the input samples. If the samples have a mean above or below zero, then the weight updates will have a bias towards that direction that will have to be 'unlearned' by the model. Therefore, a mean closer to zero means that there will be less bias which will in general lead to faster convergence.

However, a problem was encountered when implementing standardisation for this project. The process for standardisation is simple. The mean and standard deviation of pixel values is found. Then for every pixel, the new pixel value is found by subtracting the mean, and dividing the result by the standard deviation. The problem arises when calculating the mean and standard deviation values because of floating-point precision problems. A small amount of precision can be lost with each mathematical operation on a floating-point value. With around three million images in the final train set that are each 224x224 pixels, there are around 150 billion individual floating-point values in the dataset. When calculating mean and standard deviation over this many values floating point precision becomes a factor and can impact the resulting mean and standard deviation. Since these errors would be transferred to all the pixel values it was decided to skip standardisation and only rescale the data. There are ways of mitigating these problems somewhat but since standardisation is optional anyway it was decided not to implement these.

Chapter 6

Neural Network Training

The basic training loop was kept the same across all networks to keep the code simple. Initially every model was trained for five epochs then the weights were saved and the results on the validate set analysed to see if more training was required or if the model had already converged.

6.1 Model Selection

There are many available CNN architectures and it can't be known beforehand which architecture will perform best for a given task. To get an idea of which architecture would perform the best, several different CNN architectures were tested on the validation set. The selected architectures were ResNet, VGG and DenseNet and MNASNet. The idea behind this selection process was to get state-of-the-art models with very different architectures, each of which will be summarised in this chapter.

6.1.1 VGG

VGGNet [54] was the runner-up in the ImageNet [55] 2014 challenge. It has a "classical"structure, closely following the original CNN architecture described in the literature review. To recap, this is an architecture that uses convolutional and pooling layers to extract features and then a fully connected network to combine the features and make a prediction. This architecture was chosen because it is a good example of the "classical" CNN approach which will allow comparison against the more complex architectures.

6.1.2 ResNet

An interesting result observed in the VGG paper is that increasing depth improved performance up to 19 layers, at which point the performance starts going down again. This reduction in performance is not due to overfitting, as the training error also increases. This means that deeper models are just harder to optimise in some way. ResNets [56] tackle this problem by adding skip connections. A skip connection takes the input to a layer and adds it to the output of a layer further along the network. These skip connections were added because very deep networks can suffer from the *vanishing gradient* problem. In the lower layers of deep networks the partial derivates of the error function can be very close to zero, which leads to weight updates being tiny or nonexistent. This basically means that these layers cannot learn anything because the weight updates are too small. Skip connections increase the updates to these layers early in the training process to allow learning to start. The idea is that once learning has started then the network will learn to ignore the values from the skip connection. Figure 6.1 shows an example of a skip connection. A ResNet won the ImageNet 2015 challenge. ResNets with 18, 34 and 50 layers will be analysed to see how depth affects these networks.

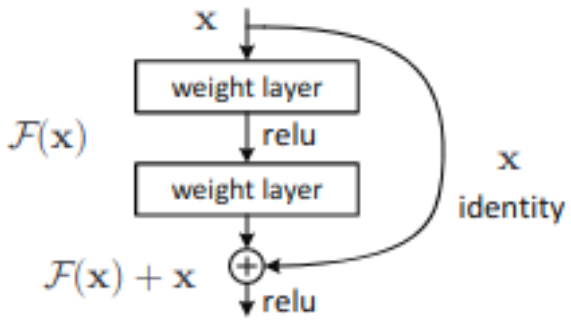


Figure 6.1: A "residual block" in a ResNet. The input from one layer is added to the output of a layer further down the network. A ResNet is made of many residual blocks stacked on top of one another. Taken from [56].

6.1.3 DenseNet

ResNets improved performance massively over conventional CNNs due to the skip connections. DenseNets [57] further expand on this idea by adding many skip connections. DenseNets are made up of several stacked dense blocks. The output of every layer within

a dense block is added to the input to each layer following it in the same dense block, so each dense block contains many skip connections. This allowed a slight increase in performance over ResNets by allowing even deeper networks to be trained (the paper reports best results with a 190 layer network). Figure 6.2 shows the structure of a DenseNet.

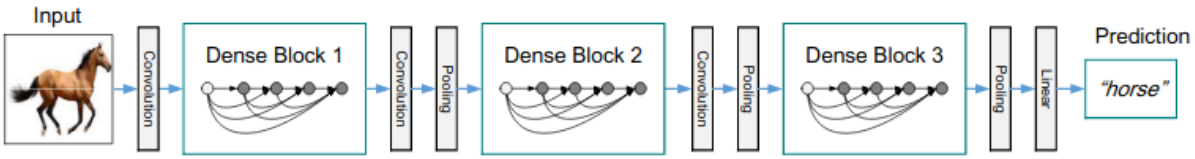


Figure 6.2: Diagram showing a DenseNet with three dense blocks. Within the dense blocks there is a skip connection from each layer to all following layers. Taken from [57].

6.1.4 MNASNet

MNASNet is a model optimised for use in mobile phones and other embedded devices where hardware is limited and time to predict needs to be low. It achieves similar accuracy to ResNet-50 on the ImageNet challenge with 4.8 times fewer parameters. This model was added to the analysis to see how a model with fewer parameters could perform, and to allow analysis of prediction time against accuracy across various models.

6.2 Architecture Results

All models were trained for five epochs, where an epoch is one run using all samples in the training set. For every model the learning rate started at 0.1 and was divided by 10 when the training loss plateaued. A momentum of 0.9 was used, this value is fairly standard in deep learning papers. Learning rates and momentum are discussed in the Deep Learning section of the Context Survey. The train set contained 2954097 images and the validate set contained 704685 images. A batch size of 128 was used. At the end of every epoch the model weights were saved so training could be resumed if the model had not converged.

Five is a low number of epochs compared to the number that would be used to train these models on similarly sized datasets, such as ImageNet. For example, the ResNet that won ImageNet 2015 was trained for 1200 runs over the 1.2 million image dataset. The first reason for this disparity is the time it takes to train these networks. It took around 8 hours per epoch on a machine with 2 Nvidia 1080Ti's and around 4 hours per epoch on a machine with 2 Nvidia Tesla V100's, though access to the latter machine was limited. The speedup is likely as much due to the faster storage as the more powerful GPUs. The

second reason for the low number of epochs is the overfitting observed across all models, which will be discussed in the analysis. The effect of storage access times on training will be discussed further in the Problems section at the end of this chapter.

At the end of every epoch the model was evaluated on the validate set, and the training and validation accuracy were found to allow for analysis of the training process. At the end of the training process the model was evaluated on the validate set again, this time writing all the predicted and actual labels to a file to allow for detailed analysis, such as measuring precision and recall and creating confusion matrices. The intention behind this step is to find the architecture that can model the problem best, and then experiment with hyperparameter tuning to drive up accuracy as much as possible to use on the test set.

6.2.1 Accuracy

The models were all trained and evaluated on the validate set to select a model to be tuned further to try to drive performance as high as possible. The accuracies of each model on the validate set can be seen in Table 6.3 which is plotted in Figure 6.4.

Model	Accuracy
DenseNet	78%
ResNet18	76%
ResNet34	52%
ResNet50	68%
VGG16	85%
MNASNet	71%

Table 6.3: Table of results with each model and the accuracy achieved on the validate set.

From the results it can be seen that the VGG16 model has the highest accuracy on the validate set after training for 5 epochs. The plot also shows the large difference between training and validation set accuracy for every model. A large difference between training and validation set accuracy is likely caused by the model overfitting on the training set. For more insight into how well the model has generalised the training and validation curves can be inspected.

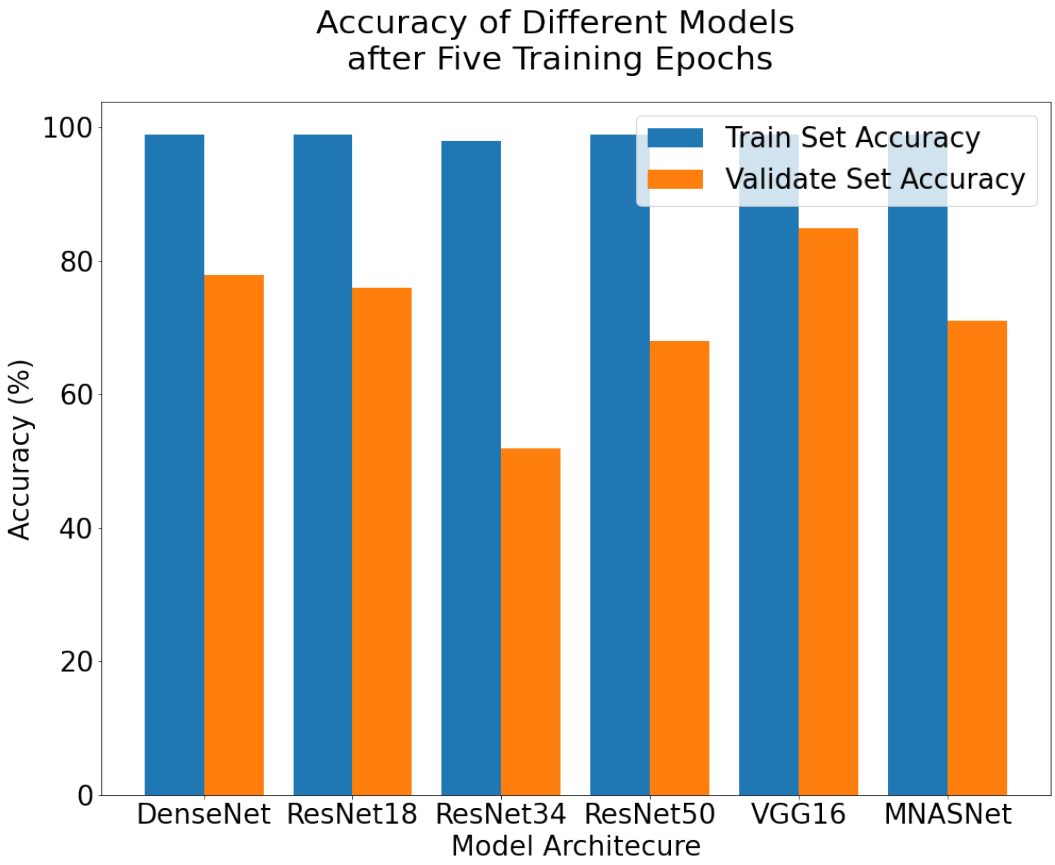


Figure 6.4: Plot showing the classification accuracy of different models on the validate set after being trained for five epochs.

6.2.2 Training and Validation Accuracy Plots

Training-validation accuracy plots show how the training and validation accuracy changes each epoch. If the training and validation curves stay close together then the model is probably generalising well, because it is performing the same on seen and unseen data. If the training accuracy is much higher than the validation accuracy then the model is overfitting on the training set and may not perform well on the test set. If the accuracy values are still increasing, i.e. the curves have not flattened, then the model has not converged and needs more training epochs. The training-validation curves for the VGG16 can be seen in Figure 6.5 and the curves for the ResNet50 can be seen in Figure 6.6.

Both plots clearly show the models are overfitting on the training set. Both models reached similar training accuracies, but the results suggest that the VGG model has generalised more.

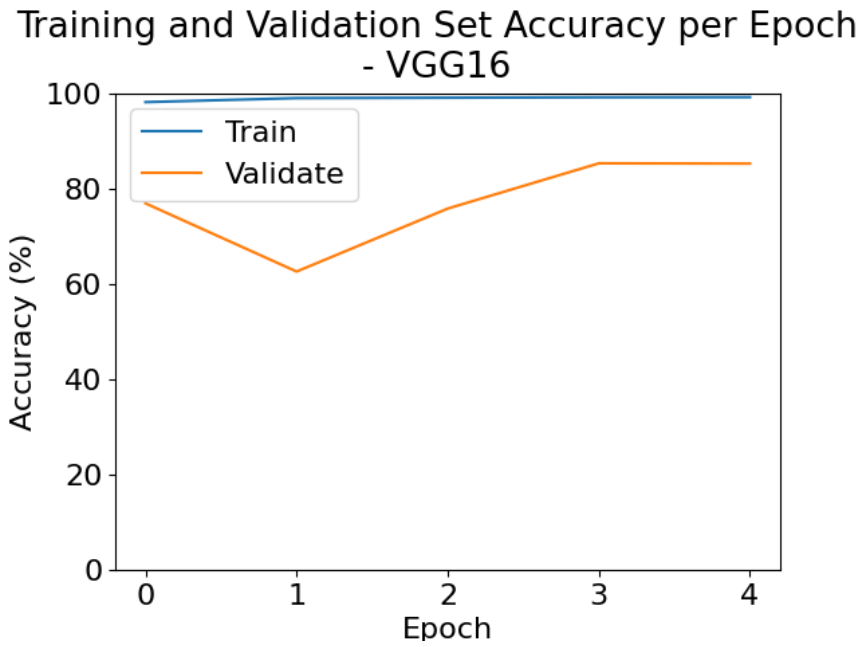


Figure 6.5: Training and validation accuracy at the end of every epoch for the VGG16 model.

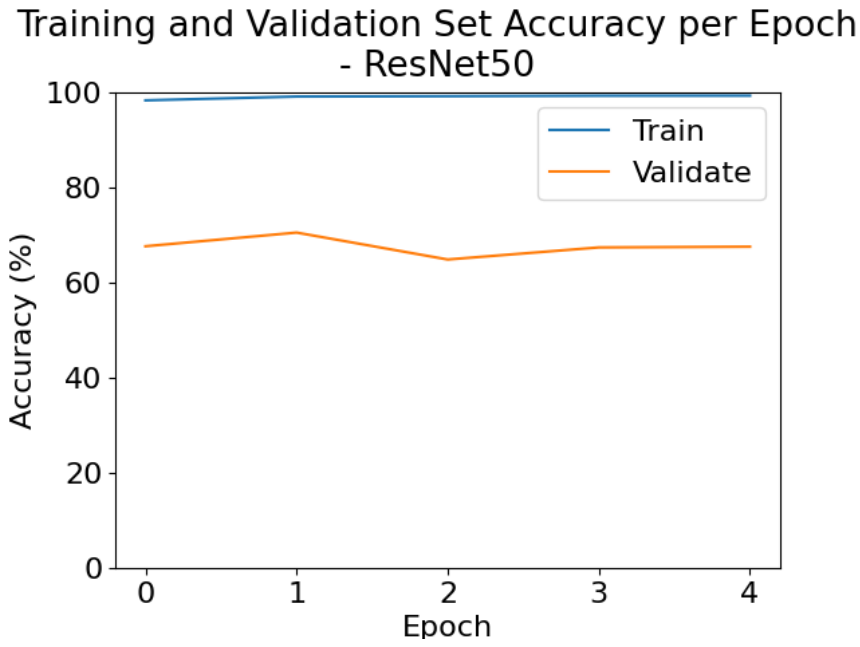


Figure 6.6: Training and validation accuracy at the end of every epoch for the ResNet50 model.

6.2.3 Discussion

Interestingly, VGG16 had the highest accuracy when it is the least deep model. Generally deeper models have been found to perform better in deep learning tasks. All the models had a very high difference between training and validation accuracy, which suggests they were all overfitting on the training set. VGG16 has been selected for hyperparameter tuning because it performed best. The hyperparameter tuning will focus on regularisation, to attempt to bring the training and validation accuracies closer together.

6.3 Hyperparameter Optimisation

All models are overfitting, but there are ways of forcing a model to generalise. A model with fewer parameters has to learn more general features it does not have the parameters to learn as many features. This project will experiment with Dropout layers and L2 regularisation.

