Contents

Chapter 1

Literature Review

This chapter is a review of the relevant literature around deep learning and how this type of machine learning has advanced in the past years . In particular, the report will discuss how deep learning can be used in the diagnosis of pathologies in chest X-rays. As well as this we will review successful projects from the past that are closely linked to this thesis.

1.1 Overview

Radiology is a branch of medicine which uses medical imaging technology to generate images so that radiologists can view structures within the human body. The types of images range from X-rays, CT scan(computed tomography scan) and ultrasounds. (Bradley, 2008). For the past few years, radiology departments in the UK have seen disruption in the National Health Service(NHS). A reoccurring issue amongst radiology departments is lack of staff to interpret scans. According to the Royal College of Radiologists, the NHS does not have enough radiologists to meet diagnostic imaging demands leaving patients at risk. (Rimmer, 2017).

1.1.1 Faster Diagnosis

One area which can augment the diagnosis of medical scans is image classification. This process of image classification involves teaching a model to learn features from images and then test on unseen images to compute accuracy. In recent times, image classification through deep learning methods has produced superhuman performance on several image-based classification tasks(Krizhevsky, Sutskever, & Hinton, 2012).

1.2 Traditional Methods of Machine Learning

Before the advancements in deep learning, less sophisticated methods of image classification were employed. This method of machine learning required the crucial step of feature extraction before inputting data into a classifier. In order to visualise the process that needs to take place when using traditional methods of machine learning please refer to figure 1.1.

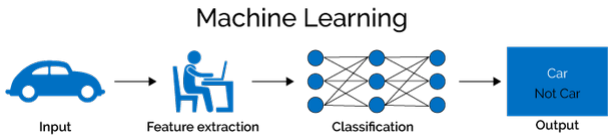


Figure 1.1: Figure showing the process of traditional image classification using feature extraction(Image taken from TowardsDataScience)

Computers see images as pixel values, and each image can be viewed as a 2D matrix of pixel values. When building a machine learning pipeline to classify an image, we need to be able to tell what is unique about each image so that the classifier shown in figure 1.1 can correctly classify. This uniqueness in an image can also be viewed as the features. Classification can then be thought of as detection of features in an image, and if features associated with a class are present in the input, we can predict if the input image is of a specific class (Alexander, 2019)

In order to build a classification pipeline, the model needs to know what features it is looking for in an input image. These features it is looking for are a predefined set of features. In order to apply this rule, we can leverage human knowledge about a picture in a given domain and extract the most important features. We can then detect those manual features and use that detection to make a classification prediction (Popescu & Sasu, 2014)

However, images can have lots of variation. If we want to build a robust pipeline, our model needs to be invariant to image variation, while still being sensitive to the differences in individual classifications. Manual features defined by humans can be used, but where this manual detection breaks down is in the detection task itself, and this is merely based on the fact that one class can have incredible amount of variation. This can be illustrated with the 1.2 provided by (Szegedy et al., 2015)

Figure 1.2: Two distinct class from 1000 classes of the ImageNet database. Domain knowledge is required to distinguish between separate classes

So, how can we do better? We want a way to both extract features and detect their presence in an image automatically. This is where deep learning can be harnessed to extract features, no matter how the image is presented, automatically.

1.3 State-of-the-Art

Convolutional Neural Networks are a special kind of Multi-layered Perceptron(MLP). Like almost every other neural network they are trained with a version of back-propagation algorithm. Where they differ is in the architecture. This contrast can be viewed in figure 1.3 and figure 1.4. CNN's are designed to recognise visual patterns directly from pixel values with minimal preprocessing. They can recognise patterns with extreme variability(such as handwritten characters) and with robustness to distortions and simple geometric transformations. (LeCun et al., 1989) CNN's differ from MLP's in that they take advantage of spatial information of an image by convolving a patch of the input image with a filter weight of the same dimension. This results in a 2D matrix called a feature map. These filter weights are applied over the whole image and are iteratively refined in the training process , so that the correct features can be extracted. Different weight filter can be used to extract different features. This is achieved by passing the input across multiple convolution layers. In order to visualise this please refer to figure 1.4. From the diagrams it can be appreciated that MLP's lose all spatial information about an image by flattening every pixel and connecting these pixel values to a neuron in the hidden layer. As well as this we can see that MLP's have many more parameters in the network compared to CNN which can lead to over-fitting especially if the number of training samples is limited. (Szegedy et al., 2015) Starting with LeNet-5 , CNNS's have typically had a standard stacked structured of convolutional layers, followed by ReLU operation, max-pooling and then a fully connected layer. There are variants of this basic design in different image classification

tasks and they have all yielded respective results on classification tasks such as MNIST and CIFAR. Although, the most notable includes the ImageNet classification challenge where 1.2 million images were classified into 1000 different classes. In the past years CNN architectures have dominated this challenge and the first publication using CNN architecture was produced by Alex Krizhevsky and Geoffery Hinton in 2012(Krizhevsky et al., 2012). The architecture was given the name Alex-Net and achieved a top-5 error rate of 15.3% outperforming the previous state-of-the-art, SIFT (Lowe, 2004) which achieved a 26.2% error rate using traditional methods. Since then, new CNN architectures have been published with improved results and this can be seen in the table below.

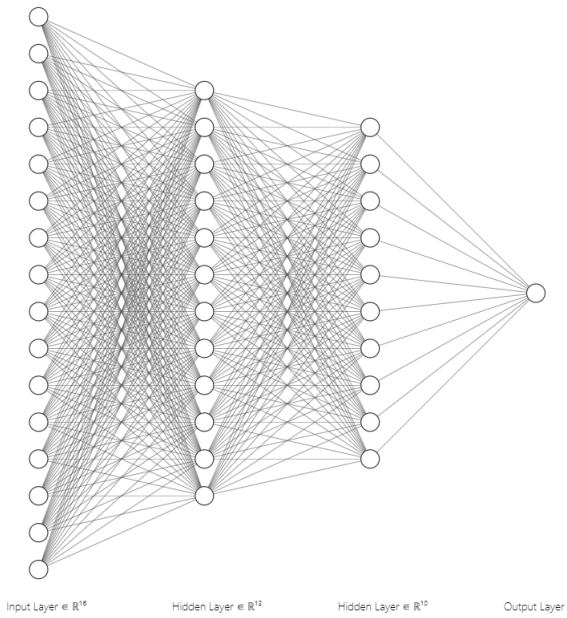


Figure 1.3: Architecture for MLP. Shows how all pixels value in image are flattened losing all spatial information(https://alexlenail.me/NN-SVG/index.html)

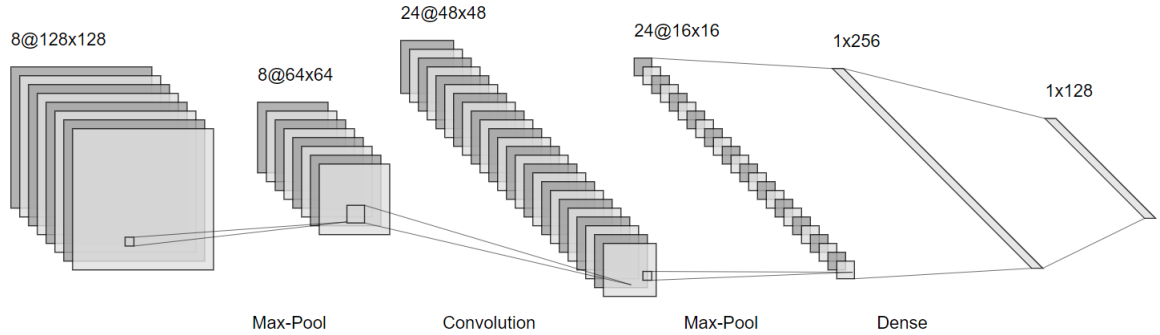


Figure 1.4: Architecture for Le-Clun CNN. Shows how spatial information is preserved as well different operations applied (https://alexlenail.me/NN-SVG/LeNet.html)

Model	Top-5 error rate	Number of Layers
AlexNet(2012)	15.3%	8 layers
VGG16(2014)	7.3%	19 layers
GoogleNet(2014)	6.7%	50 layers
InceptionV3(2015)	3.58%	50 layers
ResNet(2015)	3.57%	152 layers

Table 1.1: Results of different models with Human error rate at 5.1%

From the table we can see that for large datasets such as ImageNet, there is seems to be a trend developing. That is that the number of layers as well as layer size is increasing. A bigger size means that the model holds more parameters and can make it more prone to over-fitting. To address this problem, a technique developed by Google called "dropout" was developed (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012)

1.4 Related Work

Many papers have been published in the domain of using deep learning for medical image analysis. This has been spurred in part due to large carefully curated labelled datasets which have led to papers showing expert-level performance on many medical image classification tasks (Irvin et al., 2019). In this section we will review work related to the thesis title.

1.4.1 CheXNet

In this project, led by Stanford ML Group, an algorithm was developed to interpret chest X-ray images and detect if Pneumonia is present while showing a heat map of the area in the image that had the highest activations (Rajpurkar et al., 2017). Example output can be see in figure 1.5

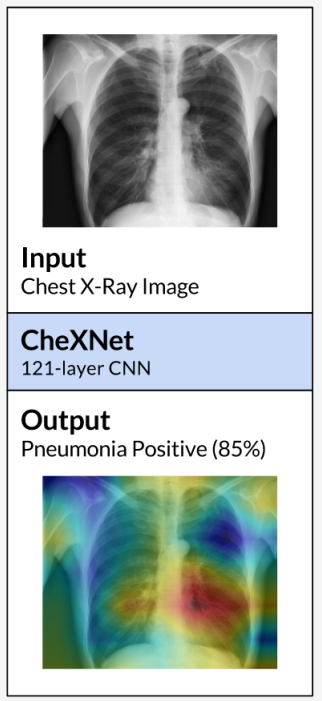


Figure 1.5: Example of CheXNet output of class label along with probability from softmax layer and heat-map

Results showed that the algorithm can detect the pathology at a level exceeding radiologists. The algorithm used a 121-layer CNN which was trained on ChestX-ray14 (Wang et al., 2017) which is currently the largest publicly available chest X-ray dataset, containing over 100,000 frontal view images and 14 different pathologies. The CNN used is based on the DenseNet architecture (Huang, Liu, Van Der Maaten, & Weinberger,

2017) but modifications were made for single output in the fully connected layer by applying a non-linearity sigmoid function, so that only Pneumonia would have a probability. Pre-trained weights were initialised from a model trained on ImageNet (Deng et al., 2009) to speed up the process of training.

1.4.1.1 Results

Precautions were put in place to make sure that patient images between training and testing sets never overlap. In order to test the accuracy of the model, 420 chest X-rays were collected and 4 practising radiologists annotated where they think the Pneumonia is present. Through the CheXNet tests it showed the algorithm exceeds average radiologist performance on the **F1 metric**. The model was comparative with expert radiologists on metrics such as **accuracy**, **sensitivity** and **specificity**. The biggest difference was in terms of speed as radiologists took around 4 hours on average to interpret the images whereas CheXNet algorithm took less than 2 minutes (Rajpurkar et al., 2017). Results from figure 1.6 present the F1 score for algorithm and average of the 4 clinicians.

