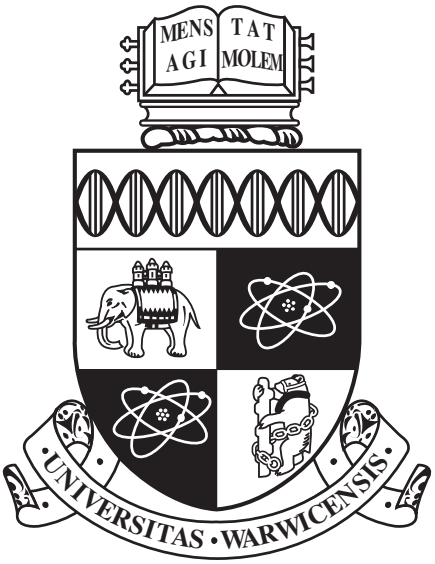
University of Warwick
Department of Computer Science

CS310

Developing AI-based tumor detection techniques for CT scans of lungs



Supervisor: Dr. Ligang He
Author: Cem Yilmaz
Year of Study: 3

Abstract

Early diagnosis of cancer, particularly lung cancer, substantially increases the outcome of the patient due to the reduction of the waiting time and the increase in the probability of finding a recoverable lesion. This thesis explores practical diagnostic approaches that exploit the benefits provided by deep learning to accelerate the time to classify. This study considered two deep learning architectures: a Convolutional Neural Network (CNN) inspired by AlexNet and a Convolutional Long Short-Term Memory Network (ConvLSTM). After many tests, the results showed that the designed CNN architecture provided high F1 and TNR scores (98%). Contrary to our assumption, ConvLSTMs performed poorly in all metrics. ConvLSTM's underperformance is attributed to the architectural simplicity, which could have learned more effectively from the complex imaging data despite much tuning. This study establishes that CNNs are more adept at classifying cancer because they can handle complex features more robustly, delivering better results. Another limiting factor is the computational complexity of ConvLSTM architecture being bottlenecked by our GPU. Our success of CNN's architecture opens new avenues for applying deep learning approaches in clinical diagnostics to ensure early and rapid diagnosis.

Keywords

Deep Learning, Image Analysis, Carcinoma, Convolutional Neural Network, Recurrant Neural Network, Medical Imaging, Long Short-Term Memory

Acknowledgements

I would like to thank my parents and my sibling for providing me the opportunity and the finances to study in the University of Warwick. Without their support, this dissertation would have not been written. I would also like to acknowledge and thank my supervisor, Dr. Ligang He, for his continuous support and suggestions in my dissertation. Lastly, I would like to acknowledge and thank Vesnet Medical Diagnostic Centers (MDC) for providing me the data set I needed.

CONTENTS

List of Figures

1 Introduction

Lung cancer, medically abbreviated as lung carcinoma, is one of the deadliest forms of modern carcinomas in the United Kingdom (UK). It is also by far the most common cause of cancer death in the UK, accounting for around a fifth (21%) of all cancer deaths in the years 2017-2019 (Cancer Research UK, 2022). Although the causes of lung cancer per individual vary, the main suspect of lung cancer cause is the modern lifestyle, such as smoking and pollution. With the rise of modern problems caused by modern development, it is necessary to develop methods to mitigate or adapt.

Unfortunately, lung cancer is often diagnosed at advanced stages due to delayed symptoms, while early detection dramatically improves survival rates (World Health Organization, 2023). As such, with the problem of lung cancer, we must adopt early detection. The development of new robust technologies such as computer tomography (CT) for lungs has drastically increased the efficiency and detection of lung cancers. Published research suggests that CT is routinely and commonly used for detecting lung cancer due to its efficiency (Nooreldeen & Bach, 2021).

Recent development of lung scanning technologies and artificial intelligence opens a new area for research and development. Given the recent development and discovery of both artificial models, such as the convolutional neural network and deep learning, there is new room to develop methods of detecting cancer in the lungs with heightened precision and accuracy. Furthermore, through automation or assisting decision-making using modern artificial intelligence methods, it is possible to decrease waiting times for CT scan results and make lung scanning a regular and uncostly practice.

However, detection of cancer is not easy. Cancerous tumours come in various shapes, sizes, and locations inside a lung. Consider figure 1 below:

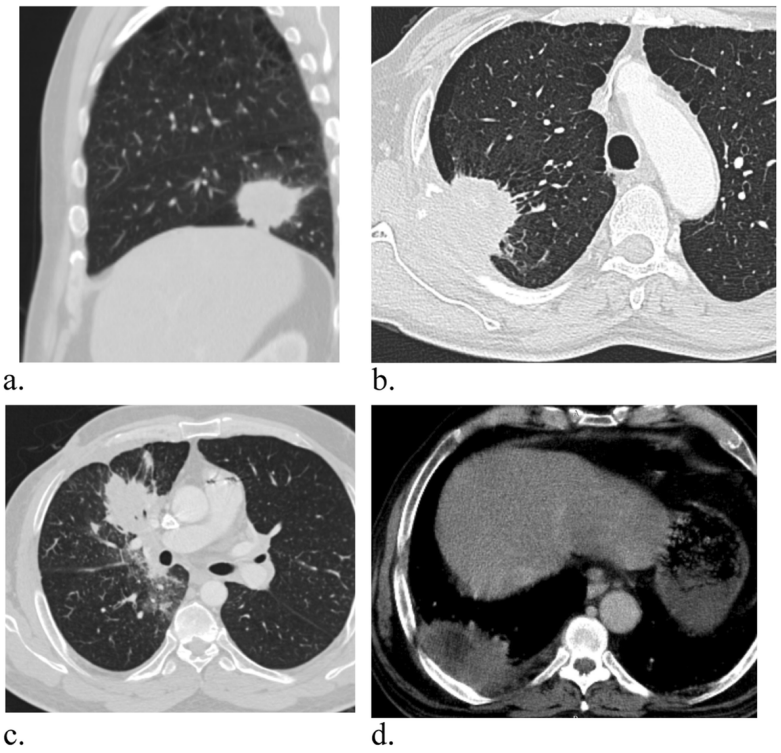


Figure 1: Types of cancerous lung tumours

Case a depicts a tumour on a lower lobe attached to the mediastinum. Case b depicts a mass lesion tumour that is invading the chest wall. Case c depicts cancerous tumour invasion on bronchial branches. Finally, case d depicts a thick-walled lesion with pleural invasion (Gharraf, Mehana, & ElNagar, 2020). Given the hard problem, where cancerous tumours of various shapes and sizes, it is essential to develop a robust solution using deep learning.

The purpose of this dissertation is to develop and compare different deep learning methods in solving lung cancer detection problems. We will examine several methods and their effectiveness in solving this problem by comparing their error rates in the conclusion stage and reflecting on the results. The development of each technique will be inspired by possible existing literature and will be present in throughout our text.

2 Project Management

2.1 Objectives

At the beginning of the dissertation, several objectives were put in MOSCOW form. Some of these objectives were unrealistic to succeed in the time frame, e.g., implementing a complete UI; other objectives were too lenient, e.g., $< 50\%$ false positive rate from the sample of all failures. This cause is the project's uncertainty and initial unfamiliarity with the topic. These objectives have been adjusted for the final report in the context of implementation. The achieved and very slightly modified objectives of this final report include:

- Make software compatible with data format submitted using DICOM.
- Must be able to sort DICOM data to ensure correct learning.
- The usage, analysis, and comparison of 2 different methods.
- $\geq 95\%$ accuracy in a CT-scan from 1 method.