Dropout

Dropout is a way of reducing overfitting by randomly dropping neurons and all weights associated with them. This reduces the number of parameters which is a good way of reducing overfitting. Since the neurons are dropped randomly it should force neurons to learn independently, so there should be less redundancy in features extracted. Each Dropout layer has a hyperparameter p, which is the probability any individual neuron will be dropped.

L2 Regularisation

L2 Regularisation adds the L2 norm of the weight vector to the error term. The L2 norm is the square root of the sum of the squared weights:

$$l^2 = \sqrt{\sum_{i=0}^{n} \theta_i^2} \tag{6.1}$$

This means a model with many weight values with have a much higher loss, even if it is very accurate on the training set. This encourages the optimiser to push weight values to 0 to reduce the error, which in turn reduces the number of parameters (because setting a parameter to zero is functionally the same as removing it from the model). L2 regularisation

is controlled by a parameter called 'weight decay', which is the factor the L2 norm of the weights is multiplied by before being added to the loss. This is typically a very low number, around 0.001.

6.3.1 Tuning Results

As a starting point 5 Dropout layers with $p = 0.5$ were added to the network. Then this network was trained with increasing L2 penalty, from 0 to 0.02.

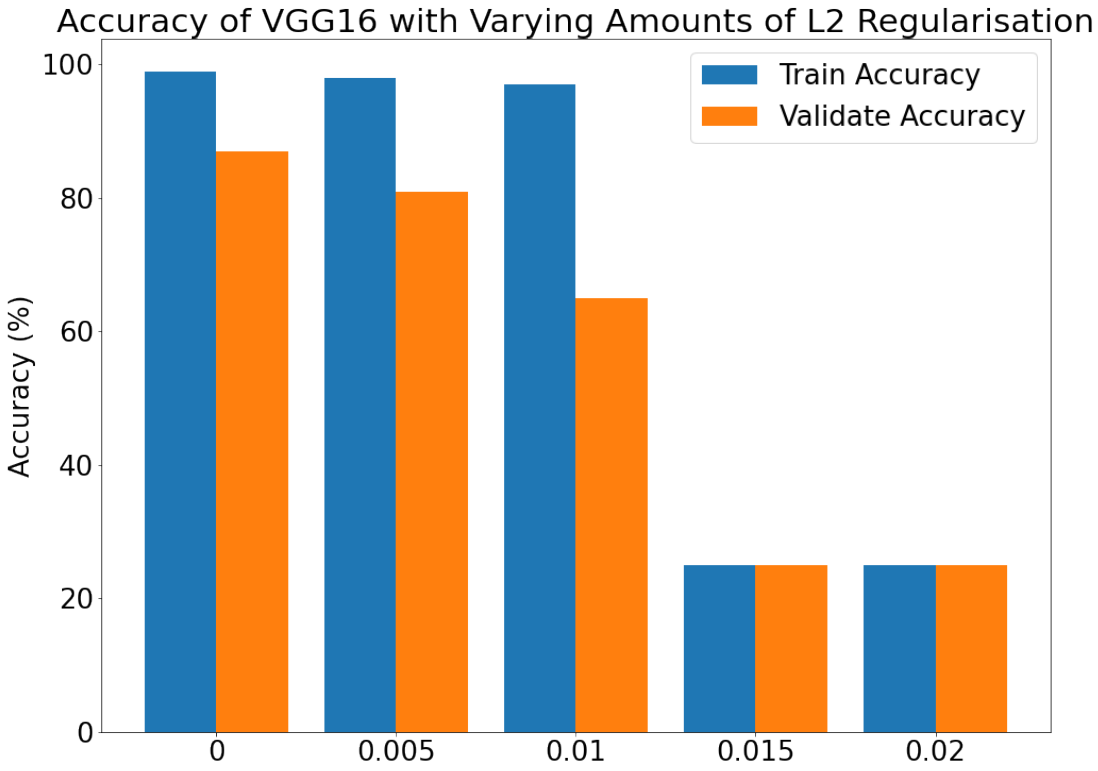


Figure 6.7: Train and validate set accuracy after 5 epochs with varying amounts of L2 regularisation.

Figure 6.7 shows the results from the hyperparameter tuning experiment. The model with the Dropout layers and no L2 regularisation penalty had the highest performance on the validate set but did not do much to reduce the large gap between the performance on the validate and train set. Ideally more experimentation would be done with changing the

p value for the Dropout layers and changing the number of layers in the model but time constraints meant this was not possible.

With this hyperparameter tuning complete the models can be evaluated on the test set, which has been held back until the evaluation step.

6.4 Problems

Training these networks took a long time because of problems with the hard drive the data was stored on and because it is just a huge amount of data, the preprocessed data takes up 863GB. In the first iteration of the training loop it was observed that around 2000 images into training the time to process a batch of images increased. The time to process a batch increased by around 4-8 times at this point which is a huge slowdown as the training process can take days. This increase could be due to the hard drive that was used, or operating system thrashing.

6.4.1 Aside: Virtual Memory and Pages

To explain thrashing, virtual memory and memory paging have to be mentioned. These are complex topics and the explanation here will stay quite high level. Virtual memory is basically an OS level abstraction to make writing programs easier and to help the OS prevent processes from accessing memory they shouldn't. When a program allocates some memory it gets a pointer to where the memory is stored. This pointer is not a direct location on the physical hardware. Instead, it gets a location in virtual memory and the OS handles where the data is actually stored. This abstraction is useful because if there is a program that has allocated a lot of memory, some of that data can be held in storage until it is needed. This allows the user to have many programs running because some data is being stored on disk instead of in memory. The obvious problem with this is loading from storage is very slow and the OS wants to do that as little as possible, so there are clever techniques in hardware to make sure that addresses that are being referenced more often are held in memory.

When a process is using as much or more virtual memory than the system has physical memory, the OS can spend so much time managing the virtual memory that it becomes unresponsive to requests from processes. When the system enters this state it is called thrashing. PyTorch's DataLoader class provides a way of creating multiple subprocesses to "pre-fetch" data to hold it in memory so that it is ready for when the batch is going to be processed by the GPU. This is useful because it means the CPU can be doing IO

while the GPU is making predictions. However, it seems that for this very large dataset these subprocesses each take up a large amount of virtual memory, around 30-60GB each from observing the processes using the command line tool `htop`. With multiple processes trying to use this much memory it is possible that thrashing was occurring, which would explain the large increase in time taken to process batches. To mitigate this problem this pre-fetching optimisation was disabled.

6.4.2 Random Access

Another problem is that the data was stored on a disk drive, which is a slow form of storage because it has mechanical parts that need to move to read and write data. This kind of storage was chosen because it is cheap and a large amount of storage was needed to store all the data. When using hard drives ideally as many read instructions as possible should be sequential. A hard drive has a moving part called a "read head" that moves around to read the data. One of the factors limiting how quickly the hard drive can read is how far the read head has to move to access the data. Hard drives read more quickly when the data is stored close together in memory, because the parts don't have to move so far. When the data was saved after preprocessing, it was saved in individual files for each image. This was done because PyTorch provides some classes to abstract over loading data in this format, and it allows the batch size to be chosen as a hyperparameter instead of set before training. However, what this means is that for a batch size of 128, 128 files need to be read for every batch. And because the order of files to read is random by design for neural network training, it is 128 files randomly distributed across the hard drive. Random access is the worst case for loading from storage because there is no way of optimising it. To mitigate this problem, another step was added after the preprocessing that pre-batched the training and validation set to make the reads more sequential. The process for this was to randomly select 128 paths from the pool of all images, load them in to memory, turn them into an array 128 images long, then save that array back to disk to be used in training. This should decrease training time because now the batches are stored sequentially in memory. There is a trade-off here because the random order of samples generally speeds up convergence in neural network training. However, the order of the batches is still random so the training process should still have an element of randomness.

6.4.3 Hard Drive Buffer

Most file transfers are quite small, so storage is optimised for short bursts of work. To that end, almost all hard drives have some built in cache memory called the *disk buffer* that is controlled by the microcontroller on the hard drive. The microcontroller moves data that is probably going to be requested into this buffer because the buffer memory is faster than

the storage. Again there are several techniques but a simple and effective one is just to load the blocks of memory immediately after the block that is requested. E.g. if block 10 is requested, the microcontroller might return block 10 and while that is happening, load blocks 11 and 12 into the buffer so that of they are requested they can be returned more quickly. What this means is for reads smaller than the size of the buffer (in this hard drives case 256MB), the hard drive can perform more quickly than if it didn't have the buffer, because the buffer memory is faster than the hard disk. However, once this buffer is full the read speed is limited by the speed of storage, which is why the time to process a batch still increases after a few minutes.

While it isn't exactly clear whether thrashing was occurring it seems likely, as the combination of reducing the degree of multiprocessing and making the reads more sequential combined to reduce the training time noticeably. Ideally when training a neural network the fastest hardware possible should be used, which for storage currently would be an SSD, the fastest being an M.2 NVME SSD. However for large amounts of storage the price difference is quite significant, with 2TB of SSD storage costing the same amount as 12TB of hard disk storage.

Chapter 7

Evaluation

7.1 Data

The test set contains 867231 samples in total.

- CT - 331968
- MR - 118017
- XR - 54391
- PT - 362855

7.2 Hardware

The models were tested on a machine with an Intel(R) Xeon(R) CPU E5-1650 v4 @ 3.60GHz with 6 physical cores (12 threads), 250GB of RAM and two NVidia GeForce GTX 1080Ti's. Towards the end of the project access was gained to a node on the St Andrews HPC Cluster, which has two 16 core processors, 128GB of RAM and two NVidia Tesla V100 GPUs. This node also has much faster storage than the first machine which was beneficial to training times.

7.3 Results

There are a variety of metrics for evaluating the performance of deep learning models. This section will look at how the trained models perform on unseen data.

7.3.1 Accuracy

Accuracy is the rate of correct predictions, i.e.

$$accuracy = \frac{correct}{total} \times 100 \tag{7.1}$$

Accuracy is a useful metric because it gives a rough idea of how well a network is modelling the problem. The accuracy of each network evaluated can be seen in Table 7.1 and for a plot of these results see Figure 7.2.

Model	Accuracy
DenseNet	31%
ResNet18	52%
ResNet34	23%
ResNet50	36%
VGG16	42%
MNASNet	17%

Table 7.1: Table containing the accuracy of various models on the test set. Each model was trained for five epochs.

As can be seen from the results, all the models have very poor results when compared to the accuracies on the train and validate set. This is likely because these models are overfitting, so it would be expected that the regularised model created in the last section would gain higher accuracy as it should have generalised more. However, the VGG16 with extra dropout layers achieved an accuracy of 38%, so it also failed to generalise. Table 7.3 shows the accuracy of the same model with increasing L2 penalty in the training process. The results with 14% are because the model got stuck in a local minima and learned to only predict MR. Clearly the attempts at regularisation did not achieve the desired effect and all models performed worse than the un-regularised VGG16.

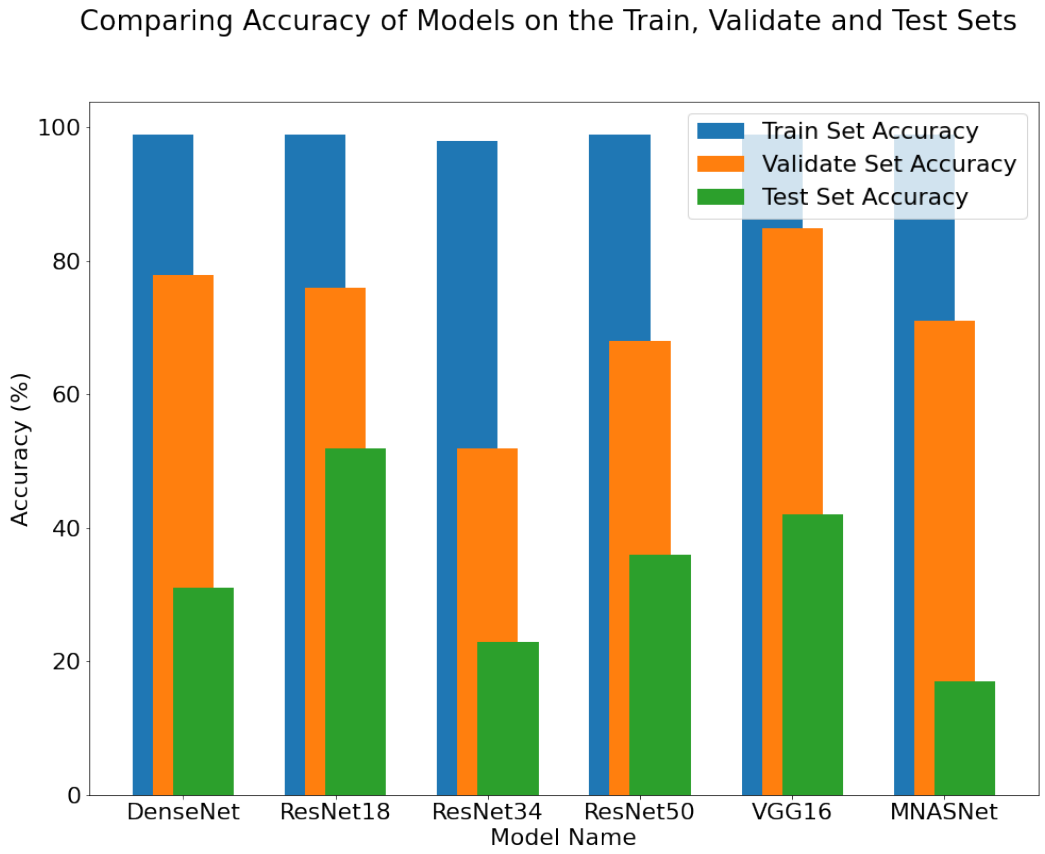


Figure 7.2: Plot showing the accuracies of different models on the train, validate and test set. These results are significantly lower for the test set than the results on the training and validation set.

VGG16 L2 Penalty	Accuracy
0	38%
0.005	18%
0.01	38%
0.015	14%
0.02	14%

Table 7.3: The effect changing L2 penalty in the training process has on accuracy on the test set.

7.3.2 Balanced Accuracy

One way of evaluating how well the model performs across all classes is 'balanced accuracy'. This is a way of equally weighting the performance on each class, which normal accuracy does not do. This was found using scikit-learn's balanced accuracy metric [58, 59], which is defined as the mean of the recall over each class. The results can be seen in Table 7.4 and Figure 7.5.

Model	Balanced Accuracy
Regularised VGG16	32%
DenseNet	27%
ResNet18	36%
ResNet34	18%
ResNet50	33%
VGG16	41%
MNASNet	12%

Table 7.4: Table with the balanced accuracy of different models on the test set.

The balanced accuracy results show a slight drop when compared to accuracy for most models and a larger drop of 16% for the ResNet18 model, which had the highest classification accuracy at 52%. This suggests that the ResNet18 model learned to model the more common classes (CT, MR, PET) and did not model XR well. The VGG16 model that had the highest balanced accuracy on the validate set also had the highest balanced accuracy on the test set, suggesting this model has generalised slightly better. To see if these hypotheses are true the confusion matrices for the models can be viewed.