	F1 Score (95% CI)
Radiologist 1	0.383 (0.309, 0.453)
Radiologist 2	0.356 (0.282, 0.428)
Radiologist 3	0.365 (0.291, 0.435)
Radiologist 4	0.442 (0.390, 0.492)
Radiologist Avg.	0.387 (0.330, 0.442)
CheXNet	0.435 (0.387, 0.481)

Figure 1.6: F1 Score for CheXNet and average F1 score for 4 radiologists

1.4.2 Multi-task Learning for Chest X-ray Abnormality

In this paper, researchers built a model trained on 297,541 chest X-rays. The system is based on a novel multi-task deep learning architecture that in addition to classifying and showing a heat-map of the abnormalities also supports the segmentation of the lungs and heart (Guendel et al., 2019). The research demonstrated that by training the model concurrently on these tasks, one can increase the classification performance. This research produced state of the art performance of 0.883 AUC on average across 12 different abnormalities. ChestX-Ray 14 (Wang et al., 2017) and PLCO datasets were combined to increase the amount of variability of images and this also showed increase in performance.

As mentioned before techniques were applied to increase accuracy. One of the methods

employed was to normalise the images. A challenge in processing chest radiographs is that there can be large variabilities in the appearance of the image and this is due to the acquisition source, radiation dose (Guendel et al., 2019). The paper proposes to dynamically window each image by adjusting the brightness and contrast via a linear transformation of the image intensities. Example of output is shown in figure 1.7

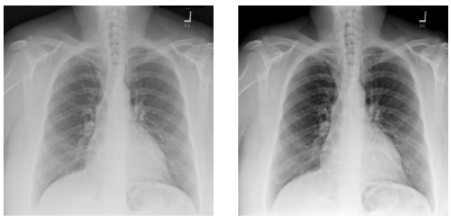


Figure 1.7: We can see that certain parts of the X-ray become more visible hence making abnormalities more clear for the model

In addition , the paper also harnesses spatial knowledge related to individual pathologies as well as underlying structure i.e the heart and lungs which can be exploited to increase classification performance. The paper proposes that once can focus the learning task to the heart and lung region as information outside of these regions may be regarded as irrelevant for the diagnosis of heart/lung abnormalities.

In order to achieve this segmentation, a DenseNet model shown in figure 1.8 (Huang et al., 2017) has been used and figure 1.9 presents the architecture as well as an example showing the segmentation technique. Another technique that was used to take advantage of spatial information was the use of several approximate spatial labels provided in the PLCO dataset. For five abnormalities(Nodule,Mass,Infiltrate,Atelectasis,Hilar Abnormality) there are rough location information available to help the model locate where the abnormality is usually located.

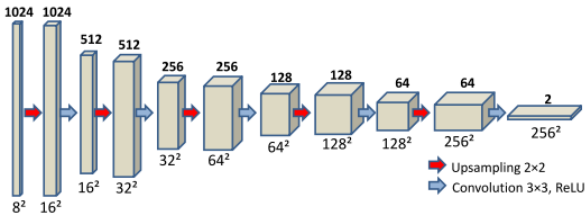


Figure 1.8: Architecture for DenseNet

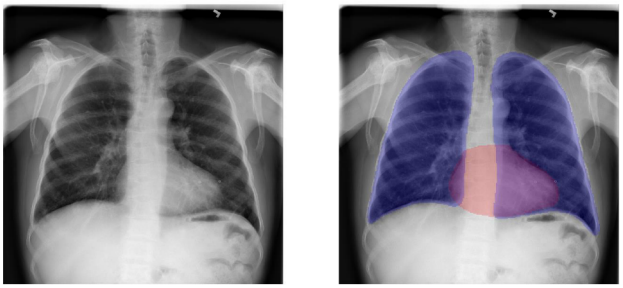


Figure 1.9: Distinct segmentation of lungs and heart

1.4.2.1 Results

For this paper, different experiments were undertaken to show the performance of the model. As a baseline, performance was measured by only training the classification part on the PLCO dataset. The test was evaluated with an average **AUC** score across 12 abnormalities of **0.859**.

Results also showed improved classification scores when the network was additionally trained to generate lung and heart segmentation masks (as shown in figure 1.10). Performance increased to 0.866 on average across the abnormalities.

As mentioned another technique was to use spatial knowledge and the impact of this showed an average improvement of 0.011

This paper also used 2 datasets and by combining both, the average AUC score reaches 0.870. This increase in score is due to more variability in the learning process. As well as this the normalisation technique previously shown in figure 1.7 was applied when training the model and this had a 2 fold benefit. The first was that the training process was reduced on average 2-3 times. The researchers hypothesised that this is due to the images being more aligned in terms of brightness and contrast. Another reason was due to the generalisation of the model parameters and this lead to a performance gain of 0.876.

Finally, the researchers upscaled the input image size to 512 in each dimension and adjusted the DenseNet layer to take in 16x16 patch size at each layer. This final network architecture change also added all of the previous techniques mentioned before. The following image shows the results obtained across all techniques and finally the result of combining all the techniques which produces a state-of-the-art performance of **0.883**

Dim. Size	256 × 256	256 × 256	256 × 256	256 × 256	256 × 256	512 × 512
Data	PLCO	PLCO	PLCO	PLCO+NIH	PLCO+NIH	PLCO+NIH
Features	-	Seg.	Loc.	Loc.	Loc.+Norm.	Loc.+Norm.+Seg
Nodule	0.810	0.815	0.830	0.832	0.831	0.881
Mass	0.829	0.839	0.840	0.867	0.869	0.884
Granuloma	0.884	0.886	0.887	0.887	0.893	0.912
Infiltrate	0.865	0.863	0.864	0.877	0.882	0.891
Scaring	0.841	0.842	0.843	0.850	0.848	0.861
Fibrosis	0.870	0.875	0.863	0.875	0.871	0.884
Bone/Soft Tissue Lesion	0.841	0.846	0.834	0.840	0.850	0.848
Cardiac Abnormality	0.926	0.928	0.922	0.923	0.927	0.926
COPD	0.882	0.883	0.877	0.880	0.882	0.874
Effusion	0.909	0.938	0.927	0.932	0.949	0.939
Atelectasis	0.849	0.858	0.858	0.867	0.855	0.832
Hilar Abnormality	0.796	0.815	0.812	0.815	0.850	0.859
Mean (Location Labels)	0.830	0.838	0.841	0.852	0.857	0.869
Mean	0.859	0.866	0.863	0.870	0.876	**0.883**

Figure 1.10: Results showing increasing mean AUC score

1.4.3 Abnormality Detection and Localization in Chest X-Rays using Deep Convolutional Neural Networks

For this paper, researchers conducted experiments on three datasets. From the datasets, researchers explored the performance of various deep CNN's for detection of heart disease from chest X-rays. Researchers used binary classification on diseases such as Cardiomegaly against normal images (Islam, Aowal, Minhaz, & Ashraf, 2017). The paper explores several CNN models such as AlexNet, VGG-Net and ResNet. These models have different layers and typically achieve higher accuracy as layers increase as noted in State-of-the-Art section.

Another part of the research, similar to previous papers reviewed was to produce a heat-map showing which areas of the X-ray the model has the highest activation on. This paper used the sensitivity of softmax scores of occlusion on a certain region in the chest X-ray to find which region in the image is responsible for the classification decision.

1.4.3.1 Results

The first experiment used single model CNN's on the Indiana dataset, and from the tests, it was shown that deeper models like VGG-19, ResNet improve the accuracy significantly. It was found that Cardiomegaly detection improves 6% compared to using shallower models like AlexNet. Although results showed high specificity for ResNet-101 and high sensitivity for VGG-16, the VGG-19 gave the highest AUC score of 0.94. This at least 1% higher AUC for other models. As well as this, dropout was used, which increased performance on the shallower networks but degraded deeper ones. For all experiments, it was noted that extracting features from earlier layers lead to improved accuracy of 2-4%. It was concluded that shallower networks were better for detecting smaller objects, and as an example, it was found that shallower layers from ResNet-15 trained on Cardiomegaly had better performance than later layers across different

metrics. figure 1.11 shows a more in depth picture of findings.

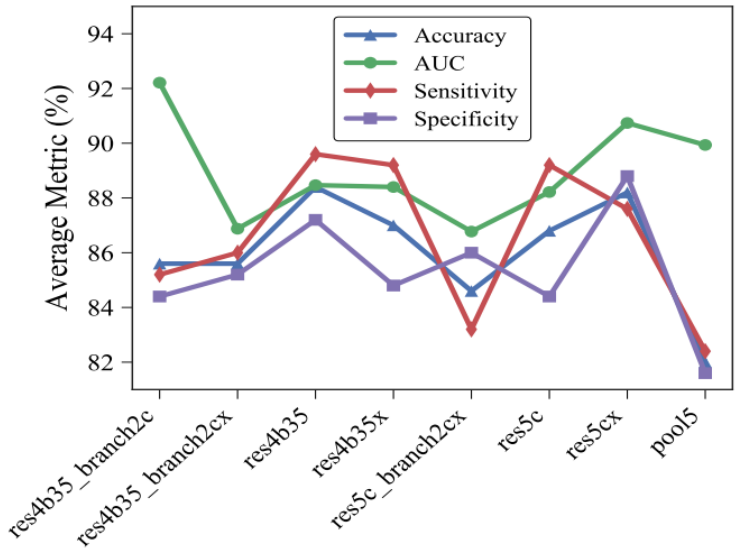


Figure 1.11: Researchers showed that shallower layers in models had a higher accuracy on Cardiomegaly detection compared to deep layers

Another fascinating result was on the localisation of Cardiomegaly shown in the image below. It can be observed that from the figure 1.13, the model has higher activations from the heart region, which is expected as Cardiomegaly concerns abnormal enlargement of the heart. What was revealing was that other than an enlarged heart for a patient with Cardiomegaly, there is not much difference in features between a normal image and an image classified as Cardiomegaly. But the model can still differentiate between a normal heart which is enlarged due to age or physical attributes of the patient.

Figure 1.12: Occlusion sensitivity used to create heap map of region where it thinks Cardiomegaly exists

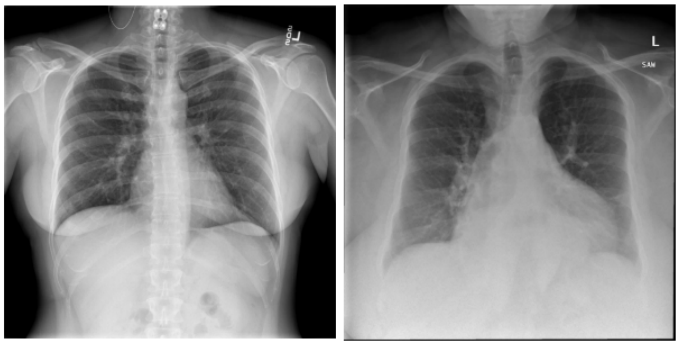


Figure 1.13: Example of normal heart on the left and Cardiomegaly diagnosed heart on the right

Finally, another point raised in the paper was the model used for localisation of cardiomegaly were counter-intuitive to traditional methods of cardiomegaly diagnosis, which consider the relative size of heart and lung. But, most of the signals contributing to the softmax score come from the heart alone indicating features in the shape of the heart and its surrounding region alone help in detection of the pathology and perhaps the lung and relative size to the heart are less critical in the diagnosis for the model.