The original objectives can be found in appendix A of this dissertation.

2.2 Software Development Method

Due to the uncertainty and unfamiliarity with the topic when we started, the software development method chosen was agile methodology. The sprints were carefully and effectively planned early and proven useful. The sprints designed were as follows:

1. Initial research on existing methodologies and papers will be conducted. The initial theory around machine learning, deep learning, and neural networks will be developed.
2. A script that sorts DICOM data into readable data such as .png will be programmed.

3. Further research done on the initial steps of AI development. At this point, the first framework for the AI will be pinpointed.

4. The development of the initial AI framework begins until it is finished.

5. Data is separated in 2 ways: data used for training and data used for testing (and possibly validating). After separation, data is translated and learned by the AI.

6. The learning will be tracked: there will be an attempt to find the best parameters and hyper-parameters possible for the data I have been given. As such, training errors and test errors will be tracked, and methods such as k-fold validation will be used.

7. Accuracy is tested of the AI framework with data that has been kept in reserve for this exact purpose only. Accuracy will be measured in specific ways, e.g., overall accuracy, false positives, etc.

8. The method is either modified or a new method is proposed.

9. If a new method is proposed, steps 3 to 5 are repeated. However, initial AI will likely be flawed, and therefore, slight changes to parameters or methods are likely before the proposal of a new method.

As seen above, several steps have dependencies; for example, we may only start training the neural network after sufficient workable data. However, with careful planning, these have been foreseen, and the workflow was continuous.

2.3 Unforeseen Problems

Several unforeseen problems were identified during the development of this project. Each unforeseen problem also incorporated a solution that we developed. This section will discuss the specific problems encountered and solutions administered.

- Data disparity - data disparity refers to how the data is distributed. In particular, approximately 20% of the data obtained shows signs of cancer tumours. As such, there is data disparity, which can cause unwanted behaviour in training. To mitigate this problem, we have designed our models in consideration of the data disparity by considering things such as weight, bias, etc., in calculations of our deep learning architectures.

- Obtaining the data promptly - obtaining data when needed has unfortunately turned into a logistical nightmare due to the hospital's load of doctors and ITicians. This problem was approached and solved by applying extra pressure in conversations.

- GPU limitation - GPU limitation refers to our current GPU's ability to handle data and affects the architectures we are able to design. Our current GPU is an RTX 4080, which only has 16 GB of vRAM, causing issues. Our adaption to this problem includes designing neural networks with smaller batch sizes, smaller architectures, and other optimisations.

- Lack of literature on using ConvLSTM for medical imaging - we initially thought the idea would be well-explored. However, research on ConvLSTM is minimal, giving us little information on implementing our architecture. This problem was adapted through lots of trial and error and the design of new, fresh ideas in the architecture that have not been done before.

2.4 Legal, Ethical and Social Issues

No legal, ethical, or social issues are associated with this project. However, due to the nature of sensitive data, the data gathered was handled with extreme caution. Further information can be found in sections 4.1 Data Gathering and 4.2 Data Treatment.

3 Background

3.1 Perceptron

Perceptron is a type of artificial neuron inspired by the biological neuron. It is one of the most commonly used artificial neuron models found in many neural networks, including the ones we will use for this dissertation. A perceptron has a finite number of i inputs, denoted as $x_1, x_2, ..., x_i$. Furthermore, from its definition, we have that $\forall i, x_i \in \mathbb{R}$. Each input x_i is defined to have an associated weight to it. Weights let the neuron adjust the "importance" of a specific input. Formally, the neuron begins its calculations after multiplying the input by its weight, that is:

$$x_i w_i$$

The perceptron run calculates the sum, z, which considers all inputs and weights defined as follows:

$$z = \sum_i x_i w_i$$

However, the sum z lacks a linear transformation. Therefore, a bias b is introduced. The necessity of bias b can be shown if we represent everything using linear algebra. Let us consider

$$\mathbf{w} = \begin{bmatrix} w_1 & ... & w_i \end{bmatrix}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ \dots \\ x_i \end{bmatrix}$$

Then, using a dot product, we can see that we obtain what is similar to the equation of a line without a linear transformation:

$$\mathbf{w} \cdot \mathbf{x}$$

However, if we introduce the bias b, we get the full equation of what is similar to a line:

$$\mathbf{w} \cdot \mathbf{x} + b \tag{1}$$

Finally, we introduce the activation function, which is applied to equation 1. The activation function controls whether a neuron should fire or not. In particular, a common activation function is the Heaviside function $H(x)$, defined as

$$H(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$$

Therefore, if we define f to be the final output of a perceptron, we can define it as

$$f = H(\mathbf{w} \cdot \mathbf{x} + b)$$

This also provides the intuition that we can view b as a way of shifting the activation function. A perceptron is usually depicted using a Rojas diagram, pictured below in figure 2.

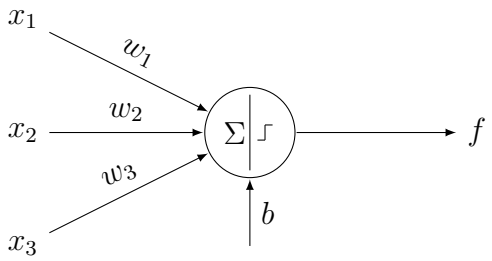


Figure 2: Rojas diagram of f

3.2 Artificial Neural Networks (ANNs)

Artificial neural networks are a connection of multiple neurons in an attempt to learn and create more complex functions. As stated, a single perceptron is just a line; however, if multiple neurons are connected together, we are able to create more complex functions. In order to ensure and control learning, we introduce new parameters that are necessary:

- Learning rate η - determines how fast the parameters should update within each neuron, e.g., adjusting the size of weight w. This parameter is set manually as needed.
- Loss function $\mathcal{L}$ - determines the difference in predicted output compared to the ground truth. For example, the loss function can be defined as the mean squared error between the input and expected output. However, we note that the definition of loss function can vary.
- Batch size B - number of samples/inputs used in an iteration of training the network.
- Epoch E - represents one complete pass through an entire dataset.

An artificial neural network typically consists of three layers: the input layer, the hidden layer, and the output layer. The input layer is usually the first column of layers, and this layer does not have an input from any perceptron. Instead, its input is the chunk of data itself. The hidden layer may consist of multiple columns, with a different number of layers on each. The output layer's output is the final output, and it does not provide any information to other neurons. Each column of neurons connects to every next neuron found in the next column, providing its output. A demonstration of the described can be found in the figure below 3.