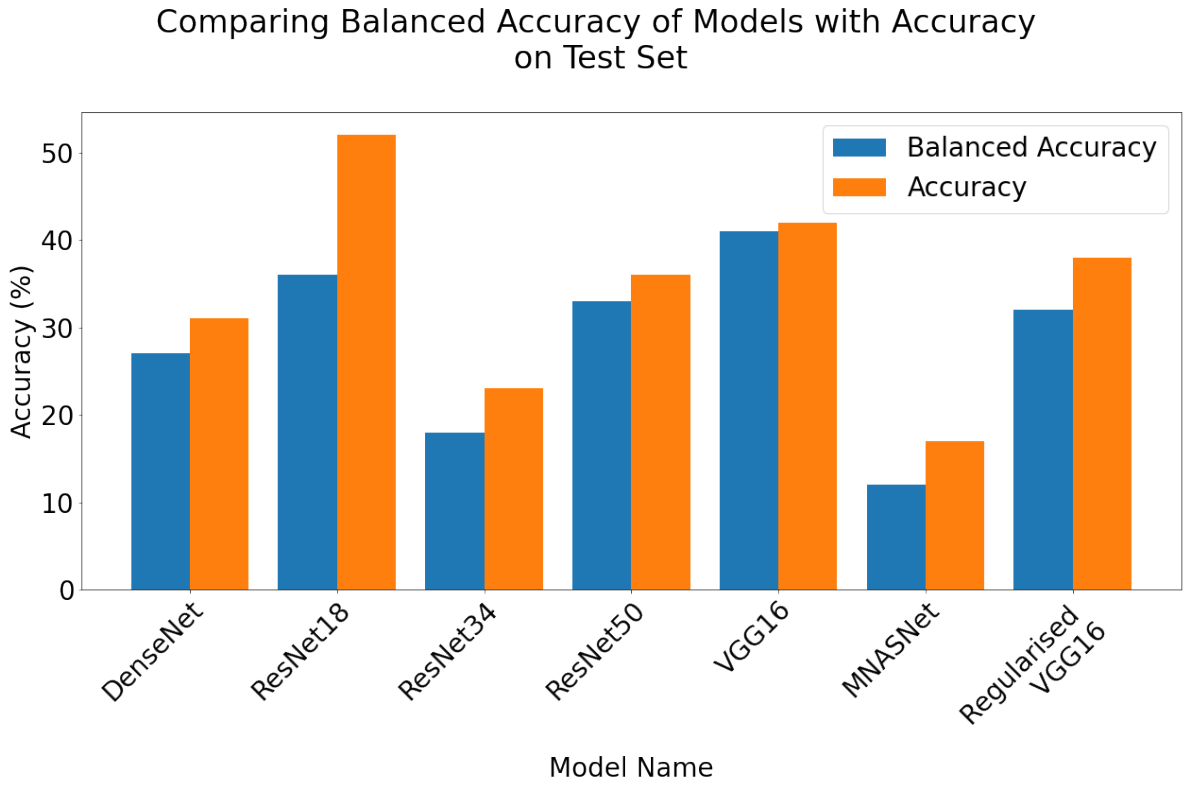


Figure 7.5: Plot showing balanced accuracy of different models on the test set.

7.3.3 Confusion Matrices

A confusion matrix plots the predicted class against actual class for every sample in the dataset. This gives a visualisation of what classes a model is getting right the most, and what classes a model is struggling to differentiate between. First the analysis will focus on the confusion matrices for ResNet18 and VGG16, because they performed best on the accuracy and balanced accuracy scores respectively. To interpret these note that the rows are the predicted labels and the columns are the actual labels. So where the MR row meets the MR column, the entry is the number of predictions for MR that were labelled MR. Where the MR row meets the XR column, the entry is the number of samples that were labelled MR and predicted as XR. The confusion matrix for the ResNet18 can be seen in Figure 7.6, and the matrix for the VGG16 can be seen in Figure 7.7.

The ResNet18 confusion matrix is quite interesting (Figure 7.6). Most of the model's predictions are CT, and it predicts almost all the MR scans as CT. MR and CT scans are possibly the most similar looking out of all the options so that makes sense. However, the model predicts many PET scans as CT and MR. This is quite unusual because PT scans are very distinct from all of the other classes. Figure 7.8 shows how different PET scans look to CT and MR scans on the same body part. Most of the X-ray scans are predicted as being MRI. The higher accuracy is due to the model getting almost all the CT scans correct, and roughly one-third of the PET scans correct. Its balanced accuracy is much lower because it gets very few MR and XR scans correct.

The VGG16 model shows similar strange results with respect to PET scans (Figure 7.7). The model predicts almost all the PET scans as MR. Again, these modalities do not look similar at all. These results suggest the models are overfitting and focussing on features that aren't obvious.

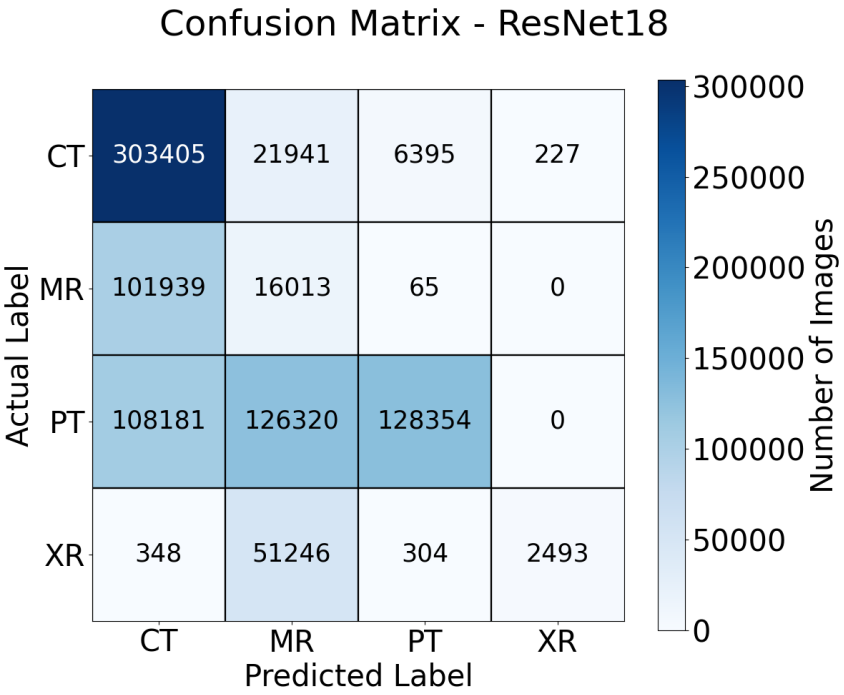


Figure 7.6: The confusion matrix for the ResNet18 model.

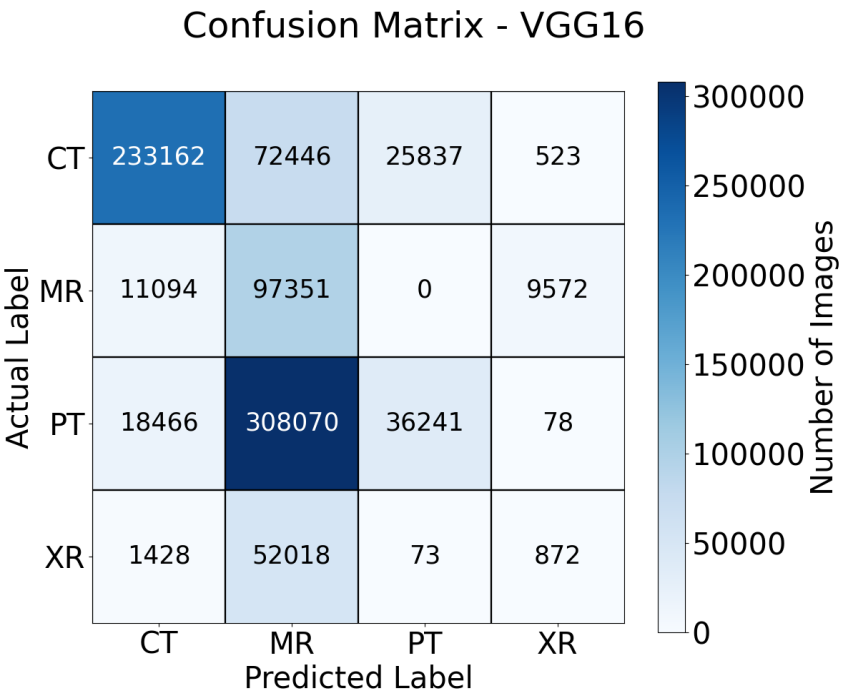


Figure 7.7: The confusion matrix for the VGG16 model.

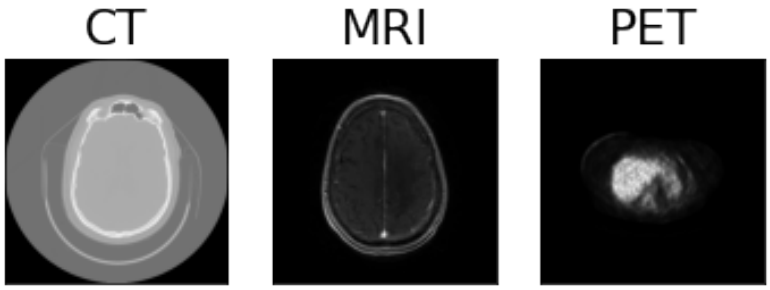


Figure 7.8: Image demonstrating how different PET scans look to CT and MR scans on the same body part, in this case the brain. However, CT and MR scans look similar.

7.3.4 Training and Prediction Times

Figure 7.9 shows the time in hours taken to train each model. This long training times led to a long time spent developing the models because it took many hours to see if a change had improved results. These models were all trained on the machine with the slower hard drive which is part of the reason the models took so long to train. When using the HPC node mentioned at the beginning of the section the training time for the VGG16 model with Dropout layers was 32 hours, a reduction from the previous machine of 23%. This will be due to the faster hard drives and more powerful GPUs in the HPC node.

Prediction time is another useful metric when evaluating neural networks. While most of the computational cost of neural networks is the training process, some models return predictions more quickly which can be beneficial when processing many images. Table 7.10 shows the prediction times for each of the models. The NVidia GTX 1080Ti GPUs are very powerful so the prediction times are almost identical across all the models.

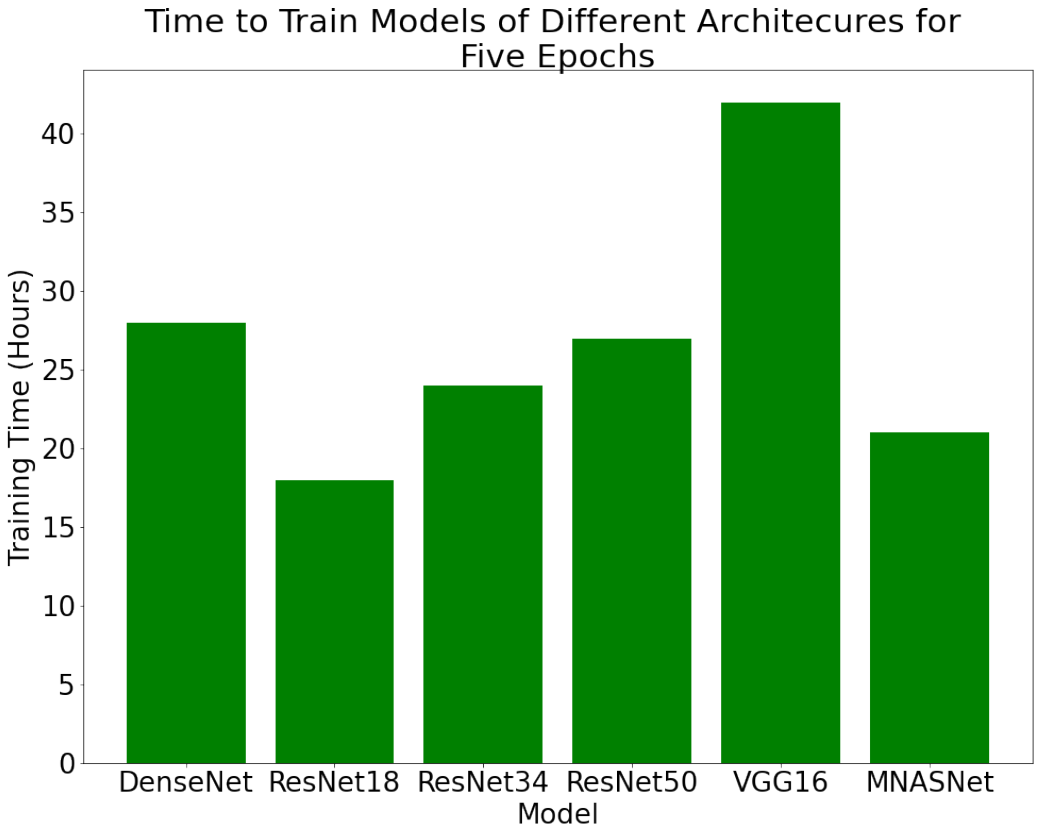


Figure 7.9: Time in hours to train the models for 5 epochs over the training set.

Model	Accuracy	Prediction Time (ms)
DenseNet	31%	0.0015
ResNet18	52%	0.0015
ResNet34	23%	0.0015
ResNet50	36%	0.0015
VGG16	42%	0.0015
MNASNet	17%	0.0015

Table 7.10: Table containing the accuracy of various models on the test set and average time to make a prediction for a single sample. Because the NVidia GTX 1080Ti GPUs are so powerful the prediction times are almost identical.

7.4 Problems with Generalisation

The models trained in this project showed clear signs of overfitting and experimentation to combat this with regularisation was not successful. Because training across this many datasets has not been attempted it is important to reflect on how this happened so that future research can learn from this project.

The possible cause of this problem is the method of rescaling images that was selected. As discussed in the feature scaling section of the Data chapter, not all medical images have a set range of valid pixel values. For a neural network to be able to properly interpret information all the data should be in the same range. This is because the magnitude of the feature has meaning. In this case the magnitude of the feature is the intensity of the pixel. There is no standard pixel range for X-ray, MRI or PET scans which means there is no way of knowing the maximum and minimum possible values the machine that captured these images could detect. For the purposes of this project, the maximum and minimum possible values of the data type used to store the image were used. For example, if each pixel was stored as an 8 bit unsigned int then the minimum value would be 0 and the maximum value would be 255 for that scan. This decision was made based on the assumption that the devices would store the data in a data type that was minimal for the range of values the device could capture. That is, an image wouldn't be stored as a 32 bit signed integer if the values the machine captured couldn't be higher than 1024. However, this assumption appears to have been mistaken with a variety of different data types observed within the same dataset and modality. This could lead to much of the image data being rescaled to very near zero semi-arbitrarily. This would make it much harder for the models to fit the data.

The other option mentioned in the feature scaling section would be to scale based on the maximum and minimum values within the image. This could mean an image that is only intended to have shades of gray in it would have increased contrast, because the lighter shades would move closer to white and the darker ones towards black. However, if the maximum and minimum values are represented often enough in medical scans then this method of rescaling could be a better option than the one used for this practical.

To test this hypothesis the data was preprocessed using the described rescaling. Time constraints meant that only three models could be trained and tested so VGG16, ResNet18 and ResNet50 models were chosen.

7.5 A Different Rescaling Approach

The metrics discussed in the above section suggest that the trained models are not performing as well as could be expected from a deep learning model on a classification task. The low performance is possibly due to the method of feature scaling that was chosen in the preprocessing step of the project. This hypothesis was tested by running the preprocessing code again with a different rescaling technique. To achieve this functionality was added to MedPicPy to rescale each image based on the maximum and minimum pixel value present in the image. Then, all images were rescaled and saved again as discussed in the Data chapter.

The rescaled images were used to train the models with the same hyperparameters as in the Training chapter. To recap, that is training for 5 epochs with a momentum of 0.9, learning rate of 0.1 that was divided by 10 when the error plateaus, batch size of 128. The models were then evaluated on the test set to be compared to the previous models that were trained with the first method of rescaling.

7.5.1 Training and Validation Accuracy

Figures 7.11, 7.12 and 7.13 show the training and validation accuracy curves for the ResNet50, ResNet18 and VGG16 models respectively.

The small gap between the training and validation accuracies suggests that the models are not overfitting. The very high validation accuracy across all models suggests that the method of rescaling was the problem that led to the poor results. To confirm this more metrics can be evaluated. For a visualisation of the effect the rescaling had on the validation accuracy, the validation accuracies at the end of each epoch can be plotted for both ResNet50 models, seen in Figure 7.14.