1.5 Analysis of Related Work

In light of reviewing the related work, we can see a wide array of methods have been applied to produce several State-of-the-Art results on pathologies such as Pneumonia, Lung Nodules and Cardiomegaly. The paper from Stanford ML Group was one of the first to be tested on such a large scale dataset and made a mark in the application of deep learning on X-rays. Results from this paper show a 121-layered network produce results comparable to expert radiologists by using an architecture similar to that of DenseNet with some tweaks. As stated the ChexNet model has a higher F1 score than the average F1 score for four practising radiologists. After the success of ChexNet, more papers were released in the same domain trying to improve on accuracy metrics. The article discussed in section 1.4.2 uses more advanced features to locate the pathology more accurately. Given these additional features, it allows for the model to disregard certain regions of the images hence reducing noise in training process. From the results presented. It shows a clear improvement in accuracy across different pathologies once these features are introduced to the model. Lastly, the paper from X demonstrates a clear difference of performance on know CNN architectures. In section X, it showed that deeper layers led to better performance on the ImageNet classification task. However contrary to most image classification tasks, paper shown in section 1.4.3 shows how even though we see a better accuracy on deeper models, we see a higher AUC score for models like VGG-19 compared to ResNet-101, which has more layers, and this was due to earlier layers being better at detecting smaller objects. We also see the use of an ensemble of models to increase accuracy, which was something that was not experimented with in the previous papers reviewed. As well as this, we see a model trained on Cardiomegaly questioning the current ways of detecting this pathology such that it disregards traditional methods of looking at the relative size of the lung and heart in comparison. Lastly, from the three papers reviewed, we have seen the constant evolution which has allowed for more sophisticated methods of pathology detection and perhaps a combination of techniques in the future can increase detection amongst heart and lung pathologies.

1.6 Medical Images

As mentioned in the Overview section, radiology departments in the UK have struggled to meet imaging diagnostic demands. The survey published by the Royal College of Radiology reported 97% of radiology departments were unable to do so within their working hours. As mentioned in the paper, "It points to an insufficient number of radiologists to meet the increasing demand for imaging and diagnostic services". According to the report it cited that radiology has the second lowest proportion of trainees to consultants: 26 trainees to every 74 consultants. This is compared to an average in all specialities of 40 trainees for every 60 consultants. The report also cites that there was a particular workforce shortage prominent in Scotland, where the consultant workforce grew by 7% from 2010 to 2016 but demand for CT,MRI scans increased by 10%. One of the profound impacts of such shortage is the risk to patient care as well as economical effects. The NHS paid nearly £88m in 2016 for backlogs of radiology examinations to cover backlogs of radiology examinations to be reported. To cover these backlogs 92% of radiology departments paid staff overtime, 78% outsourced to private companies and 52% employed ad hoc locums. This amount could have paid for at least 1028 full time radiology consultants.(Rimmer, 2017)

Given this what has been done to augment the analysis of medical images in the NHS and reduce extreme workload and large expenses ? One particular software which has bee used in the clinical setting is Computer Aided Diagnosis(CAD). CAD is technology which is designed to decrease observational oversights and false negative rates for physicians interpreting medical images such as mammographies (Castellino, 2005). According Dr. Paul Chang, the reason this type of software did not receive as much attention, compared to the possibility of using deep learning, was that it used traditional methods of machine learning. (*Video from RSNA 2018: Dr. Paul Chang on AI and radiology*, n.d.)

1.6.1 Challenges

There many challenges associated when applying new technologies in a clinical setting. Medical images are very large and complicated and for most of the image classification tasks in the ImageNet example , distinct features or pixels can be learned by a model. Dr. Matthew Lungren , radiologist at Stanford Medical Center, mentioned how most of the pixels in a chest X-Ray from a healthy person who happens to have lunge cancer , will tell you that the image is of a healthy person. But there is only a small proportion of the image which indicates the finding of lung cancer fo the classifier. Dr Lungren also noticed how medical images may have, what is known as artefacts or tokens in the image. Doctors are able to deal with context when looking at an image , but

such artefacts or tokens may lead to a model making a decision with higher degree of confidence but in actual fact the classification is incorrect (DrMatthew, 2018)

1.7 Summary of Review

From the review, it is quite evident that with the ever-growing concern in the NHS of a shortage of radiologists (Rimmer, 2017), solutions need to be found in order to solve the issue of increased backlogs. Image classification, as mentioned, can provide augmentation of diagnosis of X-rays. Great leaps and bounds have been made in the field of deep learning and particularly well know CNN architectures show performance exceeding that of human intelligence in the case of ImageNet challenge (Krizhevsky et al., 2012)s. As well as this more sophisticated example of applying deep CNN's to medical images has shown outstanding results on pathologies such as Pneumonia and Cardiomegaly (Rajpurkar et al., 2017). This problem of diagnosing X-rays also comes with its challenges which have been outlined in the challenges section. Other hurdles must be kept in mind for implementation purposes, and these include making sure that pixel values are not lost. Given the nature of X-rays, pixel loss can lead to pathologies being removed from an image such as lung nodules and according to the Royal College of Radiologists certain rules needs to be adhered to so that images do not lose quality when it comes to diagnosing them. Lastly, when training a model on medical images, the dataset chosen needs to be balanced in class labels. As well as this one of the big obstacles is lack of datasets of reliable labelling of datasets as noise in class labels can lead to reduced performance.

1.8 Moving Forward

In this section, we will conclude how the rest of this paper will be implemented and how the proposed problem in the introduction will be approached. From the findings above the aim is to apply one of the well known CNN architectures to build an accurate image classification pipeline on predicting the presence of Pneumonia in chest X-rays. In order to train the model, a carefully curated dataset must be chosen. The dataset that will be used in this thesis comes from a public dataset carefully curated by Stanford ML Group called CheXPert. This dataset was created to solve three problems which include enough instances, strong reference standards and provide human performance metrics for comparison. It consists of 224,316 chest X-rays, and one of the main obstacles in the development of these datasets is the lack of strong radiologist-annotated ground truth. The objectives is to build an accurate model which can predict and localise the Pneumonia with techniques such as occlusion. Another objective would be to build an interface which allows the entry of a chest X-ray and a predicted class label will be

given back as well as a heat-map indicating which parts of the image had the highest activation. Lastly, if results from the initial pipeline show high accuracy, specificity, sensitivity and AUC, then the model could be extended to classify more than one pathology contained in the dataset.

Chapter 2

Requirements Analysis

In order to answer the question proposed in the literature review, several objectives must be met. As this project cover two stages: Application of deep learning model on medical images and visualisation, this requirements analysis is split up into two sections: first section relating to requirement analysis of the implementation of deep learning model and the second being visualisation.

These requirements have been gathered based on the MoSCoW rule, which prioritises different requirements by stating the project must, should or could achieve the detailed requirements by the end of the implementation stage. A requirement that the project "must" achieve has the highest priority, where if this requirement is not met, the developed product will not be fit for purpose. The ones that "should" be completed are next in the priority hierarchy, whereby the project will still be functional without them, but their addition to the project would improve the end product. The requirements that "could" be completed are those that are not necessary to the overall functionality. However, these would benefit the project.

2.1 Deep Learning Model

2.1.1 Functional Requirements

2.1.1.1 Dataset

Must pre-process images from the dataset. Since input into the neural network, used in the implementation phase, requires pixel values, every image will need to be converted to its respected pixel value. Moreover, any resizing of the images will also need to be done as input to the neural net needs to be of a specific shape. Hence, this needs to be investigated in the implementation stage.

2.1.1.2 Convolutional Neural Network

Must build a model using image classification algorithms like CNN.

2.1.1.3 Improve Performance of model

Must look into ways of tweaking model parameters, and evaluating the results of these improvements, as per the ones described in the literature review.

2.1.1.4 Comparison of different algorithms

A report showing a comparison of different algorithm employed to build a model and predict could be created depending on the time.

2.1.1.5 Extending classification

The model's architecture could be further extended to diagnose more than one pathology.

2.1.2 Non-Functional Requirements

2.1.2.1 Programming Language and Libraries

Must use Python for rapid development and prototyping of CNN model via libraries such as Keras and sci-kit learn for visualisation and explanation of results.

2.1.2.2 Proof of Concept

Must be fully functional by the proof of concept stage

2.1.2.3 Anonymised Date

Must use anonymised patient data from publicly available datasets in order to conform to GDPR rule.

2.1.2.4 Accuracy

Model should diagnose images with an acceptable accuracy rate

2.2 Web App for visualisation

2.2.1 Functional Requirements

2.2.1.1 Visualisation

If the model evaluation shows promising performance, visualisation and explainability of results must be used in the form of a web application:

- Must show a classification probability to the user when a chest X-ray is tested on the deployed model

- Should show a localisation of where the model believes the Pneumonia is present in the X-ray.

2.2.1.2 User Input

User must be allowed to upload an image through the interface and get back results from the image being tested on the model.

2.2.2 Non-Functional Requirements

2.2.2.1 Programming Language and Libraries

Must use Javascript,HTML,CSS and cloud services for web application development:

- Should use Amazon Web Services(AWS) Sage Machine and AWS Lambda to deploy trained and tested model. The model should be accessed via the deployed RESTful API

2.2.2.2 Completion Date

Must be completed by the project demonstration day or could be operational by proof of concept deadline

2.2.2.3 Compatibility

Should be compatible with all major web browsers

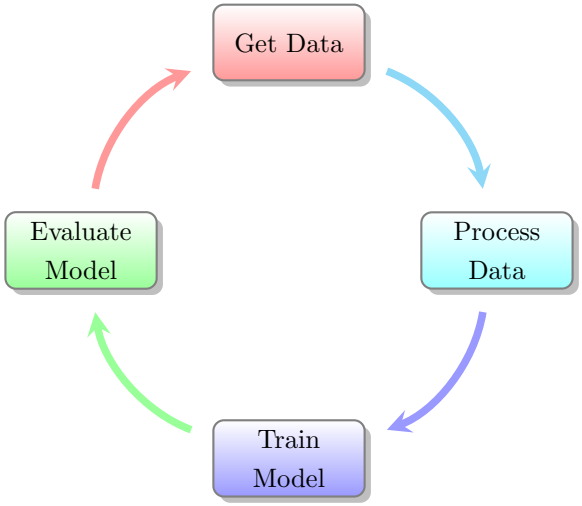
2.2.2.4 Image quality

Must not reduce image quality when passing image to cloud service where the model has been deployed upon

2.3 Methodology

The requirements in the previous section will be delivered in an agile manner. Using an agile approach consists of implementing the model, testing it against the test set, evaluating and reporting on the results of the performance. This process will be iterative until satisfactory results from the model are achieved, then leading to the deployment of the model onto a cloud service. The image shown in figure 2.1 demonstrates the steps that will be taken to build a deep learning pipeline that allows us to apply algorithms on the dataset and analyse performance.

Figure 2.1: Deep Learning Pipeline



Chapter 3

Experiment Setup

Before moving onto explaining process of applying the machine learning models for this paper, the following chapter explains the experimental setup that took place before starting the experiments. In this chapter the hardware and software setup is discussed. As well as this, a general overview of the dataset that is used for the experiments is discussed and more importantly the preprocessing that had to occur before running experiments is explained. Lastly, evaluation metrics will be discussed so that experiments in the implementation chapters can be analysed against metrics that determine the success or failure of an experiment.

3.1 Hardware Implementation

A computer system with the specifications shown in Table 3.1 will be used to carry out the experiments in this paper. Through the help of Robert Gordon University's Research Hub for Artificial Intelligence, this project makes use of the Research Hub's Nvidia DGX-1 to accelerate the deep learning experiments carried out in this paper.

Table 3.1: Computer System Specification

Operating System	Ubuntu 18.04 Linux Host OS
CPU	2x Intel Xeon E5-2698 v3 (16 core, Haswell-EP)
System Memory	512GB DDR4-2133 (LRDIMM)
GPU	8x NVIDIA Tesla P100 (3584 CUDA Cores)
GPU Memory	128GB HBM2 (8x 16GB)

3.2 Software Implementation

The setup for the experiments in this paper involves the use of **Python 3**. The solution heavily relies on **Keras**, which is an open-source neural network library written in python and is capable of running on top of **TensorFlow** as its back-end. Keras is designed to enable fast experimentation with deep neural networks and contains the basic building blocks to build one such as adding layers, objective functions, activation functions and optimisers. Moreover, Keras through the use of TensorFlow allows for the use of distributed training of deep learning models on multiple Graphics Processing Units - GPUs. This is further achieved by using CUDA, which is a parallel computing platform and Application Programming Interface(API) created by Nvidia. The CUDA platform is a software layer that gives direct access to the GPU allowing for highly parallel manipulation of large blocks of data.