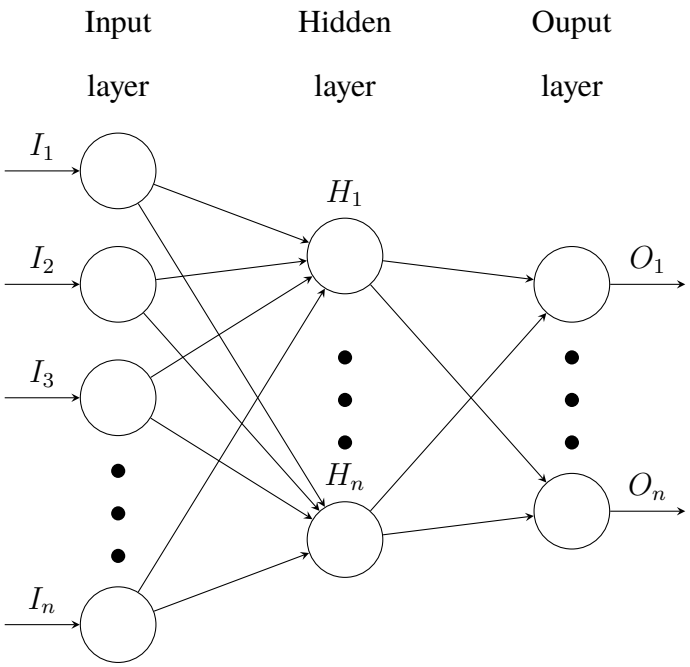


Figure 3: Artificial Neural Network

Furthermore, for ANNs, it is typical to adjust the activation function of neurons that are being used. In a perceptron, it is the Heaviside function, as discussed. However, it can be adjusted to other activation functions, such as the sigmoid $\sigma(x)$ function. In particular, the Heaviside function is an "all or nothing" - it either gives minimum output or maximum output. Introducing alternatives such as the sigmoid function ensures that we lose less information, with the difference can be seen below in figures 4 and 5 below.

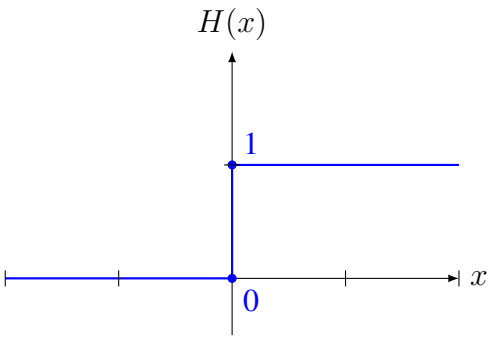


Figure 4: Heaviside step function

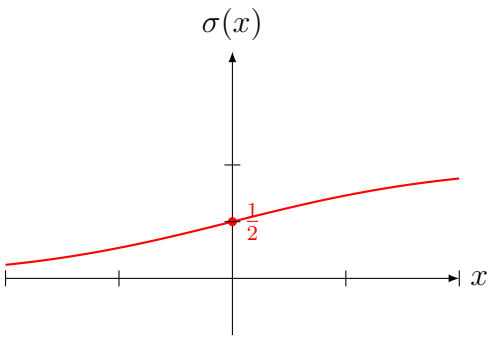


Figure 5: Sigmoid function

Indeed, there are many activation functions, such as $\tanh(x)$ or $\mathrm{ReLU}(x)$, which are chosen depending on needs of a neural network's architecture.

Lastly, learning happens through the process of forward propagation and back-propagation. Forward propagation refers to the phase when neurons are fired, with the network signals moving left to right in figure 3. The output is then computed and compared to ground truth using the loss function. Back-propagation then occurs, updating parameter values of each neuron. Formally, we define back-propagation using partial derivatives. Partial derivatives are necessary to measure how each neuron affects the error, allowing us to update each neuron individually. Let us define the ground truth to be $\mathbf{y}$; then, we can define the back-propagation as for the last layer. Let us denote the output layer as I, the first layer as 1, and the hidden layers as i. Then,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{a}^I} = \frac{\partial \mathcal{L}}{\partial \mathbf{y}} \cdot \frac{\partial \mathbf{y}}{\partial \mathbf{a}^I}$$

Where $\mathbf{a}^L$ represents the output of neurons in the last layer after applying the activation function to the weighted sum of inputs. In other words, we classify $\mathcal{L}$ as the loss function which can be defined as suited. Propagating backward, we do it similarly for hidden layers:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^i} = \frac{\partial \mathcal{L}}{\partial \mathbf{a}^i} \cdot \frac{\partial \mathbf{a}^i}{\partial \mathbf{z}^i}$$

where z^i is the weighted input of layer i. Once the partial derivative values are computed, we update the weight and the bias using some algorithm. Typically, it is gradient descent, which is defined as follows:

$$\theta^i_{nm} \leftarrow \theta^i_{nm} - \eta \frac{\partial \mathcal{L}}{\partial \theta^i_{nm}}$$

where η is the learning rate as defined, and θ is the weight or bias of connecting neuron m in layer $i-1$ to neuron n in layer i. As such, because the values converge to the truth values, we assign give random weights and biases for the initial configuration of a neural

network.

3.3 Convolutional Neural Networks (CNNs)

Convolutional neural networks were introduced to resolve a fatal flaw of ANNs. In particular, like our data, if we have an image of 512×512 in resolution, we obtain a total of $262,144$ pixels, namely, the actual input we want to feed into our network. This equates to $262,144$ input layers, which is computationally intensive and not practical. As such, CNNs introduces the idea of pooling and convolution of input data, which we will discuss in detail.

There are many different methods of pooling, such as max pooling, average pooling, minimum pooling, etc. However, for simplicity, we will discuss max pooling, as it provides enough background information on how pooling functions. Let us consider an input of an image with dimensions $h \times w$, where $h, w \in \mathbb{N}$. Pooling considers a square $p \times p$, $p \in \mathbb{N}$, where usually $p \leq \max(h, w)$. In the case of max pooling, we find the maximum value found in $p \times p$ and let this specific square be assigned to that number. We then repeat this by considering stride, i.e., the variable that sets the number of pixel shifts for each grid. In particular, it is common to see stride $= p$, to ensure consistent and non-overlapping calculations of the grid. By repeating this process, we try to retain the most important information about our image whilst decreasing the dimensions. Formally, we define max pooling as, with stride $= p$,

$$P_{i,j} = \max_{m=0}^{p-1} \max_{n=0}^{p-1} X_{i \cdot p+m, j \cdot p+n}$$

where p is the pooling size, $P_{i,j}$ is the output number in final image at i, jth location, and $X_{i \cdot p+m, j \cdot p+n}$ is the input image X, with a value at $i \cdot p + m, j \cdot p + n$.

However, notice that we still lose some information. As such, to mitigate as much infor-

mation loss as possible, CNNs also introduce a convolution layer, hence the name. We introduce a filter, which is designed to capture patterns in the data. It assigns weight to each position to signify the importance of a specific value in our input. Similar to pools, they have a size of $p \times p$ and typically stride $= p$, for which both values can adjusted as desired. The formal definition of the final output O for an image that we apply a convolution to is as follows:

$$O_{i,j} = f(\sum_{m=0}^{p-1} \sum_{n=0}^{p-1} I_{i+m,j+n} \cdot K(m,n))$$

Where I is the input, K is the weight of the input value at that specific position, p is the grid size, and f is an activation function that we specify. We note that grids avoid overlapping, and the filter's importance on a specific value is also adjusted as the CNN learns through iterations, implying that the CNN learns which features are more important to keep. It is also not unusual to see a convolution layer use more than a single filter K, which means that our image with a single color channel can have multiple dimensions other than height and width after the convolution layer.