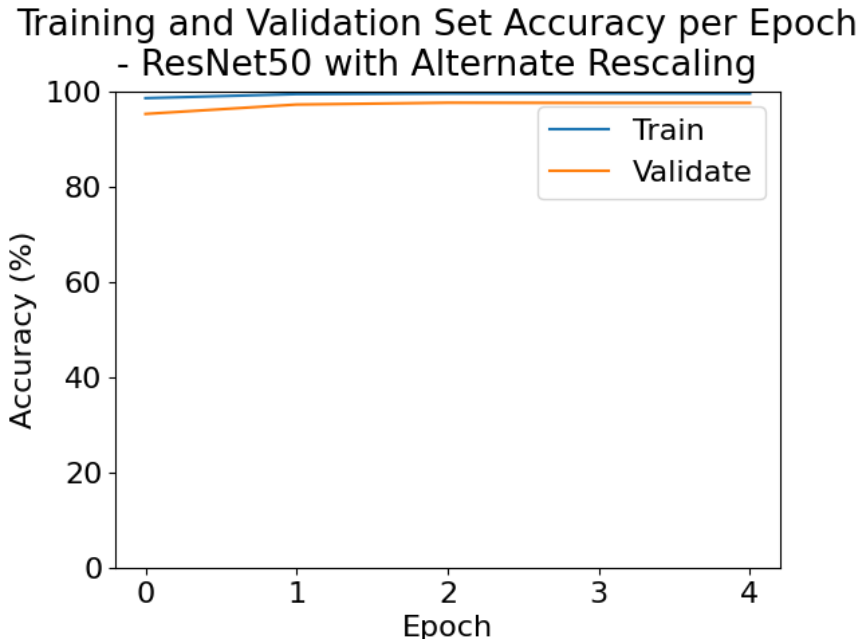


Figure 7.11: Training and validation accuracy for the ResNet50 model, found at the end of each epoch. The small gap between the training and validation accuracies suggests that the model is not overfitting.

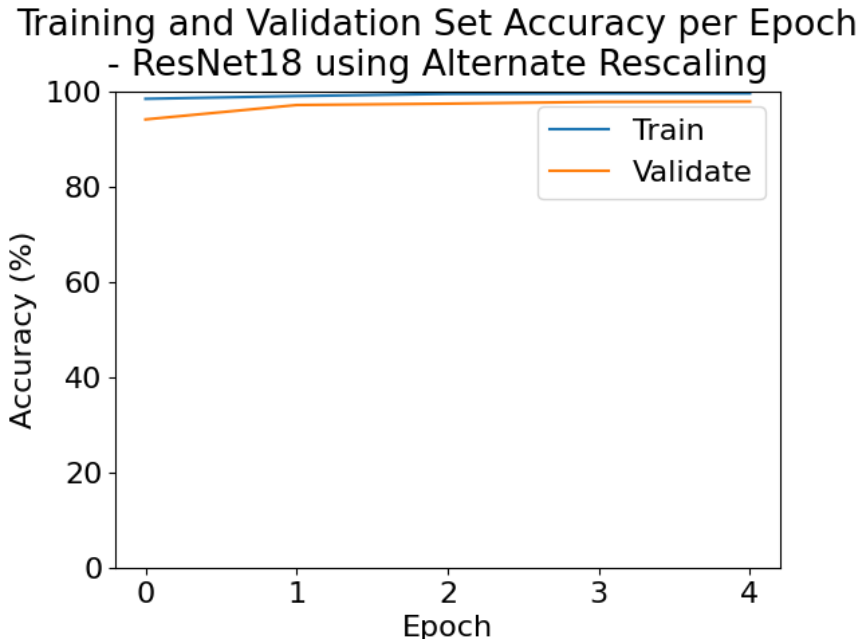


Figure 7.12: Training and validation accuracy for the ResNet18 model, found at the end of each epoch. The small gap between the training and validation accuracies suggests that the model is not overfitting.

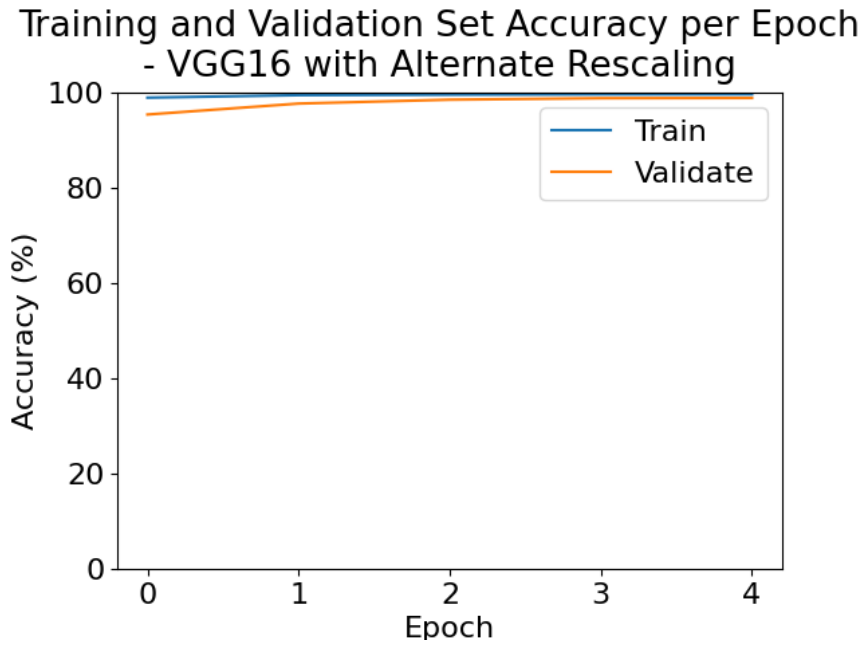


Figure 7.13: Training and validation accuracy for the VGG16 model, found at the end of each epoch. The small gap between the training and validation accuracies suggests that the model is not overfitting.

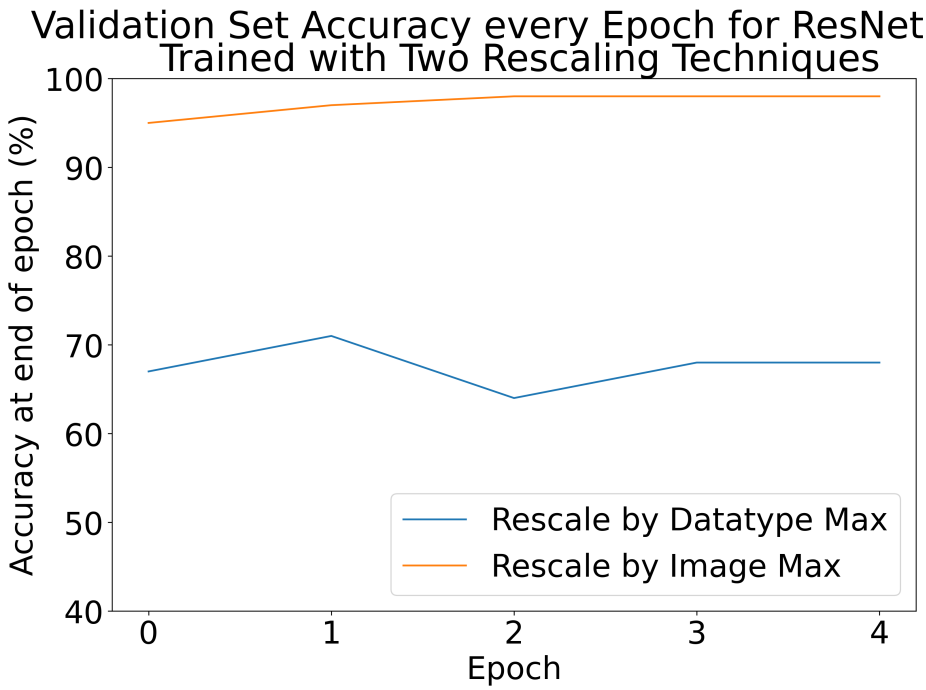


Figure 7.14: Plot showing the validation accuracy at the end of each epoch for a ResNet50 model trained with the two different rescaling techniques. When using images rescaled using the image maximum and minimum the accuracy smoothly increases. When using images rescaled using the datatype maximum and minimum the accuracy jumps up and down from epoch to epoch.

7.5.2 Accuracy

Figure 7.15 shows the accuracy of the three models trained on the images with the new rescaling. These results are in the high 90%'s across the train, validate and test sets which shows that the models have all learned the problem well. Figure 7.16 shows how the accuracies of the models trained on the two different rescaling strategies compare on the test set. These results strongly suggest that the rescaling strategy was the problem that led to the models not being able to generalise.

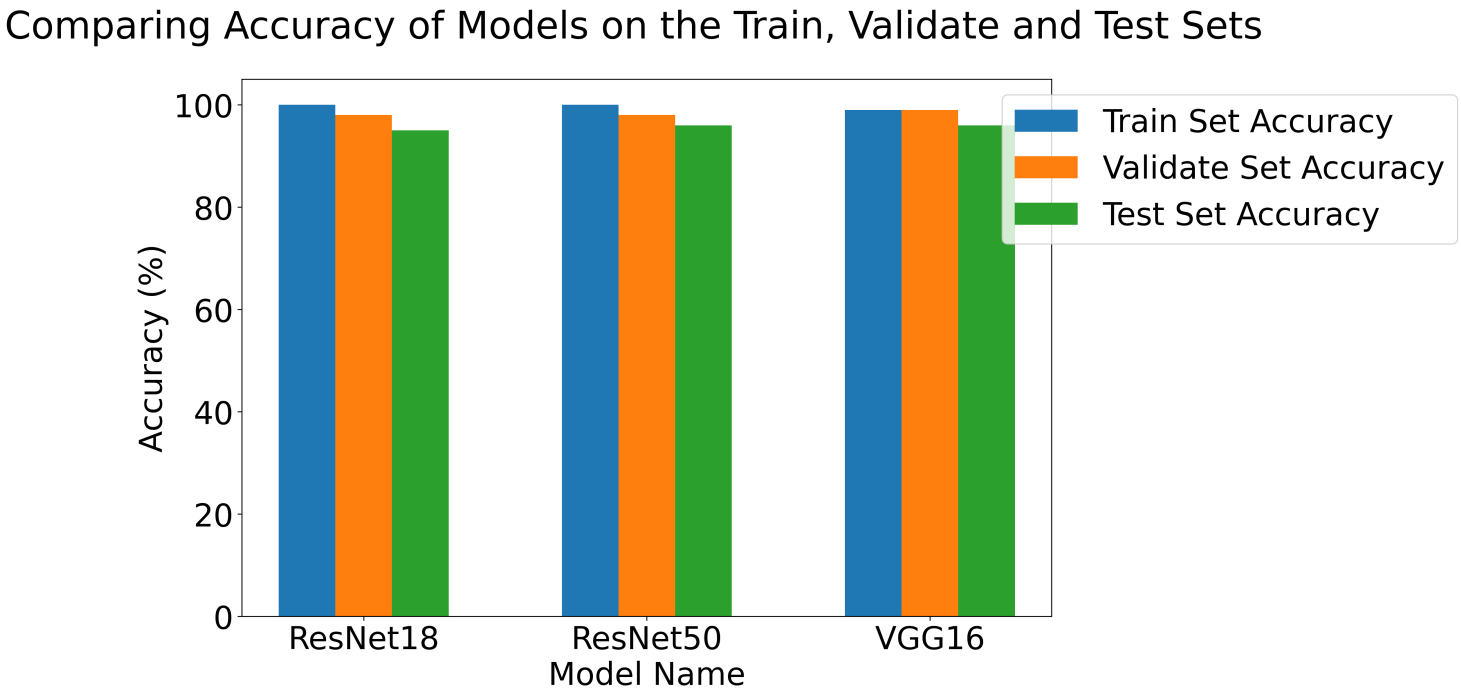


Figure 7.15: Accuracy of 3 models trained with images rescaled using the maximum and minimum values in the image, instead of the datatype.

Table 7.17 shows the accuracy and balanced accuracy for each of the models trained with the alternate rescaling method on the test set. The results clearly show better generalisation than models trained on the previous rescaling method. However, there is a roughly 10% difference in accuracy and balanced accuracy for each of the models. A difference between accuracy and balanced accuracy suggests that the models are getting higher accuracy on the classes with many samples and lower accuracy on the classes with few samples, in this case X-rays. To investigating this the confusion matrices can be inspected.

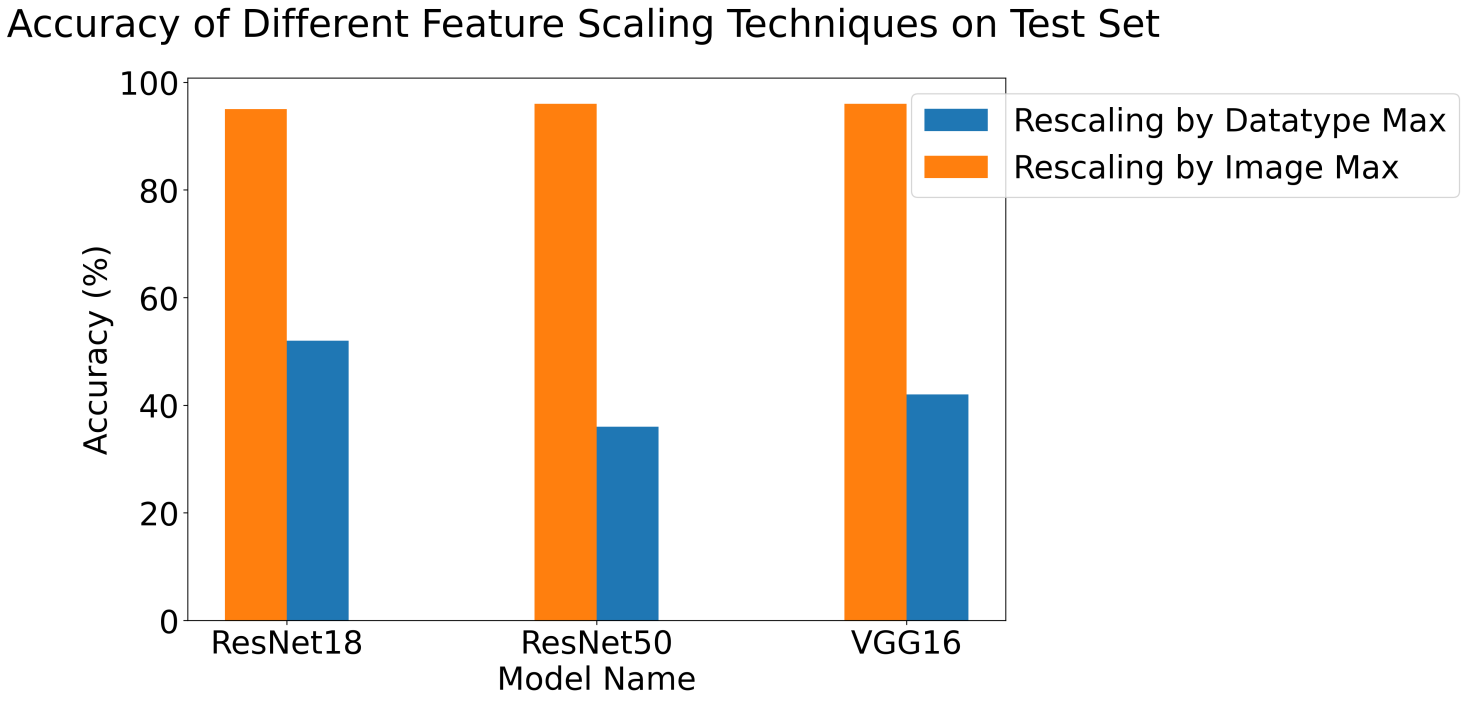


Figure 7.16: Comparing accuracy of models trained using rescaling by max and min for the datatype of the images, vs rescaling by max and min value present in image. Models trained on images rescaled by max and min within the image generalise better.