3.3 Dataset

The dataset that will be used for training and testing of the learning algorithm has been taken from a public repository (?, ?). The dataset includes X-ray images of both normal patients and patients with pneumonia infected lungs. Based on the information given by the authors that published the dataset, for use in the paper titled "Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning" (Kermany et al., 2018), the data has been obtained from the Chest X-ray images selected from a respective cohort of pediatric patients aged from one to five years old from Guangzhou Women and Children's Medical Center.

The dataset comes with two folders which include training and validation. Each folder contains subfolders for each type of classification, normal and pneumonia. The training folder contains 5218 images split into 1342 normal images and 3876 pneumonia images. The validation folder contains 624 images split into 234 normal images and 390 pneumonia images. Figures 3.1 and 3.2 help to visualise the distribution of the training and testing set respectively. The distribution of both training and testing are important to notice as this will be an important consideration when evaluating the model performance.

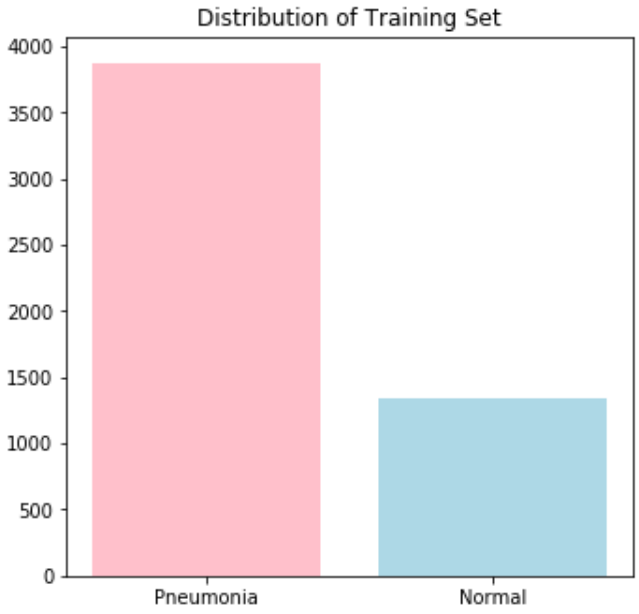


Figure 3.1: Class Distribution of Training Set

Figure 3.2: Class Distribution of Testing Set

To visualise the images used to train the learning algorithm, figure 3.3 and 3.4 shows an example of both a normal patient and a patient suffering from pneumonia. It might be noticeable in figure 3.3 that a healthy X-ray would normally be clearer, and the general area of the chest is more visible compared to cases where pneumonia is present, as

shown in figure 3.4. However, the onset of pneumonia in patients can differ depending on the severity of the infection. As such, there may only be a few pixels that denote the presence of pneumonia which makes the task of detecting the disease a challenge.

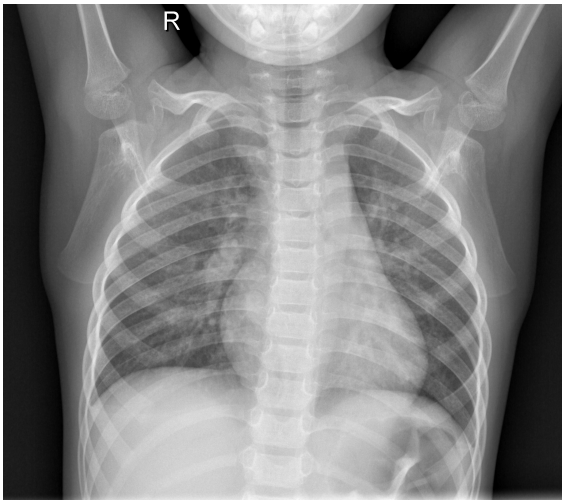


Figure 3.3: Normal Chest X-ray

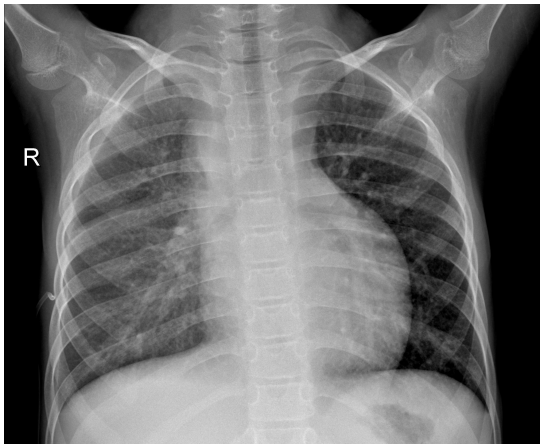


Figure 3.4: Pneumonia Chest X-ray

3.3.1 Dataset Preprocessing

During exploration of the dataset it was shown that the X-ray images provided were of different sizes. This most likely due to different X-ray imaging machines used during collection of the dataset. Before applying the learning algorithm on the X-ray images we needed to make sure that each input image was of the same dimension. This is due to the way a CNN is configured via libraries like Keras, where the size of the input layer is constant. All images were resized from their original size to 250 x 250 which meant that some image were scaled up and other were scaled down.

3.3.2 Normalising Images

Data normalisation is an important step which ensures each pixel value for an image has a similar data distribution. This makes convergence faster while training a neural network. Although non-normalised data can be presented to the neural network, this can result in challenges during training, such as slower training of the model.

As discussed in section 1.3 "State of the Art" , neural networks process inputs using small weight values which are multiplied by the input value. If these inputs are on a large unsigned integer scale, like pixel values, this may disrupt or slow down convergence of the training process. A pixel value of 0(white) compared to 255(black) are both equally as important but the 255 value will have a greater effect on the final prediction. This large value could cause the output to be wrong and lead to the training process taking longer.

For the purposes of this paper, data normalisation is applied by dividing each pixel value by 255 and as a result all inputs to the model have a range between 0 and 1.

To better understand the stages of normalisation, figure 3.5 shows the steps taken to normalise each image. [1]

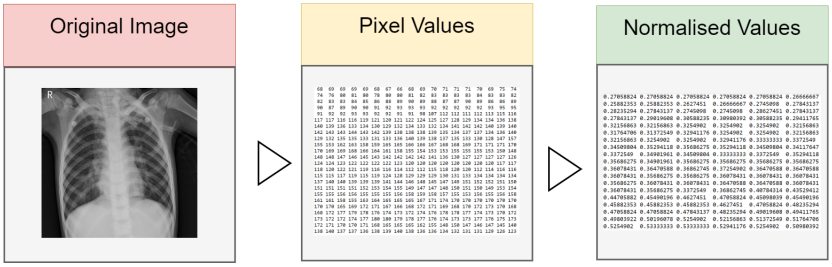


Figure 3.5: Stages of the Preprocessing and Normalisation

3.4 Going Beyond Accuracy

As shown in figure 3.2, the class distribution of the testing set is imbalanced. Due to this imbalance, it is important to look beyond accuracy to validate a model performance. For example, since the test set is imbalanced, if all the test cases were labelled positive, i.e. a class label of pneumonia, then the accuracy on the test set would be 63%. However, the accuracy does not show the underlying performance of the model, which indicates that it is classifying every patient as having the infection.

Due to the domain of this paper and its aim: Automatic classification of X-ray Images

[1] Python Notebook:Preprocessing

as either normal or abnormal, a confusion matrix needs to be considered when evaluating results. Figure 3.6 shows an example of a confusion matrix that will be used to analyse results. The number of pathological samples that are *correctly* identified as a pathological sample by the model is called a true positive (TP). The number of pathological samples that are *incorrectly* classified as normal by the model is called a false negative(FN). The number of normal samples that are *correctly* identified as normal is called a true negative(TN) and in a similar fashion, the number of normal samples that are *incorrectly* identified as pathological samples is called false positive(FP). Given the domain of this study, the cost of a false negative is higher than a false positive

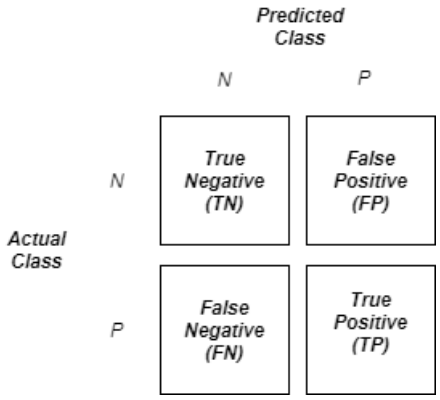


Figure 3.6: Confusion Matrix

Although accuracy is still considered, this study will also examine the values of precision, recall/sensitivity and specificity. Accuracy, specificity, recall and precision are computed as follows:

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

$$Specificity = \frac{TN}{TN + FP}$$

$$Recall/Sensitivity = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

- **Accuracy** is a basic measure shown as a percentage of the total correct classification out of all the samples in the testing set.

- **Specificity** shows the degree to which the model correctly identifies normal samples as normal.
- **Recall/Sensitivity** is a measure which shows the degree to which the model does not miss a pathological sample
- **Precision** is measure of the exactness or quality of the model when it returns results for a specific class

Chapter 4

Experiments & Results

In this chapter, the experiments that were performed during the duration of this project will be discussed. This discussion will include the successful experiments as well as the experiments that failed due to a flaw in the initial setup. For each experiment the general setup will be discussed. This includes the CNN architecture, epochs, batch size, learning rate and training and testing splits. Each of the experiment results will be discussed in detail where the evaluation metrics from 3.4 will be applied to the model results.

4.1 Experiment 1

Model Architecture: For the first experiment that was carried out a simple CNN architecture was used to classify the images. The setup of the CNN involved the use of two convolutional layers each with ReLU activation. After the final convolutional layer, Max Pooling is applied to reduce the dimensionality of the network. After applying Max Pooling the features extracted are flattened and sent through a fully connected layer consisting of 128 neurons. ReLU activation is applied once more before adding a further 2 neurons and finally a Soft-max layer. Figure 4.1 shows the layers that have been used in Experiment 1

Figure 4.1: Model Architecture for Experiment 1

Hyper-parameters

Hyper-parameter setup for the experiment is the following:

- Imbalanced training set(*Non-Shuffled*)
- Balanced testing set
- Batch Size = 64
- Epochs = 1
- Adam optimiser (lr=0.01)
- Categorical Cross Entropy Loss Function

Results

For the initial experiment, the following confusion matrix shows the models performance:

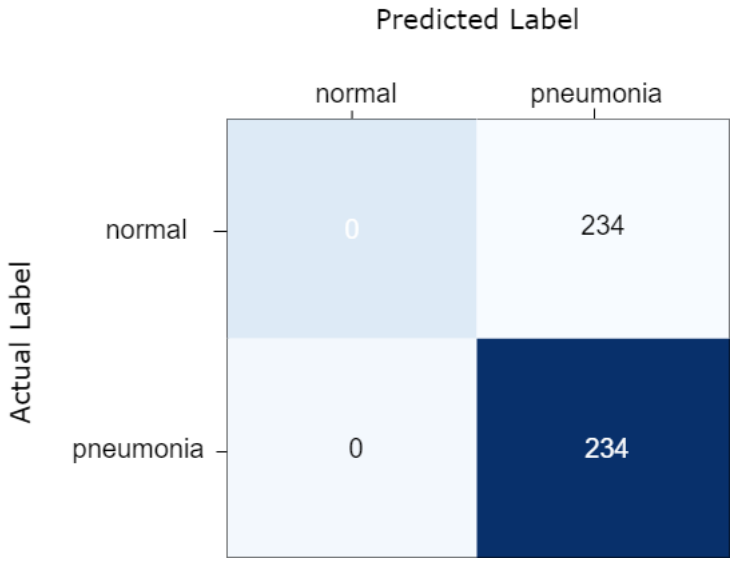


Figure 4.2: Experiment 1 Confusion Matrix

After inspecting the results above, it can be observed that the model trained on the above hyper-parameters has an accuracy of 50% on the test set shown in the confusion matrix above. From the matrix, it shows the model has a Recall/Specificity score of 1 since the model was able to identify all patients with pneumonia. However, although the model can classify pneumonia cases correctly, the opposite is true for the normal cases as the model classifies them as positive cases, hence giving a Specificity score of 0.