Finally, we introduce the flattening layer, a crucial component in CNNs. Flattening layers serve the purpose of feeding final information after convolutional and pooling layers. That is, because each neuron is able to only take a single value input, the flattening serves a purpose of taking all values and putting them into a single vector for the CNN to be able to read.

Combining pooling, convolution, and flattening, we obtain the full architecture of a CNN. Using our knowledge of pooling and convolutional layers, we can derive formulae to predict the final dimensions of our image after applying all the layers. We can introduce a padding concept for convolution layers to ensure that we keep dimensions similar to the input. Let us consider the input dimension to be $H_{\text{in}} \times W_{\text{in}}$ and the out-

put dimensions to be $H_{\text{out}} \times W_{\text{out}}$. Then, we can create a relationship between them as follows:

$$H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} + 2P - p}{S} \right\rfloor + 1$$
$$W_{\text{out}} = \left\lfloor \frac{W_{\text{in}} + 2P - p}{S} \right\rfloor + 1$$

Where P is the padding, and $p \times p$ is the dimension of the convolution grid window. If we consider different filters to apply after each layer, we obtain that the final dimension is, for several filters F:

$$H_{\text{out}} \times W_{\text{out}} \times F$$

Similarly, for pooling, we get that

$$H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} - p}{S} \right\rfloor + 1$$
$$W_{\text{out}} = \left\lfloor \frac{W_{\text{in}} - p}{S} \right\rfloor + 1$$

We note that the formulae we have provided work for a single-channel image, i.e., a black-and-white image. Having multiple channels would require a slight modification of the formulae. And as discussed, it is typical to see $S = p$; therefore we can simplify our formulae as follows, with convolution first:

$$H_{\text{out}} = \left\lfloor \frac{H_{\text{in}} + 2P}{p} \right\rfloor$$
$$W_{\text{out}} = \left\lfloor \frac{W_{\text{in}} + 2P}{p} \right\rfloor$$

And for pooling:

$$H_{\text{out}} = \left\lfloor \frac{H_{\text{in}}}{p} \right\rfloor$$
$$W_{\text{out}} = \left\lfloor \frac{W_{\text{in}}}{p} \right\rfloor$$

Let us assume that we have a neural network of $p_c = 3$ (size of convolution grid), $p_p = 2$ (size of pool grid) with two convolution layers and pooling layers. Then, the CNN architecture, along with the output size after each layer, will look as follows in figure 6:

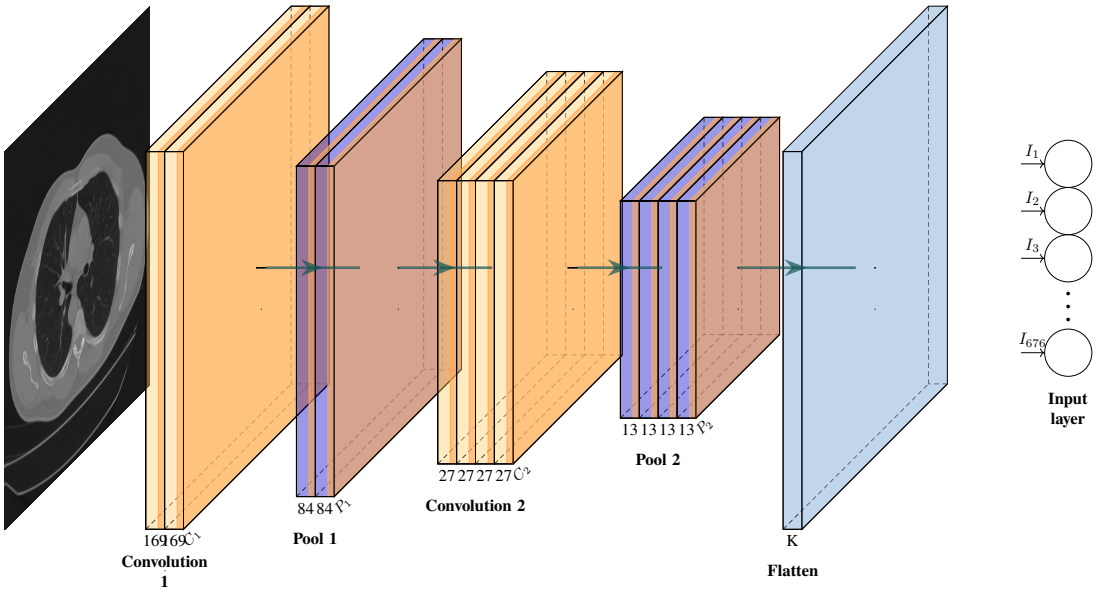


Figure 6: Example CNN Architecture

In conclusion, we show that CNNs are equipped with multiple techniques to ensure robust predictions of significant input. The CNN architecture can downsample the input size by utilising convolution and pooling layers while retaining crucial information for analysis. After down sampling using convolution and pooling layers, it is flattened and fed into input layer neurons, allowing CNN to learn.

3.4 Recurrent Neural Network (RNNs)

In this section, we will introduce the concept of recurrent neural networks, a specific architecture named long short-term memory (LSTM), and how to combine LSTM with convolution methods.

An RNN is a special type of ANN designed to learn using sequential data. Within our work, each patient's CT scan is, in fact, sequential data. That is, the CT scan creates data layer by layer over time. As such, RNN architecture may be effective within our project's context. In an RNN, each network neuron receives input from the previous time step, forming a feedback loop. There are three components which define an RNN to be a subset of ANN:

- Recurrent connections - by having recurrent connections between each node within the ANN, each neuron can receive data not only about the current frame of a CT scan but also from a previous frame, e.g., the ability to recognise that a carcinoma tumor begins to form.
- Hidden state - RNNs store a hidden state vector concerning the processed sequence. This hidden state is updated every time new information is received.
- Temporal processing - given that RNNs process sequential data one at a time, the RNN can capture temporal dependencies and patterns being formed in the image frame.

In this project, we are interested in a specific type of RNN, namely the LSTM. Unfortunately, basic designs of RNNs suffer from the vanishing gradient problem. As the sequence length increases in the RNN, the gradient magnitude is expected to decrease. There are occurrences where this can stop the neural network from updating itself and therefore stopping any training (Basodi, Ji, Zhang, & Pan, 2020). To mitigate this problem, we will develop the RNN using the LSTM architecture.