Model	Accuracy	Balanced Accuracy
VGG16	96%	85%
ResNet18	95%	84%
ResNet50	96%	87%

Table 7.17: Table containing accuracy and balanced accuracy for models trained with the alternate rescaling method. The accuracies and balanced accuracies are similar for all models but the ResNet50 appears to have generalised best, likely because it is the deepest model.

Model	Validate Balanced Accuracy	Test Balanced Accuracy
VGG16	98%	85%
ResNet18	97%	84%
ResNet50	97%	87%

Table 7.18: Table showing the difference in balanced accuracy on the validate and test set. This suggests there are some differences in the train, validate and test datasets that the models did not learn.

7.5.3 Confusion Matrices

It is interesting to view confusion matrices because they show where the models make mistakes and which classes the models perform best on. Figures 7.19, 7.20 and 7.21 show the confusion matrices for the ResNet18, ResNet50 and VGG16 models respectively.

The confusion matrices show that all three models can predict very well for the three well-represented classes (CT, PET and MRI) and less well for X-rays. This isn't a surprising result because the loss function rewards more accurate models, so it is less punished for getting less represented classes wrong. However, there is almost no gap between the accuracy and balanced accuracy on the validate set. This could mean that the validate set is not a perfect simulation of the test set, which would mean that the train-validate-test split could have been better. Table 7.18 shows this difference in balanced accuracy.

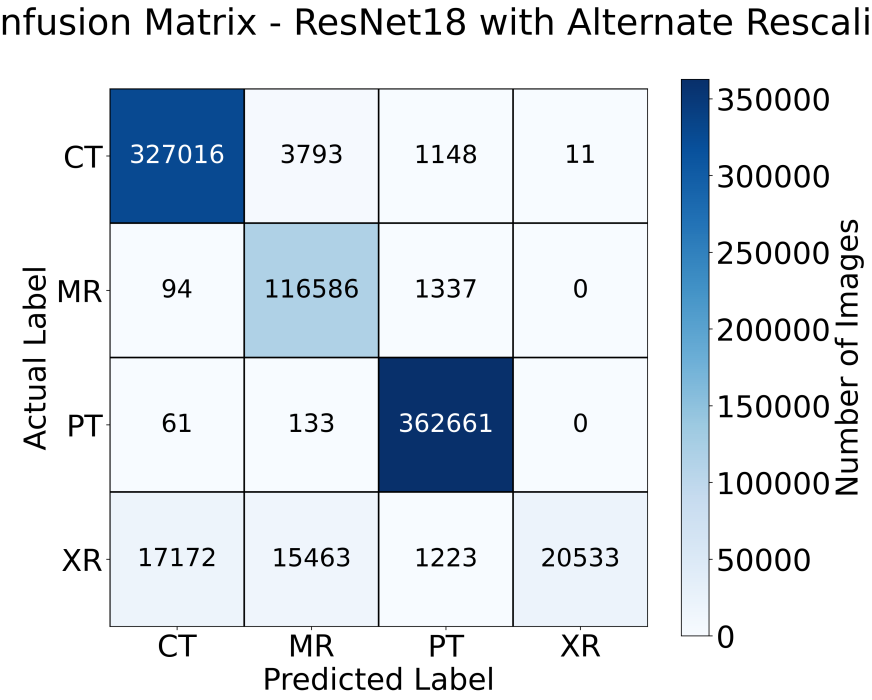


Figure 7.19: Confusion matrix for ResNet18 model trained on images rescaled using the maximum and minimum pixel value in the image.

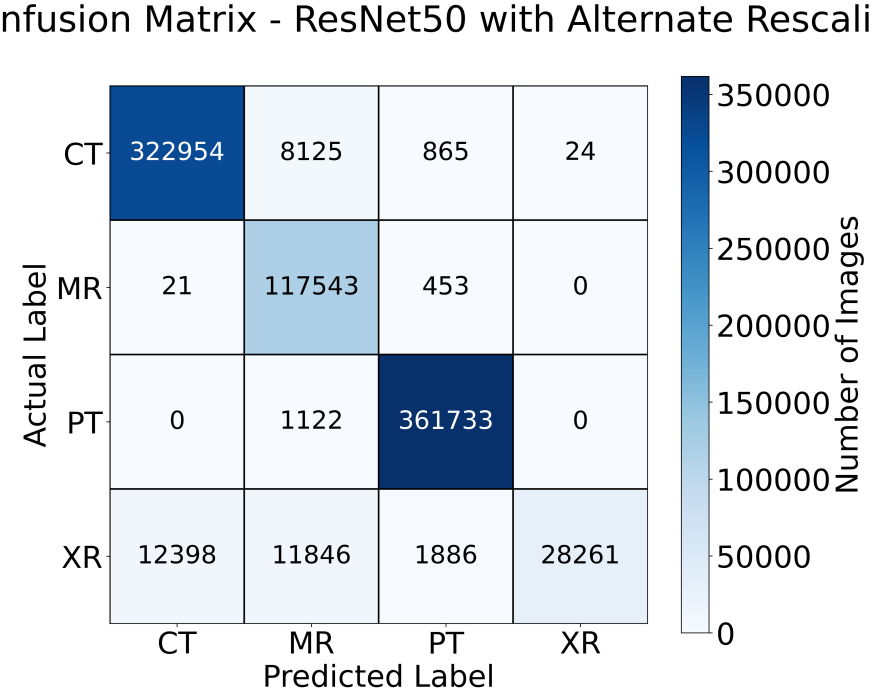


Figure 7.20: Confusion matrix for ResNet50 model trained on images rescaled using the maximum and minimum pixel value in the image

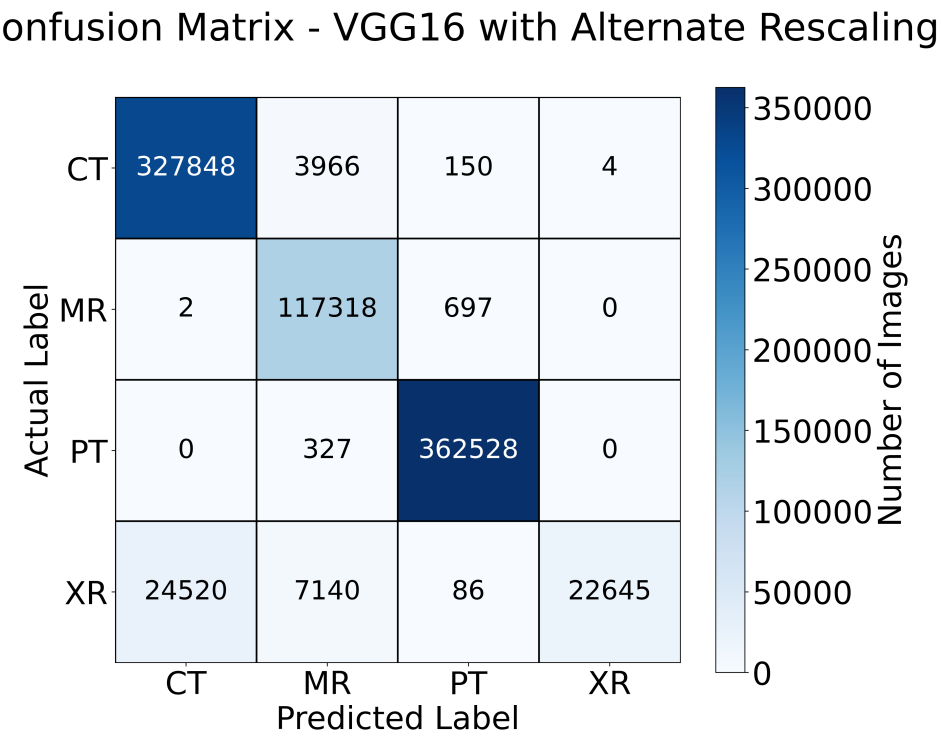


Figure 7.21: Confusion matrix for VGG16 model trained on images rescaled using the maximum and minimum pixel value in the image

7.5.4 Dataset Level Results

Because there are many datasets in the test set, it is interesting to see if mistakes are spread across the datasets or if the model is getting some datasets very wrong. Table 7.22 shows the accuracy of the model on each dataset in the test set and Table 7.23 shows the same for the VGG16 model. It is interesting to note that in both tables the X-ray performance is in the 80-90% range for the Cancer Imaging Archive X-ray datasets, then drops for the MURA and Osteoarthritis Initiative datasets. This is likely because these datasets are bone X-rays, and most of the datasets only contain chest X-rays. Therefore, a better spread of X-ray datasets is needed for the performance of these models to be improved.

Dataset	Modality	Accuracy(%)
CPTAC-LUAD	CT	99
Pelvic-Reference-Data	**CT**	**69**
C4KC-KiTS	CT	99
Anti-PD-1 Lung	CT	93
CPTAC-PDA	CT	97
NaF PROSTATE	CT	98
TCGA-READ	CT	99
QIN-HEADNECK	CT	99
CPTAC-LSCC	CT	99
CPTAC-CCRCC	CT	98
CPTAC-LUAD	MR	99
ISPY1	MR	99
Brain-Tumor-Progression	MR	97
REMBRANDT	MR	99
BraTS20	MR	99
CPTAC-PDA	MR	99
TCGA-READ	MR	100
CPTAC-CCRCC	MR	99
CPTAC-LUAD	PT	99
Anti-PD-1 Lung	PT	99
QIN-HEADNECK	PT	99
CPTAC-PDA	PT	99
NaF PROSTATE	PT	99
CPTAC-LSCC	PT	99
CPTAC-LUAD	XR	94
CPTAC-PDA	XR	88
CPTAC-LSCC	XR	92
CPTAC-CCRCC	XR	85
MURA	**XR**	**25**
Osteo-Arthritis Initiative	**XR**	**73**

Table 7.22: Table containing the accuracy of the ResNet50 model on every dataset in the test set. The bold rows contain results that are noticeably lower than most. Some datasets appear more than once in this table because they contain multiple image modalities.

Dataset	Modality	Accuracy(%)
CPTAC-LUAD	CT	99
Pelvic-Reference-Data	CT	82
C4KC-KiTS	CT	99
Anti-PD-1 Lung	CT	96
CPTAC-PDA	CT	99
NaF PROSTATE	CT	100
TCGA-READ	CT	99
QIN-HEADNECK	CT	99
CPTAC-LSCC	CT	99
CPTAC-CCRCC	CT	99
CPTAC-LUAD	MR	100
ISPY1	MR	99
Brain-Tumor-Progression	MR	91
REMBRANDT	MR	99
BraTS20	MR	98
CPTAC-PDA	MR	99
TCGA-READ	MR	98
CPTAC-CCRCC	MR	99
CPTAC-LUAD	PT	100
Anti-PD-1 Lung	PT	99
QIN-HEADNECK	PT	99
CPTAC-PDA	PT	100
NaF PROSTATE	PT	99
CPTAC-LSCC	PT	100
CPTAC-LUAD	XR	96
CPTAC-PDA	XR	96
CPTAC-LSCC	XR	96
CPTAC-CCRCC	XR	100
MURA	**XR**	**29**
Osteo-Arthritis Initiative	**XR**	**51**

Table 7.23: Table containing the accuracy of the VGG16 model on every dataset in the test set. The bold rows contain results that are noticeably lower than the rest. Some datasets appear more than once in this table because they contain multiple image modalities.

7.6 Discussion

The first iteration of training in this project trained a model to attain an accuracy of 52%. After investigating the cause of this low accuracy, a model was trained that achieved 96% accuracy on the test set.

7.6.1 Comparison to Previous Literature

There have not been any examples of a system that combines medical imaging datasets at this scale, nor have there been any papers that have looked at classifying medical images by modality so there is nothing to directly compare this paper to. However, there are many image classification papers so the results from this project can be put in some context. ImageNet [55] is a very large classification competition with around 1.2 million training images. The highest performing model currently achieves an accuracy of around 88% [60]. However, this is with 1000 classes to choose from which makes the problem more difficult. The CIFAR10 [61] challenge is a classification challenge with only 10 classes and the highest performing model on that dataset achieves 99.7% accuracy [60]. The highest accuracy achieved in this project was 96%. This discrepancy can partly be explained by how much time is spent on tuning hyperparameters for these problems. With the time amount of time left after discovering the rescaling problem it simply wasn't feasible to do a large amount of hyperparameter tuning. Some hyperparameters that could have affected the results are a different image size, a different number of layers, a different optimisation strategy, or any combination of these things. Another possible reason for the difference is in the CIFAR 10 challenge the classes are mostly quite distinct [62] while in the case of this project each class is fairly similar.

The models trained with the second rescaling method had a 10% gap between training and validation accuracy and this was almost entirely due to the difference in performance on X-ray images. This was due to the fact that there were not many bone X-rays in the train and validate set but many in the test set. To improve performance further, more X-ray datasets are needed.

7.6.2 Future Work

One of the secondary objectives mentioned for this project was to look at creating a system that could classify the body part of a medical image of any modality. This was not done because the labelling of the data was insufficient, and this is a problem that any future researcher would struggle with too. The Cancer Imaging Archive mainly hosts datasets

that have some variation of cancer/no-cancer labels. It might be interesting to take a subset of the data used in this project that has labels specifically for cancer of any kind and train a system to recognise cancer/no-cancer. This would be an improvement over current systems that can only do this in one area of the body in one modality. A general cancer prediction project would be interesting because it would allow direct comparison with systems that were only trained on a single dataset.

Another interesting topic in a similar vein as this project would be to take only the 3D scan modalities (CT, MR, PET) and use a 3D CNN to classify based on modality. If this project could be extended further the focus would be on how much of an effect different preprocessing methods have on the results of training, as it was not expected to impact results so greatly. Another extension to this project would be more hyperparameter tuning, experimenting with different sized models and more regularisation. More experimentation would allow for greater confidence in the results.

Transfer learning is taking a network that was trained for a different classification task and only training the model's final few layers to solve a different classification task. Transfer learning works because the low level convolutional and pooling layer feature extraction is transferable between different problems. Networks trained on the ImageNet dataset are often used in medical imaging [63, 64, 65] because individual datasets for medical imaging are often quite small. However, ImageNet contains colour images and medical scans are grayscale. This means to use ImageNet weights the single grayscale channel is duplicated across the 3 input colour channels. This is a bit of a hack, so it would be interesting to see if a model trained across many medical imaging datasets would be useful for transfer learning in medical imaging.

7.6.3 Successes

This project aimed to show the possibility of combining and learning across many datasets. In this respect, the project was a success. Once the rescaling problem was identified a network was trained to achieve 96% classification accuracy on the test set. This shows that the model was able to learn features that predict modality across many medical imaging datasets. The models trained initially achieved very poor results. However, advanced metrics allowed investigation in to what led to these poor results and this improved the accuracy of the models drastically, from 52% to 96%.

Combining datasets at this scale has not been attempted before in literature so having an example of training across datasets from this many sources is useful in itself. This project has also open-sourced all the code it created to ensure that future research can build similar on the work done in this project. It is useful to have clear evidence that rescaling medical images by data type does not work, and that when training across multiple modalities

rescaling using the maximum and minimum values in the image is a "good-enough" heuristic to allow learning to happen.

MedPicPy is an open-source package to simplify the loading of medical images for training deep learning models. It was created by the author of this project as part of a summer project with the University. Because this project involved working with a very large dataset from many sources MedPicPy was used for all the image loading and pre-processing code. This project required features that were not originally part of MedPicPy so several useful features were added including support for rescaling, better error handling and support for working with very large datasets that do not fit in memory. This package will hopefully be of use to any further research done in this area. The changes made for this project are currently in a branch of the main project, this will be merged at a later date `https://github.com/cdmacfadyen/MedPicPy/tree/rescaling`.