Observations were made to understand why the model produced the results in figure 4.2. Upon reviewing the setup for experiment 1, it was noted that since the training data was not shuffled it meant that for each epoch the model was learning 74% of the pneumonia cases first and a further 26% of the normal cases. It meant that the model was learning in a sequential manner which affects the weight updates. It is important that moving forward the training set is shuffled so that batches of input contain a mixture of both positive and negative samples.

Another observation noted when reviewing results is that the number of epochs was too low. The number of epochs determines the number of passes over the entire training set. It means the number of epochs must be higher so that the model has more chances to learn from the training set.

Lastly, in this experiment, it was decided that the testing should be balanced. However, after reflection experiments going forward should not balance out the testing set as this could remove vital images which could include difficult samples and may test the model's capability further. Balancing of the provided data would not reflect the medical environment where it is common to have imbalanced datasets.

4.1.1 Improving Experiment 1

Model Architecture In this experiment, the aim is to see the effect higher epochs and shuffled training data has on the model performance. The model architecture is the same as the previous experiment which is illustrated in figure 4.1.

Hyper-parameters

- Imbalanced training set(*Shuffled per epoch*)
- Validation set(*10% of training*)
- Imbalanced testing set
- Batch Size = 64
- Epochs = 10
- Adam optimiser (lr=0.01)
- Categorical Cross Entropy Loss Function

The main differences in hyper-parameters settings include increased epoch to allow the model to have more passes to learn from the training data. The training set has also been shuffled and the testing set has been changed to the unbalanced distribution mentioned in the previous experiment. To track the model performance over the entire training process this experiment used 10% of the training set to act as a validation set. By doing so we can track the model accuracy and loss over 10 epochs.

Note: Keras allows us to shuffle the data via the *shuffle* parameter which shuffles the order of the data at every epoch. This allows the model to learn in a varied manner since the sequence of the training data is different per epoch. By using this feature in Keras it means reproducibility is not guaranteed but experiments ran multiple times should show similar results.

Results

For this experiment, the following confusion matrix shows the model performance after changing hyper-parameters. Since the model has been run for more than one epoch, an accuracy and loss graph can be used to track the model performance on the training and validation set through out the training process.

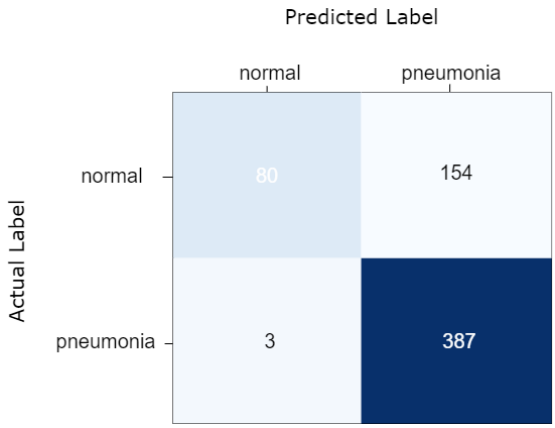


Figure 4.3: Experiment 1.1 Confusion Matrix

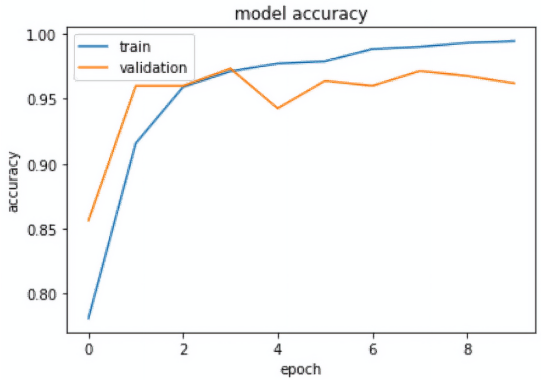


Figure 4.4: Model Accuracy

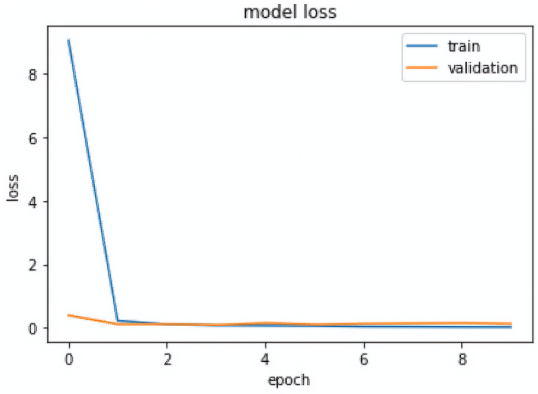


Figure 4.5: Model Loss

Results Analysis

The confusion matrix in figure 4.3 shows that after shuffling the training data and increasing the number of epochs the model achieves a 75% **Accuracy** on the test set. **Recall/Sensitivity** score is 0.99 and the **Specificity** score is 0.3. From figure 4.4 we can track the model accuracy on both the training and testing set. The graph shows us the model is able to quickly learn the training set close to 100% accuracy between the first and third epoch(Python epoch begins from 0th epoch) and continues to increase which suggests the model could have been trained for longer. Figure 4.5 shows a similar story but measures the models loss which shows the training loss drops quickly in the first 2 epochs. The validation line in figure 4.4 shows the accuracy increases a lot between the first and second epoch, however, later on in the training process validation accuracy tends to fluctuate before it starts to drop.

Although the model has a reduced Recall/Specificity score, which is expected since the

model has learned in a varied manner, the number of false positives has gone down which suggests increased epochs and a shuffled training set improves model performance. However, it still evident that due to the imbalance in the training set the model is classifying almost double the amount of false positives compared to true negatives. To diagnose the issue of why the validation data is suffering from a case of over-fitting the same experiment was ran for four additional runs where the test accuracy ranged from 71% to 75%.

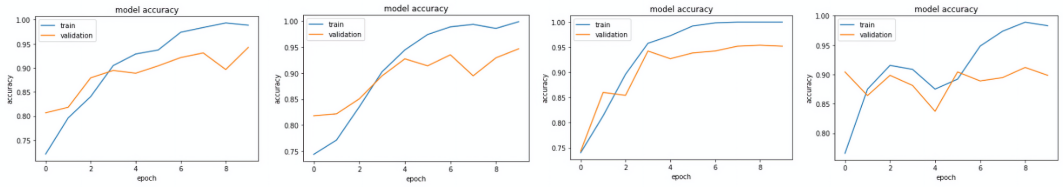


Figure 4.6: Identical experiments with different outcomes

Figure 4.6 shows that the same experiment ran multiple times results in a different training process, where even though training accuracy for all five experiments ends in a similar trajectory, the validation accuracy shows a different story as all five validation accuracies are different. On reflection the reason the same model produced different accuracies is due to the value of the learning rate being too high for this dataset.

The learning rate is a hyper-parameter that controls how much to change the model in response to the estimated error each time the model weights are updated. In this specific situation we hypothesis that since the data is shuffled each of the experiments sees a different sequence of data which when coupled with a high learning rate causes drastic changes in weight updates which effects the training and validation accuracy.

4.2 Hyper-Parameter Tuning

In this experiment(s) the aim is to apply hyper-parameter tuning to improve performance of the previous experiment and stabilise the training and validation accuracy and loss. There are a range of hyper-parameters to experiment with however, the experiments carried out in this section will focus mainly on adjusting Adam learning rate, increasing epochs, changing the number of convolution operations in the convolutional layers and number of dense connections in the fully connected layers.

4.2.1 Learning rate

It was noted in the experiments carried out in section 4.1.1 that after running the same experiment five times the overall testing accuracy varied as well as the validation

accuracy and loss. The aim of this experiment is to reduce the learning rate to see the effect it has on stabilising results.

Model architecture

The model architecture is the same as before and is illustrated in figure 4.1

Hyper-Parameters

The model hyper-parameters are largely the same with learning rate changing from 0.01 to 0.0001.

Results

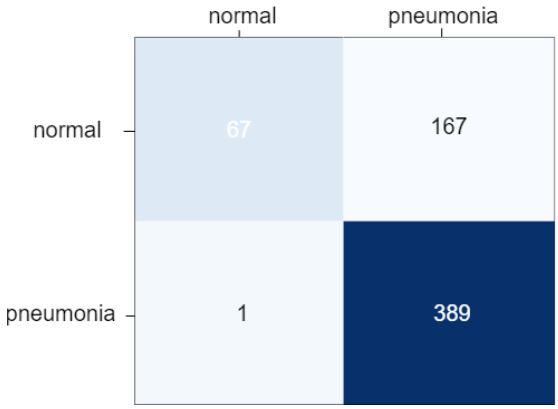


Figure 4.7: Confusion Matrix after adjusting learning rate

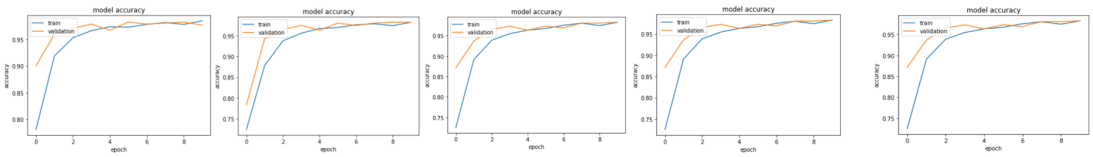


Figure 4.8: Model accuracy after changing learning rate

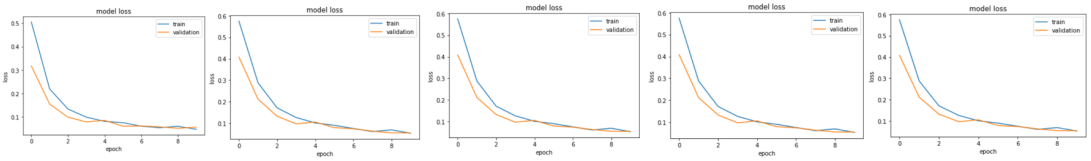


Figure 4.9: Model loss after changing learning rate

In this particular experiment, although we see a decrease in performance, the same model ran five times produces results clustered closer together with testing accuracy across five models being the same except one which differs by one percent. In figure

4.8 and 4.9 we see that for the five models ran, all of them show a similar trajectory with both training and validation accuracy closely follow each other. A change in learning rate also appears to reduce the discrepancy between the training and validation accuracy acting as a regularisation effect which stopped over-fitting. The same can be said for the loss graph show in figure 4.9 which shows the training loss is still lowering with the validation loss starting to flatten out. This suggests that the model could have been trained for longer.

Comparing figure 4.6 and 4.8, we can see that the model after tuning the learning rate is stable across multiple runs. As a result of reducing the learning rate, although it stabilised results, the model has smaller weight updates which slows down the process of convergence.