The LSTM has different components to capture the architecture of an RNN. The architecture breakdown of an LSTM cell can be found in figure 7 below:

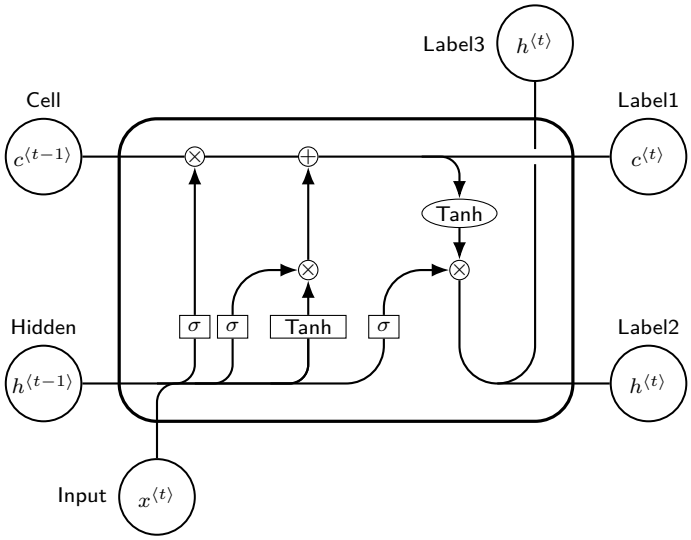


Figure 7: LSTM Cell

where t is the step time, $x^{\langle t \rangle}$ is the fresh data input at time t, $h^{\langle t-1 \rangle}$ LSTM's internal representation at the previous time step whilst $h^{\langle t \rangle}$ is the representation at current step, σ is the sigmoid function $\sigma(x)$ described previously, $c^{\langle t \rangle}$ is the cell state at time t that retains long-term information at time, and $\tanh(x)$ is the function defined as

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$\tanh(x)$ is a function similar to $\sigma(x)$, with the exception that the range is $(-1, 1)$, allowing for better gradient flow during training. We need to note the difference between c and h: h focuses on capturing short-term dependencies for prediction, whereas c is designed to retain long-term information for prediction. We now break down each sub-component of the LSTM cell as follows:

- Input gate i - this is a component of the LSTM cell that decides to extract useful information using $x^{\langle t \rangle}$ and $h^{\langle t-1 \rangle}$. In the context of figure 7, this is represented by

the second arrow pointing upwards from the left. Mathematically, we describe it as follows:

$$i = \sigma(w_i \odot [h^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_i)$$

where w_i and b_i are the associated weight and bias of the gate i respectively. The $\sigma(x)$ denotes the application of the sigmoid function for each element in a matrix in this context. The notation $[h^{\langle t-1 \rangle}, x^{\langle t \rangle}]$ denotes horizontal matrix concatenation of matrices/vectors $h^{\langle t-1 \rangle}, x^{\langle t \rangle}$ respectively. Finally, the notation $\odot$ denotes the Hadamard product, i.e., element-wise multiplication of two matrices. Formally, for two matrices $A_{i,j}$ and $B_{i,j}$ of the same size, the Hadamard product is defined by

$$(A \odot B)_{i,j} = A_{i,j} \cdot B_{i,j}$$

In other words, we observe that the input gate is responsible for extracting important information about the current and previous states by considering learned weights and biases.

- Candidate gate c - The candidate cell associated with the third upward arrow pointing at $\tanh$ from figure 7. Formally, like the input gate, we denote it as

$$c = \tanh(w_c \odot [h^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_c)$$

- Forget gate f - this gate is shown in figure 7 by the first arrow pointing upwards from left. It is formally defined by

$$f = \sigma(w_f \odot [h^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_f)$$

To be precise, it determines how much of the previous cell state $c^{\langle t-1 \rangle}$ should be

retained or forgotten when updating the current cell state $c^{\langle t \rangle}$.

- Output gate o - the output gate determines what information from the current cell state $c^{\langle t \rangle}$ should be passed on to the next time step's hidden state $h^{\langle t \rangle}$. Formally, this is defined similarly to other gates, albeit with its own weight w_o and bias b_o:

$$o = \sigma(w_o \odot [h^{\langle t-1 \rangle}, x^{\langle t \rangle}] + b_o)$$

We shall now discuss how the gates are combined in the LSTM cell to create the required final results. Combinations of gates are usually denoted by the circles with the operator used. For example, $\otimes$ denotes Hadamard product, and $\oplus$ denotes matrix addition. Using our gates, from figure 7, we observe that the input gate is combined with the control gate. In other words, by combining the two gates, the input gate modulates the influence of the control gate on updating the cell state, ensuring that only relevant information is added or removed. Let us denote this combination as $\widetilde{c}^{\langle t \rangle}$ and define it as such:

$$\widetilde{c} = i \odot c$$

Then, we can define the new cell state $c^{\langle t \rangle}$ using old cell state $c^{\langle t-1 \rangle}$:

$$c^{\langle t \rangle} = c^{\langle t-1 \rangle} \odot f + \widetilde{c}$$

Finally, the output gate is combined with the new cell state to obtain a new internal representation of our time step t:

$$h^{\langle t \rangle} = o \odot \tanh\left(c^{\langle t \rangle}\right)$$

This operation aims to ensure that the output gate controls which parts of the cell state are passed on as the final output.

By examining all the gates of LSTM, we gain insights into how an LSTM cell processes sequential data, retains long-term dependencies, and generates meaningful predictions. However, LSTM retains a fundamental flaw in its input: our input is of 512×512 resolution; therefore, it is very computationally complex. As such, we introduce convolution layers. The new architecture is called ConvLSTM, and its introduction is very recent, being published in 2015 (Shi et al., 2015). The state-of-the-art architecture is a hybrid model of convolutional layers from a CNN with a modification to a fully connected LSTM (FC-LSTM). An FC-LSTM slightly modifies the LSTM by adding more relationships between each gate. In particular, the modification can be seen in figure 8 (Chen, Li, Liu, & yi, 2021) below:

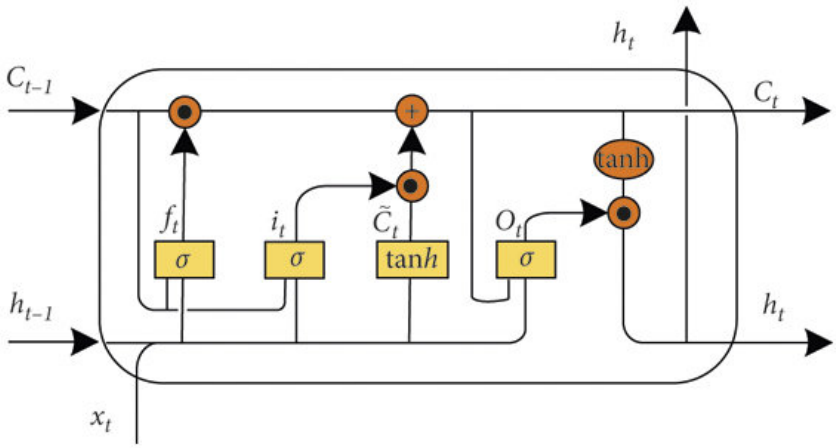


Figure 8: FC-LSTM Architecture

In this dissertation, we delve into a novel exploration of the performance of ConvLSTM in the specific context of medical image analysis of lungs, a topic that has not been extensively studied. We also compare its performance to that of a CNN, providing a unique perspective on these two models.

3.5 Metrics and Information

In this subsection, we will discuss the metrics and information we will use to measure the success of the models we create.

- Time to segment (TTS) - provides us a metric for the time taken by a neural network to produce segmentations of CT scans. It measures the computational efficiency of the segmentation process by checking the time taken. This is essential as time is important in cancer and obtaining results for patients.