This project also highlights some potential problems that could cause difficulties for future researchers and some potential solutions. The main problems were floating-point precision considerations for standardisation, and hard-drive speed considerations for training deep networks with this many images. The code for downloading many datasets from The Cancer Imaging Archive should also be useful to any researchers in the future who want to try training across many datasets. All code for this project has been made open source and is available at this link: `https://github.com/cdmacfadyen/classify-modality`

Chapter 8

Conclusion

Deep learning has many applications in medical imaging. However, research generally focuses on creating models that can achieve high classification accuracy on one specific dataset for one specific task. When multiple datasets are combined it is possible to train a model to perform multiple classification tasks at once. This project combined 102 medical imaging datasets to create one large dataset with 4.5 million images and then used it to train a deep CNN to classify the modality of the images. The highest accuracy achieved on the test set was 96%.

Multiple deep networks were trained and evaluated on the dataset and different regularisation strategies were attempted. All models suffered from overfitting to varying degrees. The main accomplishment was training a deep network on a dataset of this size and demonstrating that the model had learned the task to some degree. Learning over this many medical imaging datasets has not been attempted before. The open-source image-loading package MedPicPy was also given large updates as part of this project to allow it to work with datasets of this size. The report also highlights some problems future researchers may encounter when working with a dataset this large. Specifically, these problems were with slow storage and inconsistent metadata in medical imaging files. The project has shown that when using multiple modalities, rescaling using the maximum and minimum values in each image is a "good-enough" heuristic to allow learning to happen. All the code for this project has been made open-source to allow future work to build on this project. Future work could focus on using this dataset to predict the location of the scan, or create a disease classification model that can predict on multiple modalities.

References

[1] Jeremy Irvin et al. "CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison". In: *CoRR* abs/1901.07031 (2019).

[2] C Daniel Johnson et al. "Accuracy of CT colonography for detection of large adenomas and cancers". In: *New England Journal of Medicine* 359.12 (2008), pp. 1207–1217.

[3] Elizabeth R Gerstner et al. "ACRIN 6684: assessment of tumor hypoxia in newly diagnosed glioblastoma using 18F-FMISO PET and MRI". In: *Clinical Cancer Research* 22.20 (2016), pp. 5079–5086.

[4] Peter Muzi, Michelle Wanner, and Paul Kinahan. "Data From RIDER Lung PET-CT". In: *The Cancer Imaging Archive* (2015).

[5] Sarah S Aboutalib et al. "Deep learning to distinguish recalled but benign mammography images in breast cancer screening". In: *Clinical Cancer Research* 24.23 (2018), pp. 5902–5909.

[6] Aaron J Grossberg et al. "Imaging and clinical data archive for head and neck squamous cell carcinoma patients treated with radiotherapy". In: *Scientific data* 5 (2018), p. 180173.

[7] W Lingle et al. "Radiology data from the cancer genome atlas breast invasive carcinoma [tcga-brca] collection". In: *The Cancer Imaging Archive* (2016).

[8] O Akin et al. "Radiology data from the cancer genome atlas kidney renal clear cell carcinoma [TCGA-KIRC] collection". In: *The Cancer Imaging Archive* (2016).

[9] P. Li et al. "A Large-Scale CT and PET/CT Dataset for Lung Cancer Diagnosis [Data set]." In: *The Cancer Imaging Archive* (2020).

[10] Choyke P et al. "Data From PROSTATE-MRI". In: *The Cancer Imaging Archive* (2016).

[11] S Kirk et al. "Radiology data from the Cancer Genome Atlas Thyroid Cancer [TCGA-THCA] collection". In: *Cancer Imaging Archive. doi* 10 (2016), K9.

[12] National Cancer Institute Clinical Proteomic Tumor Analysis Consortium. "Radiology Data from the Clinical Proteomic Tumor Analysis Consortium Uterine Corpus Endometrial Carcinoma [CPTAC-UCEC] Collection [Data set]." In: *The Cancer Imaging Archive* (2018).

[13] Jerrold L. Boxerman et al. "Early post-bevacizumab progression on contrast-enhanced MRI as a prognostic marker for overall survival in recurrent glioblastoma: results from the ACRIN 6677/RTOG 0625 Central Reader Study". In: *Neuro-Oncology* 15.7 (July 2013), pp. 945–954.

[14] Lale Kostakoglu et al. "A phase II study of 3'-deoxy-3'-18F-fluorothymidine PET in the assessment of early response of breast cancer to neoadjuvant chemotherapy: results from ACRIN 6688". In: *Journal of Nuclear Medicine* 56.11 (2015), pp. 1681–1689.

[15] Heinz-Peter Schlemmer et al. "Global challenges for cancer imaging". In: *Journal of global oncology* 4 (2017), pp. 1–10.

[16] BruceBlaus. *Multipolar Neuron*. https://creativecommons.org/licenses/by/3.0/deed.en. 2013. URL: `https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png` (visited on 11/10/2020).

[17] David H Hubel and Torsten N Wiesel. "Receptive fields of single neurones in the cat's striate cortex". In: *The Journal of physiology* 148.3 (1959), p. 574.

[18] Aphex34. *Convolutional Layer*. https://creativecommons.org/licenses/by-sa/4.0/deed.en. 2015. URL: `https://commons.wikimedia.org/wiki/File:Conv_layer.png` (visited on 12/20/2020).

[19] Aphex34. *Pooling Layer*. https://creativecommons.org/licenses/by-sa/4.0/deed.en. 2015. URL: `https://en.wikipedia.org/wiki/File:Max_pooling.png` (visited on 12/20/2020).

[20] Pranav Rajpurkar et al. "Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning". In: *arXiv preprint arXiv:1711.05225* (2017).

[21] Monika Grewal et al. "Radnet: Radiologist level accuracy using deep learning for hemorrhage detection in ct scans". In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. IEEE. 2018, pp. 281–284.

[22] William Gale et al. "Detecting hip fractures with radiologist-level performance using deep neural networks". In: *arXiv preprint arXiv:1711.06504* (2017).

[23] Stojan Trajanovski et al. "Towards radiologist-level cancer risk assessment in CT lung screening using deep learning". In: *arXiv preprint arXiv:1804.01901* (2018).

[24] Hyewon Choi et al. "Prediction of visceral pleural invasion in lung cancer on CT: deep learning model achieves a radiologist-level performance with adaptive sensitivity and specificity to clinical needs". In: *European Radiology* (2020), pp. 1–11.

[25] Mrinal Haloi, K Raja Rajalakshmi, and Pradeep Walia. "Towards Radiologist-Level Accurate Deep Learning System for Pulmonary Screening". In: *arXiv preprint arXiv:1807.03120* (2018).

[26] Lukas Hirsch et al. "Deep learning achieves radiologist-level performance of tumor segmentation in breast MRI". In: *arXiv preprint arXiv:2009.09827* (2020).

[27] Jameson Merkow et al. "DeepRadiologyNet: Radiologist level pathology detection in CT head images". In: *arXiv preprint arXiv:1711.09313* (2017).

[28] Adrian P Brady. "Error and discrepancy in radiology: inevitable or avoidable?" In: *Insights into imaging* 8.1 (2017), pp. 171–182.

[29] Robert Lindsey et al. "Deep neural network improves fracture detection by clinicians". In: *Proceedings of the National Academy of Sciences* 115.45 (2018), pp. 11591–11596.

[30] Lale Duman et al. "Differentiation between phyllodes tumors and fibroadenomas based on mammographic sonographic and MRI features". In: *Breast Care* 11.2 (2016), pp. 123–127.

[31] Jie-Zhi Cheng et al. "Computer-aided diagnosis with deep learning architecture: applications to breast lesions in US images and pulmonary nodules in CT scans". In: *Scientific reports* 6.1 (2016), pp. 1–13.

[32] Andreas Christe et al. "Computer-aided diagnosis of pulmonary fibrosis using deep learning and CT images". In: *Investigative radiology* 54.10 (2019), p. 627.

[33] Dayong Wang et al. "Deep learning for identifying metastatic breast cancer". In: *arXiv preprint arXiv:1606.05718* (2016).

[34] Mugahed A Al-Antari et al. "A fully integrated computer-aided diagnosis system for digital X-ray mammograms via deep learning detection, segmentation, and classification". In: *International journal of medical informatics* 117 (2018), pp. 44–54.

[35] Wei Chen et al. "Computer-aided grading of gliomas combining automatic segmentation and radiomics". In: *International journal of biomedical imaging* 2018 (2018).

[36] El-Sayed A. El-Dahshan et al. "Computer-aided diagnosis of human brain tumor through MRI: A survey and a new algorithm". In: *Expert Systems with Applications* 41.11 (2014), pp. 5526–5545. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2014.01.021. URL: http://www.sciencedirect.com/science/article/pii/S0957417414000426.

[37] S.G Armato and H MacMahon. "Automated lung segmentation and computer-aided diagnosis for thoracic CT scans". In: *International Congress Series* 1256 (2003). CARS 2003. Computer Assisted Radiology and Surgery. Proceedings of the 17th International Congress and Exhibition, pp. 977–982. ISSN: 0531-5131. DOI: https://doi.org/10.1016/S0531-5131(03)00388-1. URL: http://www.sciencedirect.com/science/article/pii/S0531513103003881.

[38] Kenneth Clark et al. “The Cancer Imaging Archive (TCIA): maintaining and operating a public information repository”. In: *Journal of digital imaging* 26.6 (2013), pp. 1045–1057.

[39] Daoqiang Zhang, Dinggang Shen, Alzheimer’s Disease Neuroimaging Initiative, et al. “Multi-modal multi-task learning for joint prediction of multiple regression and classification variables in Alzheimer’s disease”. In: *NeuroImage* 59.2 (2012), pp. 895–907.

[40] Gaurav Bhatnagar, QM Jonathan Wu, and Zheng Liu. “A new contrast based multimodal medical image fusion framework”. In: *Neurocomputing* 157 (2015), pp. 143–152.

[41] Zhe Guo et al. “Medical image segmentation based on multi-modal convolutional neural network: Study on image fusion schemes”. In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*. IEEE. 2018, pp. 903–907.

[42] Zeyu Jiang et al. “Two-Stage Cascaded U-Net: 1st Place Solution to BraTS Challenge 2019 Segmentation Task”. In: *International MICCAI Brainlesion Workshop*. Springer. 2019, pp. 231–241.

[43] Konstantinos Kamnitsas et al. “Efficient multi-scale 3D CNN with fully connected CRF for accurate brain lesion segmentation”. In: *Medical image analysis* 36 (2017), pp. 61–78.

[44] Sérgio Pereira et al. “Brain tumor segmentation using convolutional neural networks in MRI images”. In: *IEEE transactions on medical imaging* 35.5 (2016), pp. 1240–1251.

[45] Peter D Chang et al. “Fully convolutional neural networks with hyperlocal features for brain tumor segmentation”. In: *Proceedings MICCAI-BRATS Workshop*. 2016, pp. 4–9.

[46] Konstantinos Kamnitsas et al. “Ensembles of multiple models and architectures for robust brain tumour segmentation”. In: *International MICCAI Brainlesion Workshop*. Springer. 2017, pp. 450–462.

[47] Andriy Myronenko. “3D MRI brain tumor segmentation using autoencoder regularization”. In: *International MICCAI Brainlesion Workshop*. Springer. 2018, pp. 311–320.

[48] Pim Moeskops et al. “Deep learning for multi-task medical image segmentation in multiple modalities”. In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2016, pp. 478–486.

[49] Xiaoxuan Liu et al. “A comparison of deep learning performance against health-care professionals in detecting diseases from medical imaging: a systematic review and meta-analysis”. In: *The lancet digital health* 1.6 (2019), e271–e297.

[50] The Cancer Imaging Archive. *TCIA REST API Guide*. 2020. URL: https://wiki.cancerimagingarchive.net/display/Public/TCIA+REST+API+Guide (visited on 12/18/2020).

[51] The Cancer Imaging Archive. *DICOM Modality Abbreviations*. 2012. URL: https://wiki.cancerimagingarchive.net/display/Public/DICOM+Modality+Abbreviations (visited on 11/03/2020).

[52] Nicholas Bien et al. "Deep-learning-assisted diagnosis for knee magnetic resonance imaging: development and retrospective validation of MRNet". In: *PLoS medicine* 15.11 (2018), e1002699.

[53] Pranav Rajpurkar et al. "Mura: Large dataset for abnormality detection in musculoskeletal radiographs". In: *arXiv preprint arXiv:1712.06957* (2017).

[54] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[55] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[56] 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.

[57] Gao Huang et al. "Densely connected convolutional networks. arXiv 2016". In: *arXiv preprint arXiv:1608.06993* 1608 (2018).

[58] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[59] scikit-learn. *Balanced Accuracy Score*. 2020. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.balanced_accuracy_score.html?highlight=accuracy#sklearn.metrics.balanced_accuracy_score (visited on 12/03/2020).

[60] Pierre Foret et al. "Sharpness-Aware Minimization for Efficiently Improving Generalization". In: *arXiv preprint arXiv:2010.01412* (2020).

[61] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[62] Vinod Nair Alex Krizhevsky and Geoffrey Hinton. *The CIFAR-10 dataset*. 2020. URL: https://www.cs.toronto.edu/~kriz/cifar.html (visited on 12/11/2020).

[63] Benjamin Q Huynh, Hui Li, and Maryellen L Giger. "Digital mammographic tumor classification using transfer learning from deep convolutional neural networks". In: *Journal of Medical Imaging* 3.3 (2016), p. 034501.

[64] Huidong Xie et al. "Dual network architecture for few-view CT-trained on ImageNet data and transferred for medical imaging". In: *Developments in X-Ray Tomography XII*. Vol. 11113. International Society for Optics and Photonics. 2019, p. 111130V.

[65] Afonso Menegola et al. "Towards automated melanoma screening: Exploring transfer learning schemes". In: *arXiv preprint arXiv:1609.01228* (2016).

[66] Hugo JWL Aerts et al. "Decoding tumour phenotype by noninvasive imaging using a quantitative radiomics approach". In: *Nature communications* 5.1 (2014), pp. 1–9.

[67] Carlos E Cardenas et al. "Head and neck cancer patient images for determining auto-segmentation accuracy in T2-weighted magnetic resonance imaging through expert manual segmentations". In: *Medical Physics* 47.5 (2020), pp. 2317–2322.

[68] Shivang Desai et al. "Chest imaging representing a COVID-19 positive rural US population". In: *Scientific data* 7.1 (2020), pp. 1–6.

[69] National Cancer Institute Clinical Proteomic Tumor Analysis Consortium. "Radiology Data from the Clinical Proteomic Tumor Analysis Consortium Cutaneous Melanoma [CPTAC-CM] collection". In: *The Cancer Imaging Archive* (2018).

[70] National Cancer Institute Clinical Proteomic Tumor Analysis Consortium. "Radiology Data from the Clinical Proteomic Tumor Analysis Consortium Head and Neck Squamous Cell Carcinoma [CPTAC-HNSCC] Collection". In: *The Cancer Imaging Archive* (2018).

[71] J. L. Tatum et al. "Imaging characterization of a metastatic patient derived model of adenocarcinoma colon: PDMR-997537-175-T [Data set]." In: *The Cancer Imaging Archive* (2020).

[72] Hesham Elhalawani et al. "Matched computed tomography segmentation and demographic data for oropharyngeal cancer radiomics challenges". In: *Scientific data* 4 (2017), p. 170077.

[73] Akshay Jaggi et al. "Stanford DRO Toolkit: digital reference objects for standardization of radiomic features". In: *Tomography* 6.2 (2020), p. 111.