Therefore, we ran the same experiment with 20 epochs which resulted in test results shown in the confusion matrix in figure 4.10 and the model accuracy loss shown in figure 4.11 and 4.12 respectively.

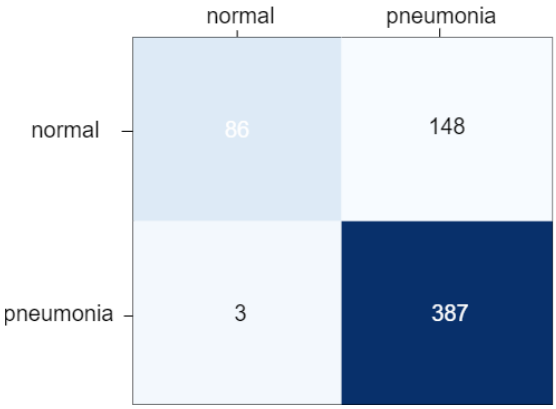


Figure 4.10: Results after 20 epochs

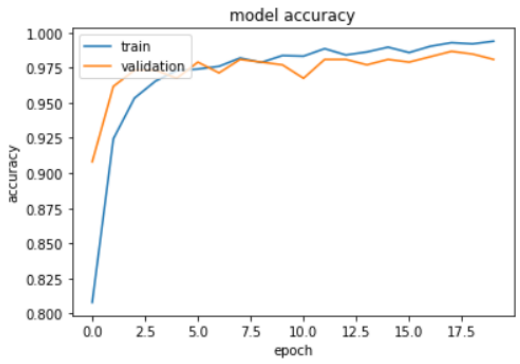


Figure 4.11: Model Accuracy

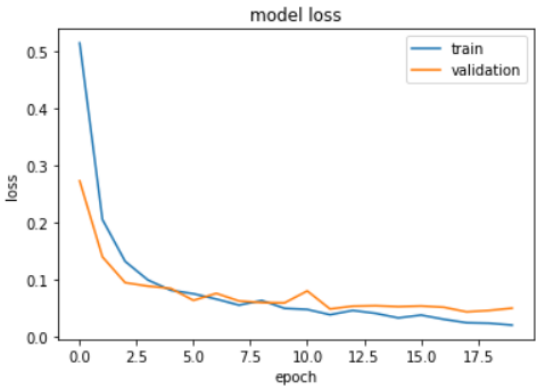


Figure 4.12: Model Loss

After running the experiment over 20 epochs, the overall testing **Accuracy** is 76% with a **Recall** and **Specificity** score of 0.99 and 0.37, respectively. From the model accuracy graph shown in figure 4.11, the training process is generally stable when comparing both the training and validation lines, which suggests that a reduced learning rate and increased epoch value has kept stability during training. There are slight deviations after the 9th epoch until the 20th where the training and validation accuracy and loss start to deviate from each other. When comparing the result from the experiment in section 4.1.1, the results in this experiment after reducing the learning rate and increasing epochs shows a slight increase in **Accuracy** and **Specificity**.

4.2.2 Adding Convolution and Fully Connected Layers

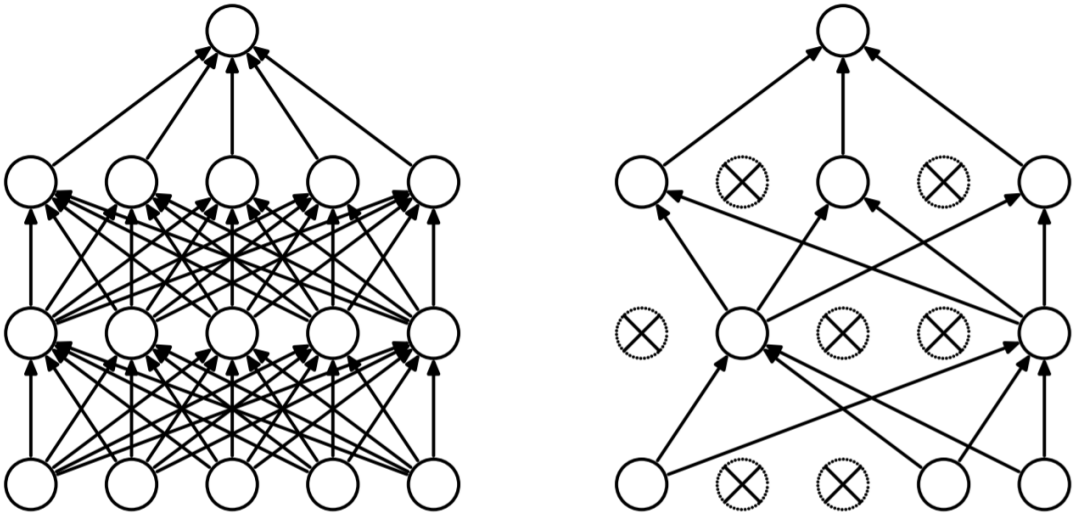
Previous experiments showed that we were able to stabilise results over many runs and increase specificity. Another approach that we took to further reduce the higher number of false positives, is to alter the convolutional and fully connected layer in the model architecture. Figure 4.13 and 4.14 shows a comparison of the model architecture between the previous experiments and the one used in this experiment.

Figure 4.13: Model Architecture for previous experiments

Figure 4.14: New Model architecture

What is noticeable is the addition of a convolution layer and the number filters in each of those layers. These layers are responsible for extracting the relevant features of an image, before passing them through a fully connected(FC) layer, where the classification process occurs. Previous experiments maintained the same number of filters in the two convolutional layers, whereas in this experiment we doubled the amount of filters after every layer. The intuition behind this choice is that CNN's typically detect lower level features via the filters from the first convolutional layer, such as edges and lines. Once the lower level features are extracted, the second layer typically detects motifs by spotting particular arrangements of edges, regardless of small variations. Finally, the third layer may assemble motifs into larger combinations that correspond to parts of familiar objects (LeCun, Bengio, & Hinton, 2015). By increasing the number of filters, a higher number of inputs are passed to the FC layer, and as such, it makes sense to increase the number of neurons in the FC layer which is why an additional layer of 64 neurons is added.

Moreover, the addition of Max Pooling helps with reducing the dimensionality of the network after every convolutional operation and lastly, a technique know as dropout was also used in this experiment to help combat over-fitting. Previous experiments showed signs of over-fitting towards the end of the training process and so increasing the number of convolutional layers could increase over-fitting and hence drop performance on the test set. Dropout is a regularisation technique where randomly selected neurons are ignored during training, which means that their contribution to the activation of downstream neurons is temporarily stopped during the forward propagation, and subsequently any weight updates during the back-propagation stage are not applied. Figure 4.15 shows an example of how dropout works.



(a) Standard Neural Net (b) After applying dropout.

Figure 4.15

In the paper which introduced the concept of dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014), the recommended value for randomly dropping out neurons is 0.5, however for this experiment we will use a value of 0.1 since the FC layer is much smaller in this experiment compared to the original paper and the previous experiments did not show a huge deviation between the training and validation lines.

Model Architecture

As mentioned above the model architecture is shown in figure 4.14.

Hyper-Parameters

The model hyper-parameters are unchanged from the previous experiment.

Results

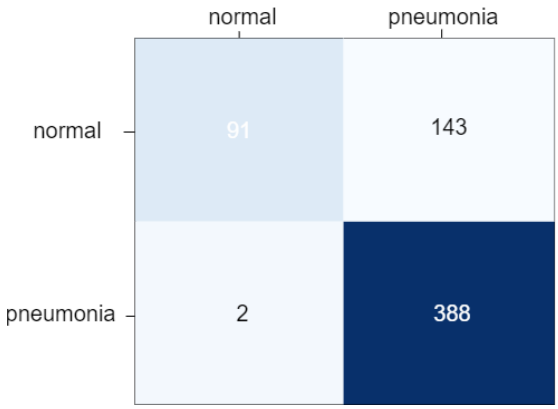


Figure 4.16: Testing results after adjusting architecture

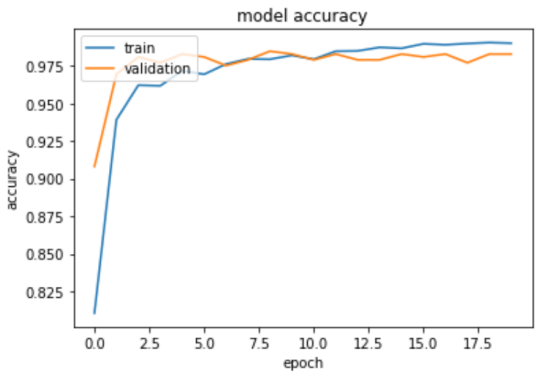


Figure 4.17: Model Accuracy

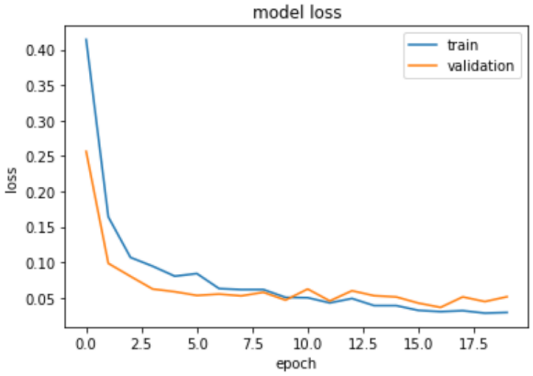


Figure 4.18: Model Loss

Figure 4.16 shows the results when testing the trained model on the test set. The results show a small increase in **accuracy** from the previous experiment of 76% to 77% and **specificity** has increased a small amount from 0.37 to 0.38, hence an improvement in the number of true negatives and false positives . **Recall** drops slightly, but remains high due to the imbalance which favours the positive class. The model accuracy and loss for this experiment shows that the training and validation lines are stable during the training process and does not show signs of over-fitting.

The changes made in this experiment did not improve performance dramatically since the aim was to increase the number of true negatives to a point where there is at least more true negatives than false positives. However, so far this experiment with model architecture and hyper-parameters, has produced the best results on specificity which is the value we are trying to increase.

4.3 Transfer Learning

A novel technique used in deep learning, is to harness pre-configured models that have been optimised for feature extraction. This approach is called transfer learning, where we take advantage of an existing CNN's convolutional layers as a feature extractor to output feature maps used as inputs to the FC layer. This type of learning has shown to overcome challenges such as limited dataset size and training from scratch (Rajpurkar et al., 2017). Examples of pre-trained models include VGG16 (Simonyan & Zisserman, 2014), which is the model that will be used in this experiment to see if transfer learning can improve results. Figure 4.19 shows the model architecture for VGG16, where a total of 13 convolutional layers are applied to the input image.