- Accuracy - accuracy provides a basic understanding of how correct a model is compared to the ground truth. However, this may not be sufficient as the dataset is imbalanced (this will be further discussed in the Data section). That is, accuracy will be measured by

$$\text{Accuracy} = \frac{\text{No. Total Correct Predictions}}{\text{Sample Size}}$$

- Precision - this metric measures the proportion of true positives among all positive predictions. The focus of this metric is only the positive predictions.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

- Recall - similar to precision, except recall focuses on negative predictions. Recall focuses on capturing all positive instances and accounts for false negatives, as false negatives represent positive instances that the model missed.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- F1 - this metric is the harmonic mean of precision and recall, creating an overall measurement of the outcome of correctness by balancing both.

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- TNR - abbreviated as true negative rate, measures the proportion of actual negative

cases that were correctly identified as negative.

$$\text{TNR} = \frac{\text{True Negatives}}{\text{True Negatives} + \text{False Positives}}$$

- Loss - measuring loss is useful information for us to keep over time to measure the progress of our models and provide feedback on the effectiveness of manually set hyperparameters, e.g., learning rate and batch size.

To ensure the metrics are relevant and to the best possible scenario, we will consider problems of overfitting and underfitting data. That is, by using the F1 metric, we can predict if the current model is overfit or underfit. Specifically, a large gap between training and validation performance may suggest overfitting, while consistently poor performance on both sets may indicate underfitting. Similarly, we can check overfitting or underfitting and reinforce our knowledge by checking the loss parameter.

4 Data

This section will discuss how data relevant to our project is used, gathered, and treated for use in our implementation.

4.1 Gathering

A medical joint stock company, Vesnet, supplies this project's data. Vesnet is a diagnostics center in Kazakhstan equipped with medical imaging devices. The steps for gathering data from Vesnet are as follows:

- Data is requested by us, with a clear specification of what is demanded.
- Message is read by an IT employee in Vesnet.
- The IT employee finds the required data.

- The IT employee then anonymises the requested data to ensure no information is leaked.
- The IT employee uploads the data into their servers, encrypting it.
- The IT employee then comes in contact with us, providing the link and how to decrypt the data
- The data gathered is then used for our project.

As seen, several measures are taken to ensure data privacy. Data-gathering ensures no confidential or sensitive information about anonymous patients is shared. Furthermore, the encryption ensures that no information is leaked to the public or can be read even if found.

It was decided to have more patients with cancer in the data than patients that are cancer free. For each patient, having approximately 200 DICOM images, cancer is only found on approximately $\frac{1}{6}$ of all images per patient. This is because tumours tend to be only at a single spot and only expand through a few frames due to their size. As such, most patients suffering from cancer have more non-cancerous frames than cancerous frames, causing discrepancies in the evenness of the data. In total, there is data of over 200 patients who are ill with carcinoma and over 20 patients who are not ill with any condition.

4.2 Treatment

We treat the data provided with the utmost care. In particular:

- When working with data, the data is decrypted. Once work with the data is finished, data is encrypted once more.
- Data is never uploaded to the World Wide Web.
- Data is only treated using local machines and software.

Data received comes in two parts:

- DICOM images of a patient.
- Doctor's report on whether the patient is diagnosed with cancer or other illnesses.

The DICOM images contain layered images of the patient's lungs in DICOM format. This format is then converted to PNGs, with the attempt to lose as little information as possible. Several different mathematical conversions were tested, whose results can be found in figures 11 9 12 10 below.

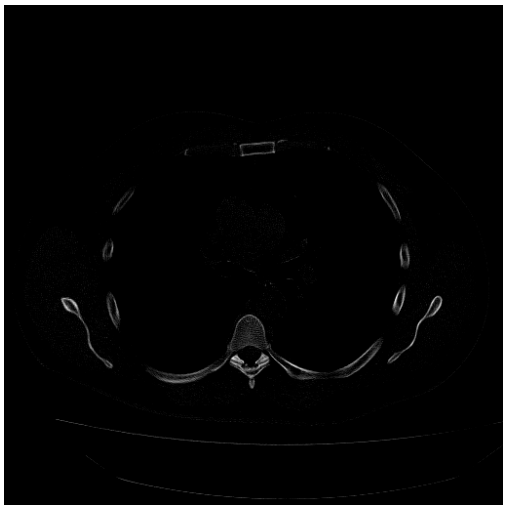


Figure 9: Skimage

Figure 10: CV2 Normalise

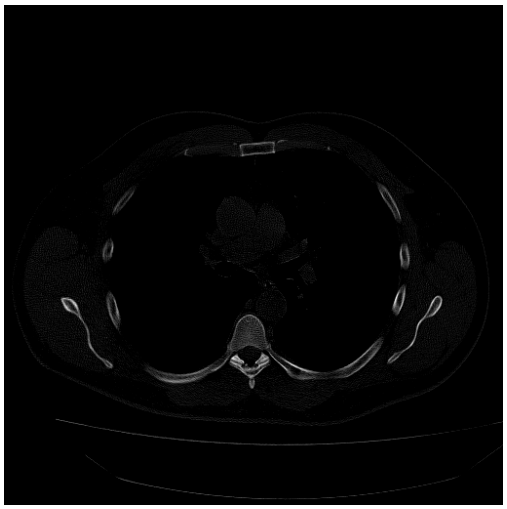


Figure 11: Values divided by max

Figure 12: Values minus min divided by mean

Through observation, it was decided that figure 12 with normalisation had the best output, being the most accurate to the original DICOM image. The formula used for this operation is precisely, for each pixel array, we replace the existing I value as follows:

$$I \leftarrow \frac{I - \min(I)}{\max(I) - \min(I)} \cdot 255 \tag{2}$$

The Python script for conversion, including formulae for the alternative methods that were tested, can be found in Appendix B.0.

Once converted, we manually check through each patient's CT scans in an attempt to find the cancer tumours if the patient is deemed to be ill through a doctor's report. Once found, the specific frames at which the tumour can be seen are recorded and saved. This helps us classify our data to ensure supervised learning.

The data is then split into three sets, serving different purposes:

- Training set - used for training the neural network, making it learn patterns in the data.
- Validation set - used to tune hyper-parameters and evaluate the model during the process of training.
- Test set - used to assess the final performance of the neural network by measuring its response to unseen data.

Considering that we have approximately 220 patients, with approximately 20 healthy patients, The split is determined as follows:

- Training set will consist of 5 healthy patients, 150 carcinoma patients
- Validation set will consist of 2 healthy patients, 20 carcinoma patients
- The test set will consist of 2 healthy patients and 40 carcinoma patients.

We note that the CNN will not have information on whether the data is from a single patient or multiple patients. Instead, it will determine whether each frame is prevalent with carcinoma. Due to the nature of carcinoma, i.e., random locations, it is likely that there will be a sufficient number of patients with healthy frames within every single possible location when considered all together. As such, all of the data will be mixed in a single folder together rather than differentiating each patient's results. In ConvLSTM, we believe that due to similar principles, current data may prove to be sufficient. The method and ability to detect whether a single patient has carcinoma will be in section 5.1, Implementation of the CNN, and 5.2, Implementation of the RNN.