[74] MA Prah et al. "Repeatability of standardized and normalized relative CBV in patients with newly diagnosed glioblastoma". In: *American Journal of Neuroradiology* 36.9 (2015), pp. 1654–1661.

[75] National Cancer Institute Clinical Proteomic Tumor Analysis Consortium. "Radiology Data from the Clinical Proteomic Tumor Analysis Consortium Glioblastoma Multiforme [CPTAC-GBM] collection [Data set]." In: *The Cancer Imaging Archive* (2018).

[76] National Cancer Institute Clinical Proteomic Tumor Analysis Consortium. "Radiology Data from the Clinical Proteomic Tumor Analysis Consortium Sarcomas [CPTAC-SAR] collection [Data set]." In: *The Cancer Imaging Archive* (2018).

[77] Jennifer Yin Yee Kwan et al. "Radiomic biomarkers to refine risk models for distant metastasis in HPV-related oropharyngeal carcinoma". In: *International Journal of Radiation Oncology* Biology* Physics* 102.4 (2018), pp. 1107–1116.

[78] Xia Li et al. "Multiparametric magnetic resonance imaging for predicting pathological response after the first cycle of neoadjuvant chemotherapy in breast cancer". In: *Investigative radiology* 50.4 (2015), pp. 195–204.

[79] Mirabela Rusu et al. "Co-registration of pre-operative CT with ex vivo surgically excised ground glass nodules to define spatial extent of invasive adenocarcinoma on in vivo imaging: a proof-of-concept study". In: *European radiology* 27.10 (2017), pp. 4209–4217.

[80] Petros Kalendralis et al. "FAIR-compliant clinical, radiomics and DICOM metadata of RIDER, Interobserver, Lung1 and Head-Neck1 TCIA collections". In: *Medical Physics* (2020).

[81] KM Schmainda et al. "Glioma DSC-MRI perfusion data with standard imaging and ROIs". In: *The Cancer Imaging Archive. http://doi. org/10.7937 K* 9 (2016).

[82] D Mackin et al. "Court L". In: *Data from Credence Cartridge Radiomics Phantom CT Scans. The Cancer Imaging Archive* (2017).

[83] Jinzhong Yang et al. "Autosegmentation for thoracic radiation treatment planning: A grand challenge at AAPM 2017". In: *Medical physics* 45.10 (2018), pp. 4568–4581.

[84] M. Patnana, S. Patel, and A. Tsao. "Anti-PD-1 Immunotherapy Melanoma Dataset [Data set]." In: *The Cancer Imaging Archive* (2019).

[85] B. J. Erickson et al. "Radiology Data from The Cancer Genome Atlas Uterine Corpus Endometrial Carcinoma (TCGA-UCEC) collection". In: *The Cancer Imaging Archive* (2016).

[86] M. L. Zuley et al. "Radiology Data from The Cancer Genome Atlas Head-Neck Squamous Cell Carcinoma [TCGA-HNSC] collection". In: *The Cancer Imaging Archive* (2016).

[87] Tatiana Bejarano, Mariluz De Ornelas-Couto, and Ivaylo B Mihaylov. "Longitudinal fan-beam computed tomography dataset for head-and-neck squamous cell carcinoma patients". In: *Medical physics* 46.5 (2019), pp. 2526–2537.

[88] Rachel B Ger et al. "Synthetic head and neck and phantom images for determining deformable image registration accuracy in magnetic resonance imaging". In: *Medical physics* 45.9 (2018), pp. 4315–4321.

[89] Martin Vallieres et al. "Radiomics strategies for risk assessment of tumour failure in head-and-neck cancer". In: *Scientific reports* 7.1 (2017), pp. 1–14.

[90] Zeynettin Akkus et al. "Predicting deletion of chromosomal arms 1p/19q in low-grade gliomas from MR images using machine intelligence". In: *Journal of digital imaging* 30.4 (2017), pp. 469–476.

[91] Rebecca Sawyer Lee et al. "A curated mammography data set for use in computer-aided detection and diagnosis research". In: *Scientific data* 4 (2017), p. 170177.

[92] Marios A Gavrielides et al. "A resource for the assessment of lung nodule size estimation methods: database of thoracic CT scans of an anthropomorphic phantom". In: *Optics express* 18.14 (2010), p. 15244.

[93] J Kalpathy-Cramer et al. "QIN multi-site collection of Lung CT data with nodule segmentations". In: *Cancer Imaging Arch* 10 (2015), K9.

[94] Sunny Jansen and Terry. Van Dyke. "TCIA Mouse-Astrocytoma Collection". In: *The Cancer Imaging Archive* (2015).

[95] B Albertina et al. "Radiology data from the cancer genome atlas lung adenocarcinoma [tcga-luad] collection". In: *The Cancer Imaging Archive* (2016).

[96] M Linehan et al. "Radiology data from the cancer genome atlas cervical kidney renal papillary cell carcinoma [KIRP] collection". In: *Cancer Imaging Arch* (2016).

[97] B Erickson et al. "Radiology Data from The Cancer Genome Atlas Liver Hepatocellular Carcinoma [TCGA-LIHC] collectionThe". In: *Cancer Imaging Archive* (2016).

[98] Ralph B Puchalski et al. "An anatomic transcriptional atlas of human glioblastoma". In: *Science* 360.6389 (2018), pp. 660–663.

[99] Asha Singanamalli et al. "Identifying in vivo DCE MRI markers associated with microvessel architecture and gleason grades of prostate cancer". In: *Journal of Magnetic Resonance Imaging* 43.1 (2016), pp. 149–158.

[100] Robert J Toth et al. "Histostitcher™: An informatics software platform for reconstructing whole-mount prostate histology using the extensible imaging platform framework". In: *Journal of Pathology Informatics* 5 (2014).

[101] Gaoyu Xiao et al. "Determining histology-MRI slice correspondences for defining MRI-based disease signatures of prostate cancer". In: *Computerized Medical Imaging and Graphics* 35.7-8 (2011), pp. 568–578.

[102] Jonathan Chappelow et al. "Elastic registration of multimodal prostate MRI and histology via multiattribute combined mutual information". In: *Medical Physics* 38.4 (2011), pp. 2005–2018.

[103] ML Zuley, R Jarosz, BF Drake, et al. "Radiology data from the cancer genome atlas prostate adenocarcinoma [TCGA-PRAD] collection". In: *The Cancer Imaging Archive. Available online: http://doi. org/10.7937 K* 9 (2016).

[104] David Newitt and Nola Hylton. "Single site breast DCE-MRI data and segmentations from patients undergoing neoadjuvant chemotherapy". In: *The Cancer Imaging Archive* 2 (2016).

[105] Daniel Barboriak. *Data From RIDER NEUR MRI. The Cancer Imaging Archive.* 2015.

[106] Martin Vallières et al. "A radiomics model from joint FDG-PET and MRI texture features for the prediction of lung metastases in soft-tissue sarcomas of the extremities". In: *Physics in Medicine and Biology* 60.14 (2015), p. 5471.

[107] S Jansen et al. "TCIA Mouse-Mammary Collection". In: *The Cancer Imaging Archive* (2015).

[108] C Roche, E Bonaccio, and J Filippini. "cited 2019 18/01/2019". In: *Radiology data from The Cancer Genome Atlas Sarcoma collection. The Cancer Imaging Archive 2016* (2016).

[109] Olya Grove et al. "Quantitative computed tomographic descriptors associate tumor shape complexity and intratumor heterogeneity with prognosis in lung adenocarcinoma". In: *PloS one* 10.3 (2015), e0118261.

[110] FR Lucchesi and ND Aredes. *Radiology data from The Cancer Genome Atlas Cervical Squamous Cell Carcinoma and Endocervical Adenocarcinoma (TCGA-CESC) collection. The Cancer Imaging Archive*. 2016.

[111] C. Holback et al. *Radiology Data from The Cancer Genome Atlas Ovarian Cancer [TCGA-OV] collection*. 2016.

[112] Nancy Pedano et al. "Radiology data from the cancer genome atlas low grade glioma [TCGA-LGG] collection". In: *The Cancer Imaging Archive* 2 (2016).

[113] Reinhard R. Beichel et al. "Data From QIN PET Phantom". In: *The Cancer Imaging Archive* (2015).

[114] Wei Huang et al. "Variations of dynamic contrast-enhanced magnetic resonance imaging in evaluation of breast cancer therapy response: a multicenter data analysis challenge". In: *Translational oncology* 7.1 (2014), p. 153.

[115] Binsheng Zhao. "Data From Lung Phantom". In: *The Cancer Imaging Archive* (2015).

[116] M. W. Linehan et al. "Radiology Data from The Cancer Genome Atlas Kidney Chromophobe [TCGA-KICH] collection". In: *The Cancer Imaging Archive* (2016).

[117] Scarpace L et al. "Radiology Data from The Cancer Genome Atlas Glioblastoma Multiforme [TCGA-GBM] collection". In: *The Cancer Imaging Archive* (2016).

[118] Samuel G Armato III et al. "LUNGx Challenge for computerized lung nodule classification". In: *Journal of Medical Imaging* 3.4 (2016).

[119] G Litjens, J Futterer, and H Huisman. *Data from prostate-3t: the cancer imaging archive*. 2015.

[120] B Nicolas Bloch, Ashali Jain, and C Carl Jaffe. "Data From PROSTATE-DIAGNOSIS". In: *The Cancer Imaging Archive. Available online: http://doi. org/10.7937 K* 9 (2015).

[121] Peter Muzi, Michelle Wanner, and Paul Kinahan. "Data From RIDER PHANTOM PET-CT". In: *The Cancer Imaging Archive* (2015).

[122] Binsheng Zhao et al. "Evaluating variability in tumor measurements from same-day repeat CT scans of patients with non–small cell lung cancer". In: *Radiology* 252.1 (2009), pp. 263–272.

[123] Edward F Jackson et al. "Magnetic resonance assessment of response to therapy: tumor change measurement, truth data and error sources". In: *Translational Oncology* 2.4 (2009), p. 211.

[124] Charles R Meyer et al. *Data from RIDER-Breast-MRI. The cancer imaging archive*. 2015.

[125] Holger R Roth et al. "Deeporgan: Multi-level deep convolutional networks for automated pancreas segmentation". In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2015, pp. 556–564.

[126] Holger R Roth et al. "A new 2.5 D representation for lymph node detection using random sets of deep convolutional neural network observations". In: *International conference on medical image computing and computer-assisted intervention*. Springer. 2014, pp. 520–527.

[127] F. R. Lucchesi and N. D Aredes. "Radiology Data from The Cancer Genome Atlas Esophageal Carcinoma [TCGA-ESCA] collection". In: *The Cancer Imaging Archive* (2016).

[128] FR Lucchesi and ND Aredes. "Radiology Data from The Cancer Genome Atlas Stomach Adenocarcinoma [TCGA-STAD] collection, 2016". In: *The Cancer Imaging Archive* 10 (), K9.

[129] Samuel G Armato III et al. "The lung image database consortium (LIDC) and image database resource initiative (IDRI): a completed reference database of lung nodules on CT scans". In: *Medical physics* 38.2 (2011), pp. 915–931.

[130] Charles Fenimore et al. "Data from QIBA CT-1C". In: *The Cancer Imaging Archive* (2016).

[131] Xiaosong Wang et al. "Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2097–2106.

[132] National Cancer Institute Clinical Proteomic Tumor Analysis Consortium. "Radiology Data from the Clinical Proteomic Tumor Analysis Consortium Clear Cell Renal Cell Carcinoma [CPTAC-CCRCC] collection [Data set]." In: *The Cancer Imaging Archive* (2018).

[133] Nicholas Heller et al. "The state of the art in kidney and kidney tumor segmentation in contrast-enhanced ct imaging: Results of the kits19 challenge". In: *Medical Image Analysis* 67 (2019), p. 101821.

[134] National Cancer Institute Clinical Proteomic Tumor Analysis Consortium. "Radiology Data from the Clinical Proteomic Tumor Analysis Consortium Lung Squamous Cell Carcinoma [CPTAC-LSCC] Collection [Data set]." In: *The Cancer Imaging Archive* (2018).

[135] Taylor R Moen et al. “Low Dose CT Image and Projection Dataset”. In: *Medical Physics* (2020).

[136] National Cancer Institute Clinical Proteomic Tumor Analysis Consortium. “Radiology Data from the Clinical Proteomic Tumor Analysis Consortium Lung Adenocarcinoma [CPTAC-LUAD] collection [Data set].” In: *The Cancer Imaging Archive* (2018).

[137] National Cancer Institute Clinical Proteomic Tumor Analysis Consortium. “Radiology Data from the Clinical Proteomic Tumor Analysis Consortium Pancreatic Ductal Adenocarcinoma [CPTAC-PDA] Collection [Data set]..” In: *The Cancer Imaging Archive* (2018).

[138] A Yorke et al. “A Statistically Characterized Reference Data Set for Image Registration of Pelvis Using Combinatorial Affine Registration Optimization”. In: *MEDICAL PHYSICS*. Vol. 46. 6. WILEY 111 RIVER ST, HOBOKEN 07030-5774, NJ USA. 2019, E340–E340.

[139] P. Madhavi, S. Patel, and A. S. Tsao. “Data from Anti-PD-1 Immunotherapy Lung [Data set].” In: *The Cancer Imaging Archive* (2019).

[140] Nola M Hylton et al. “Neoadjuvant chemotherapy for breast cancer: functional tumor volume by MR imaging predicts recurrence-free survival—results from the ACRIN 6657/CALGB 150007 I-SPY 1 TRIAL”. In: *Radiology* 279.1 (2016), pp. 44–55.

[141] Andriy Fedorov et al. “DICOM for quantitative imaging biomarker development: a standards based approach to sharing clinical data and structured PET/CT analysis results in head and neck cancer research”. In: *PeerJ* 4 (2016), e2057.

[142] S. Kirk et al. “Radiology Data from The Cancer Genome Atlas Rectum Adenocarcinoma [TCGA-READ] collection”. In: *The Cancer Imaging Archive* (2016).

[143] Karen A Kurdziel et al. “The kinetics and reproducibility of 18F-sodium fluoride for oncology using current PET camera technology”. In: *Journal of Nuclear Medicine* 53.8 (2012), pp. 1175–1184.

[144] Lisa Scarpace et al. “Data from REMBRANDT”. In: *The Cancer Imaging Archive* 10 (2015), K9.

Appendices

Appendix A

Ethics Approval Form

University of St Andrews | FOUNDED 1413

School of Computer Science Ethics Committee

11 December 2020

Dear Craig,

Thank you for submitting your ethical application which was considered at the School Ethics Committee meeting on Wednesday 2nd December.

The School of Computer Science Ethics Committee, acting on behalf of the University Teaching and Research Ethics Committee (UTREC), has approved this application:

Approval Code:	CS15171	**Approved on:**	11.12.20	**Approval Expiry:**	11.12.25
Project Title:	Classifying Medical Scans by Modality using Deep Learning				
Researcher(s):	Craig MacFadyen				
Supervisor(s):	Dr David Harris-Birtill				

The following supporting documents are also acknowledged and approved:

1. Ethical Application Form

Approval is awarded for 5 years, see the approval expiry data above.

If your project has not commenced within 2 years of approval, you must submit a new and updated ethical application to your School Ethics Committee.

If you are unable to complete your research by the approval expiry date you must request an extension to the approval period. You can write to your School Ethics Committee who may grant a discretionary extension of up to 6 months. For longer extensions, or for any other changes, you must submit an ethical amendment application.

You must report any serious adverse events, or significant changes not covered by this approval, related to this study immediately to the School Ethics Committee.

Approval is given on the following conditions:

- that you conduct your research in line with:
  - the details provided in your ethical application
  - the University's Principles of Good Research Conduct
  - the conditions of any funding associated with your work
- that you obtain all applicable additional documents (see the 'additional documents' webpage for guidance) before research commences.