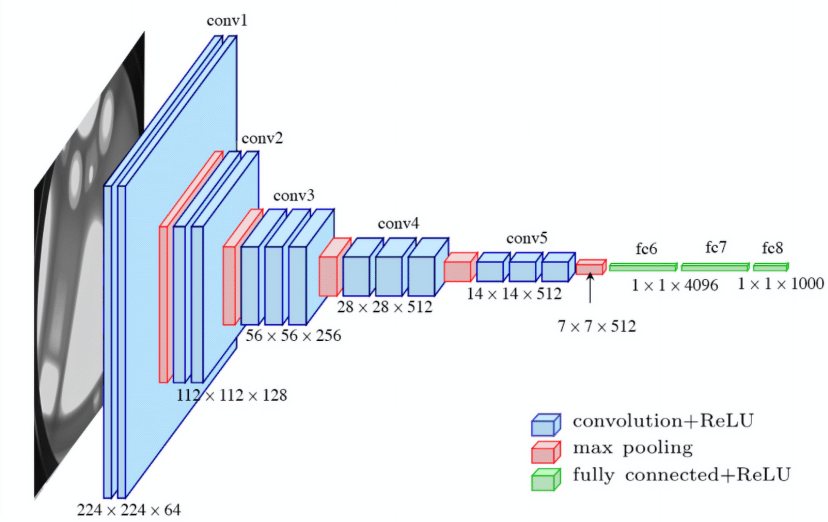


Figure 4.19: VGG16 Architecture

There are different ways in which transfer learning can be used, however in this experiment we simply use VGG16 as a feature extractor and attach our own FC layer at the end for the classification process. Figure 4.20 is a visualisation of how transfer learning is used in this experiment. To use VGG16's convolutional layers exclusively, as feature extractor, we froze the convolutional base and only trained the FC layer. Another change that had to be made to use VGG16, was to use a 250x250 input size instead of a 224x224. Lastly, VGG16 is trained on three dimensional data(RGB) so we had to increase the dimensions of the X-ray images from one to three.

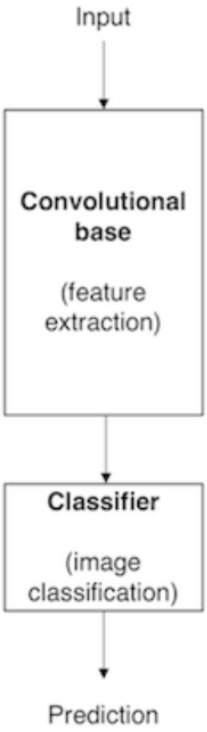


Figure 4.20: Transfer Learning technique

Model Architecture

The model architecture used for this experiment, uses the convolutional layers in figure 4.19, and the FC layer from the previous experiment as it showed the best performance.

Hyper-Parameters

The hyper-parameters remain unchanged from the previous experiment.

Results

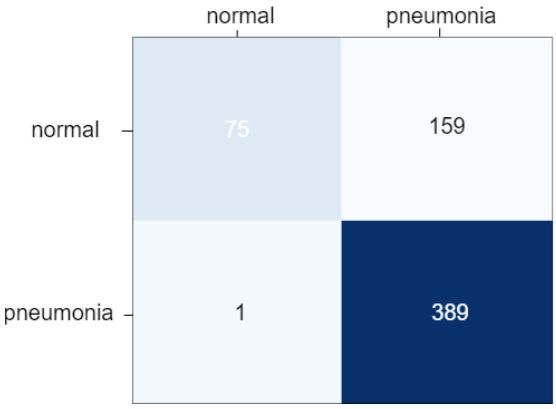


Figure 4.21: Testing results using VGG16

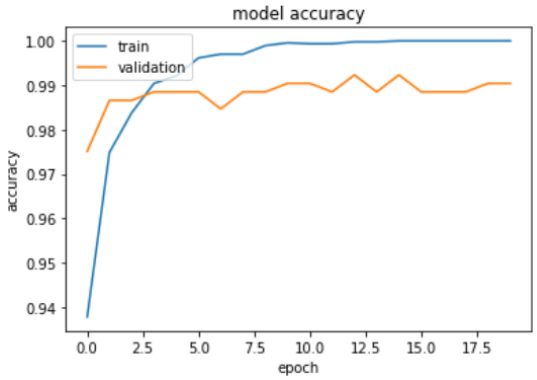


Figure 4.22: Model Accuracy

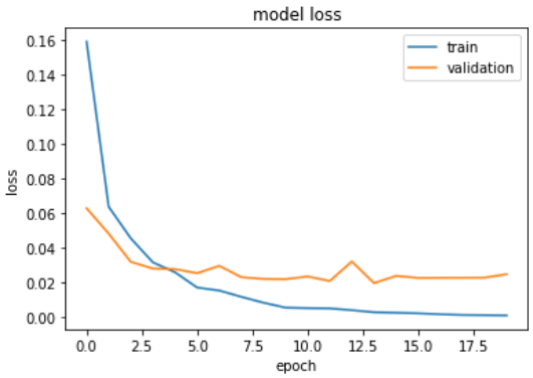


Figure 4.23: Model Loss

After reviewing the test results we can see this experiment produced sub-optimal results in comparison to the previous experiment. Analysis of the confusion matrix in figure 4.21, shows that accuracy is 74% compared to 77% from the previous experiment. The aim of this experiment was to increase the specificity score and hence increase the true negatives. However, this experiment was not able to suffice that aim.

From this transfer learning experiment, we concluded that perhaps transfer learning 'out of the box' as a feature extractor did not perform as well compared to a model with convolutional layers optimised for this specific dataset. However, further work could be done into adjusting the VGG16 model, such that a select few convolutional layers are allowed to train. The intuition behind this is based on a previous comment, where the earlier filters detect lines and edges compared to later filters which deal with the high level feature, such as the area of the infection. Hence, deeper filters could be fine-tuned to build a model with better results.

4.4 Augmentation

After optimising results via hyper-parameter tuning, the number of normal cases diagnosed as pneumonia by the model is larger than the number of cases correctly identified as normal. This issue is mainly due to the reason that the model is trained predominantly on the pneumonia class, hence, the normal class is under-represented. A common approach to overcoming the issue of low **specificity** is to find ways of oversampling the under-represented class. Since this project could not acquire new X-ray images from patients, data augmentation was used which is a common approach to improve the performance of computer vision systems. Data augmentation can be used in various ways such flipping, rotating and shearing. For this specific experiment, we chose to over-sample the normal class by rotating images randomly between -10° and +10°. Figure 4.24 shows a healthy X-ray taken from the dataset which can be rotated to create a new sample as shown in figure 4.25

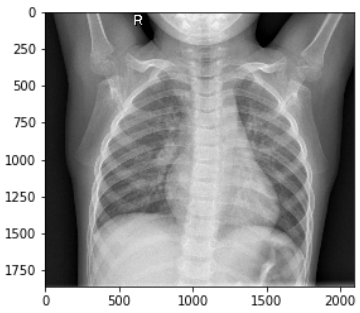


Figure 4.24: Before Augmentation

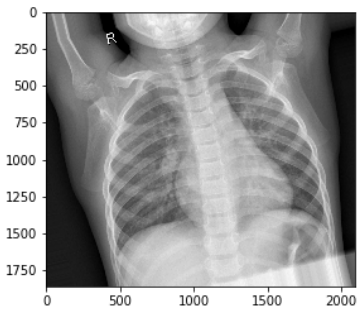


Figure 4.25: After Augmentation

The next step was to decide how much the normal images were to be oversampled by. We decided to double the amount of normal images in the training set to counteract the dominance of the pneumonia class. The validation data used to track the performance of the model against unseen data over the 20 epochs was not given the augmented images and uses 10% from the original training set.

Model Architecture

Since the best performance from the previous experiment showed that six layered CNN outperformed all other model architectures including transfer learning, the model architecture from section 4.2.2 will be used. Figure 4.14 shows a detailed look of the model architecture

Hyper-Parameters

Hyper-parameters are unchanged from the previous experiment.

Results

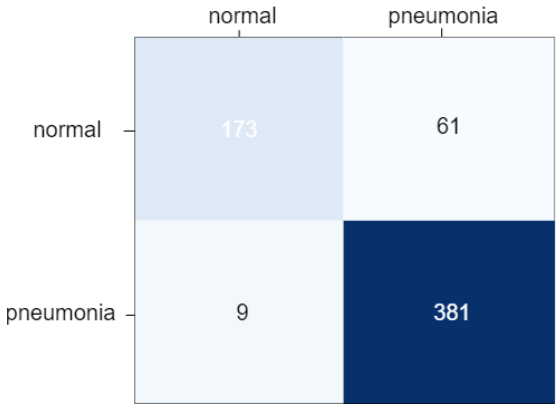


Figure 4.26: Testing results after augmentations

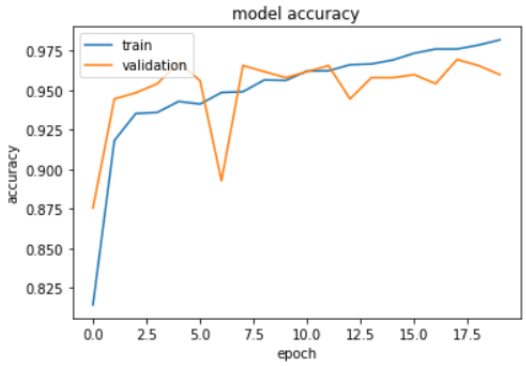


Figure 4.27: Model Accuracy

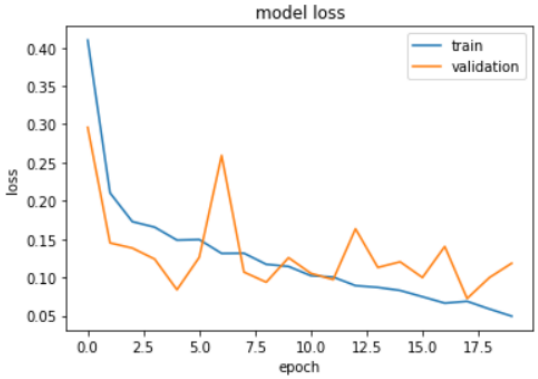


Figure 4.28: Model Loss

Figure 4.26 shows the model result on the test set after applying the augmentation techniques to increase the number of normal X-ray images. What can be observed is that there is a substantial increase in true negatives and drop on false positives. The overall accuracy of this experiment was 89% compared to the previous optimal results in section 4.2.2, which was 77%. As a result of applying augmentation, specificity score nearly doubled from 0.38 to 0.74. After applying the augmentation it was noted that recall decreased, however, this was mainly due to an increase in the normal samples which ultimately has an affect on the models weights. Figure 4.27 and 4.28 explain the model accuracy and loss after 20 epochs, which shows that the training accuracy steadily increases and could be ran for more epochs until it starts to plateau. The validation loss generally decreases from start to end, but there seems to be fluctuation and ultimately indicates a case of over-fitting. We ran the same experiment again, but

this time with 50 epochs to allow the training accuracy to flatten out and observe the affect it has on the test score.

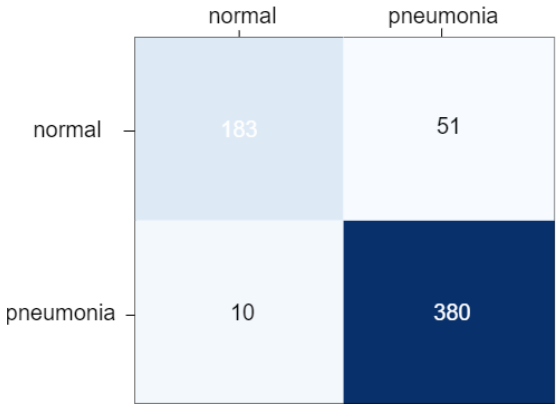


Figure 4.29: Testing results after increasing epochs

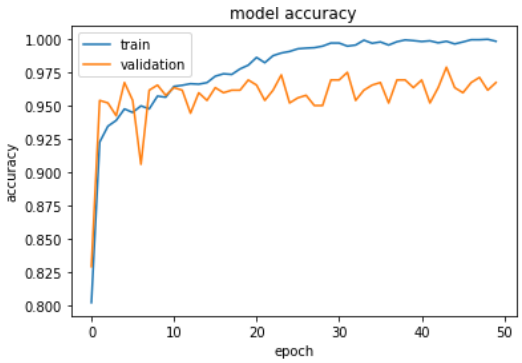


Figure 4.30: Model Accuracy

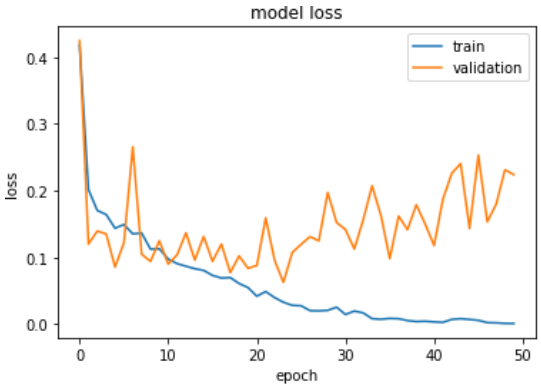


Figure 4.31: Model Loss

From figure 4.30 and 4.31, we can see that running the same experiment for 50 epochs generally flattens out the training process and this seems to increase the number of true negatives, which ultimately leads to accuracy increasing to 90% and specificity increasing from 0.74 to 0.78. Again recall is slightly lower at 0.97 compared to 0.98, but this is expected as the model is trained with a combination of higher epochs and increased number of normal X-ray images. The model accuracy and loss shows a discrepancy between the training and validation even though the testing accuracy increases compared to the previous experiment. This could suggest that the training and testing images are quite similar.