You should retain this approval letter with your study paperwork.

Yours sincerely,

Wendy Boyter
Ethics Administrator

cc.

Appendix B

Code Details

Code Directory Structure

All the code for this project can be found in the `src/` directory. All the training data and large results files have been removed, leaving just the code and the images created to visualise the results. The code has been commented to explain the purpose and function of all scripts.

```
src/
├── tcia-download
│   └── metadata
│       └── processed
├── train
│   ├── notebooks
│   ├── preprocessing
│   │   └── images
│   └── results
│       └── images
├── test
│   ├── notebooks
│   └── results
│       └── images
├── visualisation
└── other-datasets
```

`src/tcia-download/` contains the code for the download of all of the Cancer Imaging Archive data. `src/train/` contains the code for training the models, and all of the classes

for defining the structure of the models. `src/test/` contains the code for testing the models and evaluating the predictions made by the models. `src/visualisation/` contains a notebook that creates some visualisations for the report. `src/other-datasets/` contains some code that counts the number of images present in the non-TCIA datasets that were downloaded.

src/tcia-download/

```
src/tcia-download/
├── compare_api_and_links.py
├── get_collection_metadata.py
├── metadata
│   ├── create-train-val-test-split.ipynb
│   ├── processed
│   └── visualise-metadata.ipynb
├── tciaclient.py
└── tcia_download_script_sync.py
```

`src/tcia_download` contains the scripts required to download TCIA datasets en-masse. `get_collection_metadata.py` downloads metadata about each dataset hosted by TCIA. These details are PatientIDs, Number of Images, Modalities of images etc. This metadata is needed to run the download script. The Jupyter notebooks in `src/tcia_download/metadata` process this metadata and create a train-validate-test split. All of the code for deciding which datasets go into the train validate and test split can be seen in `create-train-val-test-split.ipynb` Then the download script `tcia_download_script_sync.py` can be run with the arguments "train", "validate", and "test" to download the data for each of the splits. **This takes take a long time.**

src/train/

```
src/train
├── train.py
├── evaluation.py
├── mvp.py
├── basic_model.py
├── data_loader.py
├── densenet.py
├── mnas_net.py
├── run_all_models.py
```

```
├── vgg.py
├── resnet.py
├── notebooks
│   └── nn-section-graphs.ipynb
├── preprocessing
│   ├── batch_data.py
│   ├── images
│   └── preprocessing.py
├── results
│   └── images
└── visualise_results.py
```

`src/train/preprocessing.py` resizes and rescales all of the data and saves it in '.npy' format for ease of use in the training process. `src/train/batch_data.py` is a script that creates batches of 128 images and saves them as one file for faster hard drive reads.

`src/train/train.py` contains a script for training deep CNNs on the created dataset. The other files are model declarations adapted from the models hosted on the PyTorch GitHub to take one channel images. `src/train/visualise_results.py` creates visualisations of the training metrics created in this process.

src/test/

```
src/test
├── load_model_test.py
├── notebooks
│   ├── plot_test_results.ipynb
│   └── test_results_new_rescaled.ipynb
├── results
│   └── images
├── test_loader.py
├── visualise_all.py
└── visualise_test_results.py
```

`src/test/` contains scripts for testing the networks. The training code saves the networks and the testing code loads the saved weights to evaluate their performance on the test set. `src/test/test_loader.py` evaluates a given model on the test set and saves the results to be analysed. `src/test/test_loader.py` does the visualisation and saves the images.

Appendix C

List of All Datasets Used

Name	Dataset Webpage	Citations	Split
ACRIN-DSC-MR-Brain	https://doi.org/10.7937/tcia.2019.zr1pjf4i	[13]	Train
Head-Neck-Radiomics-HN1	https://doi.org/10.7937/tcia.2019.8kap372n	[66]	Train
Lung-PET-CT-Dx	https://doi.org/10.7937/TCIA.2020.NNC2-0461	[9]	Train
AAPM RT-MAC Grand Challenge 2019	https://doi.org/10.7937/tcia.2019.bcfjqfqb	[67]	Train
COVID-19-AR	https://doi.org/10.7937/tcia.2020.py71-5978	[68]	Train
CPTAC-CM	https://doi.org/10.7937/K9/TCIA.2018.ODU24GZE	[69]	Train
CPTAC-HNSCC	https://doi.org/10.7937/K9/TCIA.2018.UW45NH81	[70]	Train
PDMR-997537-175-T	https://doi.org/10.7937/TCIA.2020.BRY9-4N29	[71]	Train
PDMR-292921-168-R	https://doi.org/10.7937/TCIA.2020.PCAK-8Z10	[71]	Train
PDMR-425362-245-T	https://doi.org/10.7937/TCIA.2020.7YRS-7J97	[71]	Train
HNSCC	https://doi.org/10.7937/k9/tcia.2020.a8sh-7363	[6, 72]	Train
DRO Toolkit	https://doi.org/10.7937/t062-8262	[73]	Train
QIN GBM Treatment Response	https://doi.org/10.7937/K9/TCIA.2016.nQF4gpn2	[74]	Train
CPTAC-GBM	https://doi.org/10.7937/K9/TCIA.2018.3RJE41Q1	[75]	Train
CPTAC-SAR	https://doi.org/10.7937/TCIA.2019.9bt23r95	[76]	Train
CPTAC-UCEC	https://doi.org/10.7937/K9/TCIA.2018.3R3JUISW	[12]	Train

OPC-Radiomics	https://doi.org/10.7937/tcia.2019.8dho2gls	[77]	Train
Acrin-FLT-Breast (ACRIN 6688)	https://doi.org/10.7937/K9/TCIA.2017.ol20zmxg	[14]	Train
QIN-Breast	https://doi.org/doi:10.7937/K9/TCIA.2016.21JUebH0	[78]	Train
Lung Fused-CT-Pathology	https://doi.org/10.7937/K9/TCIA.2018.SMT36LPN	[79]	Train
NSCLC-Radiomics	https://doi.org/10.7937/K9/TCIA.2015.PF0M9REI	[66]	Train
NSCLC-Radiomics-Interobserver1	https://doi.org/10.7937/tcia.2019.cwvlpd26	[80, 66]	Train
PDMR-BL0293-F563	https://doi.org/10.7937/tcia.2019.b6u7wmqw	[71]	Train
QIN-BRAIN-DSC-MRI	https://doi.org/doi:10.7937/K9/TCIA.2016.5DI84Js8	[81]	Train
CC-Radiomics-Phantom	https://doi.org/10.7937/K9/TCIA.2017.zuzrml5b	[82]	Train
CC-Radiomics-Phantom-2	https://doi.org/10.7937/TCIA.2019.4l24tz5g	[82]	Train
CC-Radiomics-Phantom-3	https://doi.org/10.7937/tcia.2019.j71i4fah	[82]	Train
LCTSC	https://doi.org/10.7937/K9/TCIA.2017.3r3fvz08	[83]	Train
Anti-PD-1 MELANOMA	https://doi.org/10.7937/tcia.2019.1ae0qtcu	[84]	Train
TCGA-UCEC	https://doi.org/10.7937/K9/TCIA.2016.GKJ0ZWAC	[85]	Train
TCGA-HNSC	https://doi.org/10.7937/K9/TCIA.2016.LXKQ47MS	[86]	Train
HNSCC-3DCT-RT	https://doi.org/10.7937/K9/TCIA.2018.13upr2xf	[87]	Train
MRI-DIR	https://doi.org/10.7937/K9/TCIA.2018.3f08iejt	[88]	Train
Head-Neck-PET-CT	https://doi.org/10.7937/K9/TCIA.2017.8oje5q00	[89]	Train
LGG-1p19qDeletion	https://doi.org/10.7937/K9/TCIA.2017.dwehtz9v	[90]	Train
CBIS-DDSM	https://doi.org/10.7937/K9/TCIA.2016.7O02S9CY	[91]	Train
Phantom FDA	https://doi.org/10.7937/K9/TCIA.2015.ORBJKMUX	[92]	Train
QIN LUNG CT	https://doi.org/10.7937/K9/TCIA.2015.NPGZYZBZ	[93]	Train
Mouse-Astrocytoma	https://doi.org/10.7937/K9TCIA.2017.SGW7CAQW	[94]	Train
TCGA-LUSC	https://doi.org/10.7937/K9/TCIA.2016.TYGKKFMQ	[95]	Train
TCGA-LUAD	https://doi.org/10.7937/K9/TCIA.2016.JGNIHEP5	[95]	Train

TCGA-KIRP	https://doi.org/10.7937/K9/TCIA.2016.ACWOGBEF	[96]	Train
TCGA-LIHC	https://doi.org/10.7937/K9/TCIA.2016.IMMQW8UQ	[97]	Train
IvyGAP	https://doi.org/10.7937/K9/TCIA.2016.XLwaN6nL	[98]	Train
Prostate Fused-MRI-Pathology	https://doi.org/10.7937/K9/TCIA.2016.TLPMR1AM	[99, 100, 101, 102]	Train
TCGA-PRAD	https://doi.org/10.7937/K9/TCIA.2016.YXOGLM4Y	[103]	Train
Breast-MRI-NACT-Pilot	https://doi.org/10.7937/K9/TCIA.2016.QHSYHJKY	[104]	Train
RIDER Neuro MRI	https://doi.org/10.7937/K9/TCIA.2015.VOSN3HN1	[105]	Train
Soft-tissue-Sarcoma	https://doi.org/10.7937/K9/TCIA.2015.7GO2GSKS	[106]	Train
Mouse-Mammary	https://doi.org/10.7937/K9/TCIA.2015.9P42KSE6	[107]	Train
TCGA-THCA	https://doi.org/10.7937/K9/TCIA.2016.9ZFRVF1B	[11]	Train
TCGA-SARC	https://doi.org/10.7937/K9/TCIA.2016.CX6YLSUX	[108]	Train
LungCT-Diagnosis	https://doi.org/10.7937/K9/TCIA.2015.A6V7JIWX	[109]	Train
TCGA-CESC	https://doi.org/10.7937/K9/TCIA.2016.SQ4M8YP4	[110]	Train
TCGA-OV	https://doi.org/10.7937/K9/TCIA.2016.NDO1MDFQ	[111]	Train
TCGA-COAD	https://doi.org/10.7937/K9/TCIA.2016.HJJHBOXZ	[11]	Train
TCGA-KIRC	https://doi.org/10.7937/K9/TCIA.2016.V6PBVTDR	[8]	Train
TCGA-LGG	https://doi.org/10.7937/K9/TCIA.2016.L4LTD3TK	[112]	Train
QIN PET Phantom	https://doi.org/10.7937/K9/TCIA.2015.ZPUKHCKB	[113]	Train
QIN Breast DCE-MRI	https://doi.org/10.7937/K9/TCIA.2014.A2N1IXOX	[114]	Train
NSCLC-Radiomics-Genomics	https://doi.org/10.7937/K9/TCIA.2015.L4FRET6Z	[66]	Train
Lung Phantom	https://doi.org/10.7937/K9/TCIA.2015.08A1IXOO	[115]	Train
TCGA-KICH	https://doi.org/10.7937/K9/TCIA.2016.YU3RBCZN	[116]	Train
TCGA-GBM	https://doi.org/10.7937/K9/TCIA.2016.RNYFUYE9	[117]	Train
SPIE-AAPM Lung CT Challenge	https://doi.org/10.7937/K9/TCIA.2015.UZLSU3FL	[118]	Train
Prostate-3T	https://doi.org/10.7937/K9/TCIA.2015.QJTV5IL5	[119]	Train

Prostate-Diagnosis	https://doi.org/10.7937/K9/TCIA.2015.FOQEUJVT	[120]	Train
RIDER Phantom PET-CT	https://doi.org/10.7937/K9/TCIA.2015.8WG2KN4W	[121]	Train
RIDER Lung CT	https://doi.org/10.7937/K9/TCIA.2015.U1X8A5NR	[122]	Train
RIDER Phantom MRI	https://doi.org/10.7937/K9/TCIA.2015.MI4QDDHU	[123]	Train
RIDER Breast MRI	https://doi.org/10.7937/K9/TCIA.2015.H1SXNUXL	[124]	Train
CT Colonography (ACRIN 6664)	https://doi.org/10.7937/K9/TCIA.2015.NWTESAY1	[2]	Train
Chexpert	https://stanfordmlgroup.github.io/competitions/chexpert/	[1]	Train
RSNA Bone Age	https://www.kaggle.com/kmader/rsna-bone-age	None	Train
TCGA-BRCA	https://doi.org/10.7937/K9/TCIA.2016.AB2NAZRP	[7]	Validate
Acrin-FMISO-Brain (ACRIN 6684)	https://doi.org/10.7937/K9/TCIA.2018.vohlekok	[3]	Validate
TCGA-BLCA	https://doi.org/10.7937/K9/TCIA.2016.8LNG8XDR	[11]	Validate
Pancreas-CT	https://doi.org/10.7937/K9/TCIA.2016.tNB1kqBU	[125]	Validate
CT Lymph Nodes	https://doi.org/10.7937/K9/TCIA.2015.AQIIDCNM	[126]	Validate
TCGA-ESCA	https://doi.org/10.7937/K9/TCIA.2016.VPTNRGFY	[127]	Validate
TCGA-STAD	https://doi.org/10.7937/K9/TCIA.2016.GDHL9KIM	[128]	Validate
LIDC-IDRI	https://doi.org/10.7937/K9/TCIA.2015.LO9QL9SX	[129]	Validate
QIBA CT-1C	https://doi.org/10.7937/K9/TCIA.2016.YxgR4blU	[130]	Validate
RIDER Lung PET-CT	https://doi.org/10.7937/K9/TCIA.2015.OFIP7TVM	[4]	Validate
Prostate-MRI	https://doi.org/10.7937/K9/TCIA.2016.6046GUDv	[10]	Validate
NIH 100000 Chest X-ray	https://nihcc.app.box.com/v/ChestXray-NIHCC	[131]	Validate
MRNet: Knee MRIs	https://stanfordmlgroup.github.io/competitions/mrnet/	[52]	Validate
CPTAC-CCRCC	https://doi.org/10.7937/K9/TCIA.2018.OBLAMN27	[132]	Test
C4KC-KiTS	https://doi.org/10.7937/TCIA.2019.IX49E8NX	[133]	Test
CPTAC-LSCC	https://doi.org/10.7937/K9/TCIA.2018.6EMUB5L2	[134]	Test
LDCT-and-Projection-data	https://doi.org/10.7937/9npb-2637	[135]	Test
CPTAC-LUAD	https://doi.org/10.7937/K9/TCIA.2018.PAT12TBS	[136]	Test

CPTAC-PDA	https://doi.org/10.7937/K9/TCIA.2018.SC20FO18	[137]	Test
Pelvic-Reference-Data	https://doi.org/10.7937/TCIA.2019.woskq5oo	[138]	Test
Anti-PD-1 Lung	https://doi.org/10.7937/tcia.2019.zjjwb9ip	[139]	Test
ISPY1 (ACRIN 6657)	https://doi.org/10.7937/K9/TCIA.2016.HdHpgJLK	[140]	Test
QIN-HeadNeck	https://doi.org/10.7937/K9/TCIA.2015.K0F5CGLI	[141]	Test
TCGA-READ	https://doi.org/10.7937/K9/TCIA.2016.F7PPNPNU	[142]	Test
NaF Prostate	https://doi.org/10.7937/K9/TCIA.2015.ISOQTHKO	[143]	Test
REMBRANDT	https://doi.org/10.7937/K9/TCIA.2015.588OZUZB	[144]	Test
MURA	https://stanfordmlgroup.github.io/competitions/mura/	[53]	Test
NDA Osteoarthritis Initiative	https://nda.nih.gov/oai/	None	Test
BraTS20	http://braintumorsegmentation.org/	None	Test