4.5 Summary of Results

Experiment	**Accuracy(%)**	**Recall**	**Specificity**	**Precision**	***F1-Score***
3Conv+3FC	77%	0.99	0.38	0.73	0.84
VGG16	74%	0.99	0.32	0.71	0.83
Augmentation	**89%**	0.97	**0.74**	**0.88**	**0.93**

Table 4.1: Summary of Experiment Results

Table 4.1 is a summary of the experiment results which shows the most important metrics when evaluating the model performance. The table also summarises the most important experiments as a result of tuning certain parameters. An additional performance metric used to summarise a models performance is the F1-score which provides a harmonic average of precision and recall. The F1-score is represented with the following formula:

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

What can be observed from the table above is that although the six-layered network and the transfer learning approach resulted in a very high recall score, both experiments appeared to show a bias towards the positive class, which is why specificity is low for both experiments. The final F1-Score for the two experiments were quite similar. However, a much smaller network with optimised convolutional layers achieved better results across all the metrics(except recall)compared to the transfer learning model.

After finding that a simpler model produced a higher F1-score and specificity, we needed to counteract the bias towards the positive class by introducing data augmentation to double the number of normal samples via random rotations which ultimately increased performance across all metrics, except recall as the model was fed a more balanced distribution of positive and negative samples. However, recall largely remained close to one, showing that the model did very well in recognising the infection in the images while increasing in specificity and ultimately boosting the F1-score.

4.6 Model deployment & Interface

One of the requirements for the project was to ultimately show how a model could be trained and deployed, demonstrating the applications of deep learning through an interface. To do so we created a front-end application using **Streamlit**, which is a open-source Python library that allows construction of web-apps for machine learning and data science purposes.

The web-apps purpose is to show how deep learning can be used to provide a fast diagnosis of medical images efficiently. A core principle of the web-app was to provide users with a diagnosis and a probability of how confident the model believes the infection is present, or equally, how confident it is in diagnosing the patient as healthy. To extract the model confidence we can use the last layer of the model which includes 2 neurons and a Softmax activation function, as show in figure 4.14. Softmax squashes a vector in the range of 0-1, and all resulting elements add up to one and is applied to the output score of the two neurons, s. The Softmax function cannot be applied independently to each s_i since it depends on all element of s. For a given class s_i, the Softmax function is computed as follows:

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}}$$

Where s_j are the scores inferred by the model for each class in C. Note that the Softmax activation for a class s_i depends of all the score in s.

The process of taking a user input of an X-ray image follows the same principles that is required when training the model, where the input image must be pre-processed before passing it through the model and extracting the Softmax score. This process is shown in figure 4.32, and figure 4.33 shows an example of the interface where a user uploads an X-ray image and is provided with the model prediction and confidence for each classification.

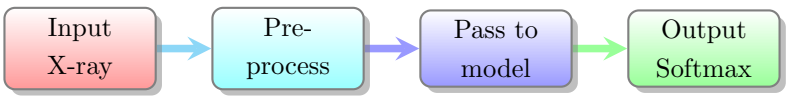


Figure 4.32: Process of diagnosis

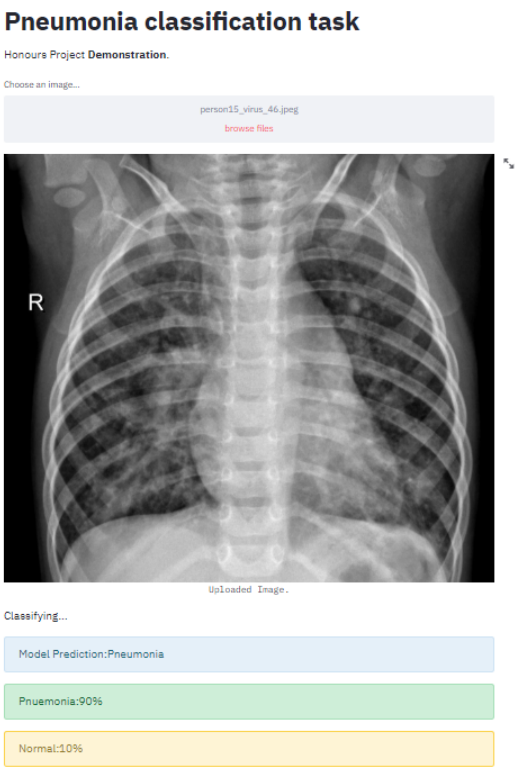


Figure 4.33: Web-app

Chapter 5

Evaluation

In this chapter, we will evaluate the project success on the requirements that were defined before the implementation stage, and provide a discussion on the results and key findings that were observed during implementation.

5.1 Overview

The project requirements laid out in chapter 2, was created using the MoSCoW framework which allows us to categorise certain requirements based on their level of importance. In the end, all of the "must" haves for this project were implemented, since this paper applied well know deep learning models like ConvNets. Fine-tuning of the hyper-parameters in order to improve results was carried out, and that was a "must" have for the project, as it showed continual process of improving results as depicted in figure 2.1. The project made use of common neural network libraries such as Keras and Tensorflow for fast prototyping which was also a "must" have. We met the criteria of complying with GDPR rules, as X-rays from patients did not include any Protected Health Information(PHI).

If results from the implementation stage showed good performance across various metrics, which it did , it was a "must" that a simple interface was created to demonstrate deployment of a model. This was achieved using a Python framework called Streamlit, which allowed us to create a web-app accessed through all major browsers which aims to provide a model prediction and confidence score via a Softmax output as percentage, when a user inputs an X-ray that needs to be diagnosed by the model.

Some requirements that were categorised as a "should" have, were not achieved. This was either due to time, or, limited dataset access. The impact of not implementing the

following requirements does not affect the outcome of the project, but meeting those requirements would have improved the overall research.

The first was being able to extend the classification pipeline to other pathologies. Since most of the development time was spent pushing the model performance for pneumonia classification, it was not possible to do so.

Another requirement that would have added to the overall research was applying pneumonia detection as well as classification, which allows a person to understand the area of lungs, which is most indicative of the model prediction. However, because the time spent increasing the classification metrics was the primary purpose, this was not possible.

The last requirement, which was classified as a "could" have, was to deploy the model to a cloud web service. However, upon reflection, we decided this was not part of the requirements list, as the deployment of the model would need to be rigorously examined before letting others access it as a potential diagnostic tool which is not verified by governing a body.

Overall, all of the essential requirements were met for this project given its limitations and it is also important to reflect on the key takeaways from the implementation stage.

5.2 Implementation Discussion

While exploring the dataset, although we did not have much knowledge of where the pathology was present, it was evident that each X-ray image was of different size and quality. As mentioned in the dataset section 3.3, we hypothesised that different X-ray machines could have been used when collecting the images and the amount of radiation applied to perform the X-ray, may vary (Bradley, 2008). This presented some challenges when training the model as resizing of images can sometimes cause a reduction in performance. Although this is not definitive, future research could be performed to examine the effect image sizes and radiation dose has on the model performance.

Moreover, although chest X-rays are the most common form of image modality, a public dataset might not be representative of either class(normal or infection). The impact class imbalance has on the training phase of a model can cause a bias towards one class, and this is evident in figure 4.16 and 4.21 , which shows a large number of false positives in the initial experiments carried out. In this body of work, the main aim was to automate the classification of pneumonia in chest X-rays, and the dataset supplied had a higher number of positive cases of the infection, compared to negative cases which

helped with the aim of this paper, but introduced problems like low specificity for some experiments.

After running multiple experiments show in chapter 4, we observed that after end-to-end training and hyper-parameter tuning of a six-layered CNN, it outperformed an 'out-of-the-box' transfer learning approach using VGG16. The VGG16 model was used exclusively as a feature extractor by freezing the convolutional layers and attaching the same fully connected architecture as the six-layered model. The six-layered model outperformed the transfer learning approach across all metrics, except recall, including accuracy, specificity, precision and F1-score. **Results suggest that training a model end-to-end was better suited compared to the transfer learning approach, for the specific dataset used in this paper.**

We also noted that a novel approach to combatting issues of low specificity was to over-sample the under-represented healthy X-ray images. After applying random rotations on the healthy images to create new samples, we retrained the optimised six-layered CNN with an increased number of normal X-ray images and observed that specificity almost doubled, with all other metrics increasing except recall, which dropped slightly but remained close to a value of one which is desirable. .

Chapter 6

Conclusion

In this chapter, we will conclude the project by taking into account the things that were learnt while trying to achieve the aims and objectives of this paper. We will further end by considering the limitations of the paper and discuss some of the future work.

6.1 Overview

In this project, we were able to apply deep learning methods to classify highly complexed X-ray images. We looked at several techniques to push the model performance

on the test set by iteratively fine-tuning the model via key hyper-parameters which control the training process. We evaluated the performance of each result on several important metrics.

Two-thirds of the global population lacks access to radiology diagnostics(cite), even when medical imaging resources are available. After pushing the performance metrics on the test set, we demonstrated a simple interface that could be used to provide quick diagnostic capabilities through a trained and tested deep learning model.

References

Alexander, A. (2019). *Mit 6.s191: Convolutional neural networks.* Retrieved from https://youtu.be/H-HVZJ7kGI0?t=6

Bradley, W. G. (2008). History of medical imaging. *Proceedings of the American Philosophical Society*, *152*(3), 349–361.

Castellino, R. A. (2005). Computer aided detection (cad): an overview. *Cancer Imaging*, *5*(1), 17.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 ieee conference on computer vision and pattern recognition* (pp. 248–255).

DrMatthew, L. (2018). *Ai in radiology at stanford: Rise of the machines.* Retrieved from https://youtu.be/Gigd1rkZTSE

Guendel, S., Ghesu, F. C., Grbic, S., Gibson, E., Georgescu, B., Maier, A., & Comaniciu, D. (2019). Multi-task learning for chest x-ray abnormality classification on noisy labels. *arXiv preprint arXiv:1905.06362*.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the ieee conference on computer vision and pattern recognition* (pp. 4700–4708).

Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., ... others (2019). Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. *arXiv preprint arXiv:1901.07031*.

Islam, M. T., Aowal, M. A., Minhaz, A. T., & Ashraf, K. (2017). Abnormality detection and localization in chest x-rays using deep convolutional neural networks. *arXiv preprint arXiv:1705.09850*.

Kermany, D. S., Goldbaum, M., Cai, W., Valentim, C. C., Liang, H., Baxter, S. L., ... others (2018). Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, *172*(5), 1122–1131.