5 Implementation

5.1 CNN

5.1.1 Literature Review

Much research has been done on classifying lung cancer in CT scans. One of these researches, conducted by Al Yasriy et al., is the most relevant to our project (Al-Yasriy, AL-Husieny, Mohsen, Khalil, & Hassan, 2020). Al Yasriy et al. have utilised CNN architecture under the name AlexNet. AlexNet is a CNN with over 9000 neurons, consisting of eight layers. The dataset of Al Yasriy et al. was collected using multiple Iraqi hospitals, with over 1100 slices and 100 patients. The patients were categorised into "normal," i.e., no tumour; "malignant," i.e., cancerous tumour; and "non-malignant," i.e., non-cancerous tumour. The proposed model of Al Yasriy et al. gives high accuracy ups to 93.548%, 95.714% for sensitivity, and 95% for specificity(Al-Yasriy et al., 2020).

The data used by Al Yasriy et al. was split into 70% training phase and 30% testing phase (Al-Yasriy et al., 2020). Approximately 100 epochs were done for testing. One can assume that an epoch correlates to a single patient (Al-Yasriy et al., 2020). Un-

fortunately, the conclusive reports and analysis of Al Yasriy et al. mainly focused on the correct classification of malignant vs. non-malignant tumours, meaning that it is inconclusive whether their model was underperforming within differentiating "normal" patients compared to "malignant" patients. In other words, it is not known what is the root of their design error.

Another approach to tackling the lung cancer classification problem is using another famous CNN architecture named ResNet. Nakrani et al. used image segmentation, where only parts of an image remain (Nakrani, Sable, & Shinde, 2020). In other words, data is processed so that only the inner lung is present in each image. One thing to notice is that Nakrani et al.'s research had significantly higher pooling and convolution windows than that of Al Yasriy et al.'s. Furthermore, (Nakrani et al., 2020)'s data consisted of 1100 patients, significantly larger than that of Al Yasriy et al.'s. Moreover, their window sizes for convolution were significantly bigger, with the smallest layer being 28×28, with approximately 40 epochs. Their accuracy had a median of 85.4%.

Several other methods, including training a CNN using 3D-based approaches, exist, which would allow mapping a single patient to a single image as done by several researchers (Alakwaa, Nassef, & Badr, 2017) (Ahmed, Parvin, Haque, & Uddin, 2020). Research suggests that doing analysis with 3D-based approaches is prone to less error and, therefore, higher accuracy (Yu et al., 2020). However, due to the nature of our data, a 3D-based approach is not possible, as each patient would be classified as a single image, implying that we would lack healthy patients.

In this dissertation, we primarily focus on "normal" patients and "malignant" patients. Our ability to obtain "non-malignant" patients is limited due to the nature of how data is collected. However, it appears that it is possible to achieve accurate results even with a low data set, as with Al Yasriy et al.. We note that Al Yasriy et al.'s accuracy achievement

may be related to their small windows for pooling and convolution, along with the high number of epochs allowing them to adjust weights more.

5.1.2 Design

We utilise PyTorch for the design of our CNN. In particular, PyTorch is a package in Python that includes multiple deep-learning functions and methodologies, providing us with an easy-to-use framework to design the CNN we have in mind. For our design and implementation, we found that the relevant packages can be found in the code below.

```
import os
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader, Dataset
import torchvision.transforms as transforms
import pandas as pd
from sklearn.model_selection import train_test_split
from PIL import Image
```

In the code above, $torch$ provides the fundamental code to create neural networks (PyTorch) and $torch.optim$ provides us with ready-to-use optimisation algorithms for our CNN. Meanwhile, $torch.transform$ provides us with a framework for pre-processing image data to be compatible with other parts of $torch$, e.g. feeding it to a tensor. Other packages, such as $pandas$ and os, provide us with methods to manipulate data and access our data, respectively. We want to stress the importance of os, as our data is divided by patient ID and their respective frames. We also have a CSV file sorted with patient data and image frame, with a label classifying 0 for no cancer and 1 for the true presence of cancer.

We now discuss the design taken in implementing the $DataLoader$ class, a class which

loads CSV and image data and ensures that it can be fed into the neural network. We note that in this architecture, we pre-load all the data for faster learning. We load images with their associated labels using PIL, the Python image library and the CSV data. Then, the images are loaded into tensors. After loading the data, we use the $scikit-learn$ package to split the data into validation, training and testing. As it stands, 70% of data is used for training, 10% is used for validation, and 20% is used for testing.

We note that all training is done on an RTX 4080 GPU with 16 GB of vRAM. We will now discuss the structure of the neural network, used functions, and optimisers. However, before we begin such a discussion, we want to note the class imbalance in the data: approximately 20% of data contains malignant tumours, whereas 20% of the images are clean from malignancy. As such, when designing the final architecture, this was considered.

AlexNet inspires our designed architecture. The number of pooling and convolution layers follows a similar style. Our literature review inspired this choice, as we noted that Al Yasriy et al.'s utilisation of AlexNet was successful. However, we would like to note that AlexNet is a more complex CNN designed with multiple classifiers in mind for classification. In comparison, our task involves only a binary classifier. Furthermore, the class of inputs of original AlexNet contained 3 channel R, G, B images, whereas our data is grayscale, therefore incorporating only a single channel. As such, the number of layers and neurons has been significantly decreased compared to the original architecture of AlexNet.

To be precise, our designed CNN contains 5122 connected neurons in the fully connected layer. The input layer contains 4608 neurons we obtained after flattening following convolution and pooling. The hidden layer contains a total of 512 neurons. The number 512 was considered due to the dimensions of our image. The final output layer contains

a total of 2 neurons, and this number was chosen to correspond to the number of classifications we have in our task (binary).

AlexNet has been designed to use max pooling. However, our architecture utilises average pooling to capture pixel contrast data more accurately. Furthermore, due to the lessened complexity of our task compared to AlexNet, we have adjusted the kernel size to be smaller, with our structure of CNN replicating the close end of AlexNet following the first 2 convolution layer and pooling layers. We have also lessened the number of convolution layers from 5 to 3 to capture data more precisely, adjusting to our task's simplicity in comparison to AlexNet's. Lastly, we have designed, similar to AlexNet, each of our pooling operations' kernel size s to be equivalent to the stride to ensure a reduction in dimensions for the fully connected input layer. However, in our convolution layers, we have decided to keep the padding and stride similar to AlexNet's, both values being assigned to 1.

Lastly, we discuss the choice of the loss function, activation function and the inclusion of an optimiser in our architecture. In AlexNet, the loss function is defined as cross-entropy loss. In particular, we can define cross-entropy loss $\mathcal{L}_{\text{CE}}$ as follows:

$$\mathcal{L}_{\text{CE}} = \sum_i y_i \log p_i$$

where y_i is the ground truth label from our CSV, and p_i is the CNN output, and i denotes the class 0 or 1. As we have observed, Cross entropy is popular and used by most modern efficient architectures, such as AlexNet. On the other hand, we have defined the optimiser function as Adam's optimiser, which is different from that of AlexNet. The reason for choosing a different optimiser is to ensure that the CNN learns despite the bias in data. However, the optimiser has other advantages, such as an optimised learning rate. Adam's optimiser is defined and functions as follows:

1. Compute gradient:

$$g_t = \nabla f(\theta_{t-1})$$

2. Update biased first moment estimate:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

3. Update biased second raw moment estimate:

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

4. Compute bias-corrected first moment estimate:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

5. Compute bias-corrected second moment estimate:

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

6. Finally, update the parameters:

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

θ_t is the updated parameter vector at iteration, θ_{t-1} is the parameter vector at the previous iteration, α is the learning rate, β_1 and β_2 are the exponential decay rates for the moment estimates, ϵ is a small constant to prevent division by zero, $\hat{m}_t$ is the bias-corrected first moment estimate, $\hat{v}_t$ is the bias-corrected second moment estimate. To further optimise our existence of data disparity, we have also assigned the frames including malignant

tumour to be worth 3 times (through consideration of distribution to be approximately $1:4$ ratio) than clean frames. For computation optimisation reasons, we include ReLU to be our activation function, which has a simple definition of

$$\text{ReLU}(x) = \max(0, x)$$

Using the compiled information, we can picture our final architecture of the CNN in figures 13 and y below, describing the fully connected layer and the convolution process respectively:

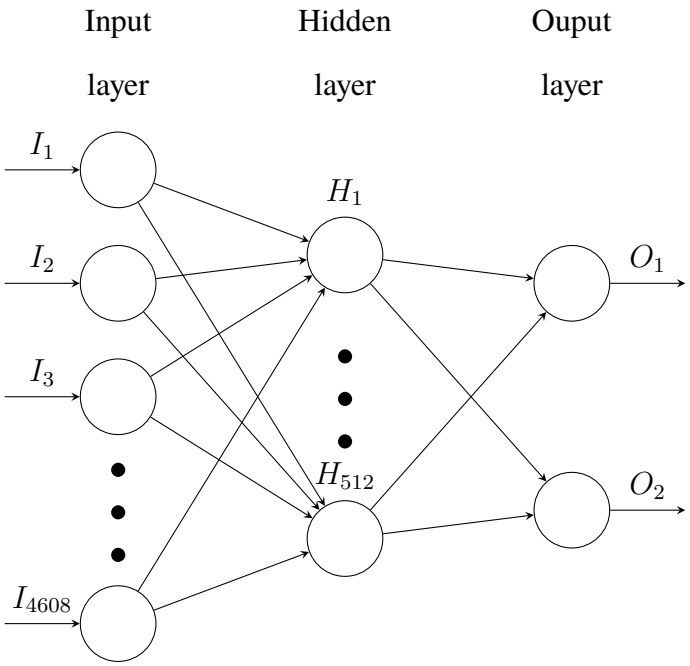


Figure 13: Fully connected layer architecture

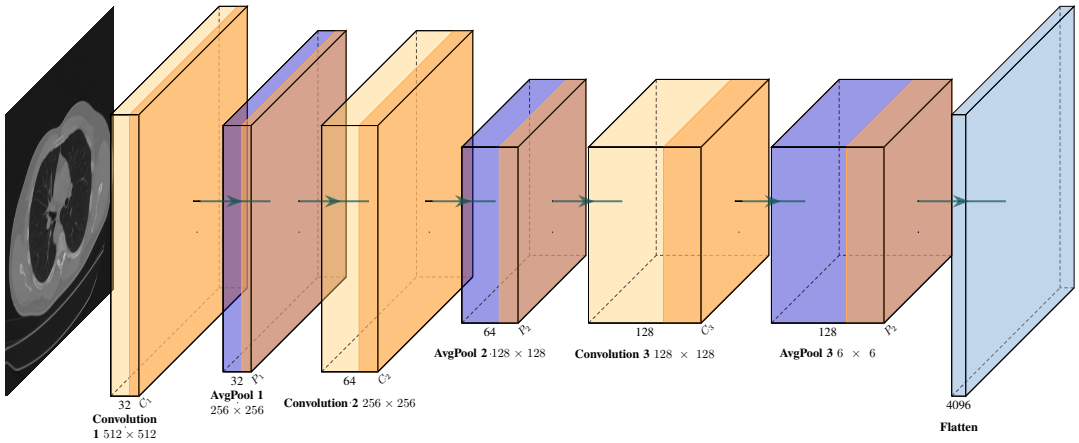


Figure 14: Convolution and pooling layers architecture

5.2 ConvLSTM

5.2.1 Literature Review

The research on ConvLSTM's effectiveness in medical imaging, specifically CT scan images of lungs, is minimal. This is a result of the fact that ConvLSTM is a new state-of-the-art architecture that was published in 2015. Ganesh et al. have used a highly modified architecture of the ConvLSTM that is hybrid with several other artificial intelligence methods such as Q-learning and the Gray Wolf Optimization algorithm (Ganesh & Nachimuthu, 2023). However, their accuracy was 92%, comparable to those of 3D CNNs (Ganesh & Nachimuthu, 2023). Their dataset for lung cancer patients comprised approximately 180 people (Ganesh & Nachimuthu, 2023).

Our main focus is the paper published by Mhaske et al. (Mhaske, Rajeswari, & Tekade, 2019), which primarily obtains results using CNN-LSTM. We note the important difference between CNN-LSTM and ConvLSTM: ConvLSTM incorporates convolutional operations within the LSTM architecture, whereas CNN-LSTM focuses on extracting features using a CNN and feeding them in an LSTM. However, because both architectures have a glaring similarity, it was deemed that the analysis of this paper is useful

to us. We note that the images were segmented in research conducted by Mhaske et al. (Mhaske et al., 2019). The dataset of Mhaske et al. consisted of 1010 patients, with a final result of accuracy 97% (Mhaske et al., 2019). However, we are cautious of the methodology used in splitting data for verification and obtaining results, as it was not disclaimed.

Lastly, one of the closest applications of vanilla ConvLSTM in medicine we have found was conducted by Kang et al. on the detection of renal tumours inside an abdomen using CT images (Kang, Zhou, Huang, & Han, 2022) which consists of 5 ConvLSTM cells and post-processing. However, the paper's primary focus is to replace a 3D CNN with a ConvLSTM as an encoder to alleviate the complexity of training and classifying (Kang et al., 2022). Unfortunately, some pre-processing of their data was still done using 3D CNNs, making it difficult to extract the true effectiveness of a vanilla ConvLSTM in a classification task.

To conclude, much research has combined various forms of LSTMs to achieve high results. We could not find any conclusive evidence of the effectiveness of vanilla ConvLSTMs due to the architecture's recency. However, we aim to fill the gap in this paper as we research artificial intelligence methods in detecting lung cancer tumours of patients scanned with computer tomography.

5.2.2 Design

The design for creating an efficient and accurate ConvLSTM proved extremely challenging due to the need for more literature data on a design suited for our task. As such, our design attempts to implement several different techniques to overcome the challenges within the architecture of ConvLSTM in our task.

The process of data preparation for ConvLSTM shares some basic aspects with that of CNN. However, we made significant modifications to the original method to cater to the sequential data requirements of ConvLSTM. Specifically, we introduced the $Subset$ package and implemented a fixed iteration through the data file. These changes ensure that data is not shuffled during extraction, thereby maintaining the fixed sequential and sorted structure of our CSV sheet, a crucial requirement for our ConvLSTM implementation.

The implementation of our ConvLSTM cell follows the standard theory previously discussed in Section 3.4 Background RNNs. Specifically, we have defined a ConvLSTM class to be able to take input of kernel size, number of cells, and the number dimensions along with bias to ensure more balanced learning. The cell class is extended by the ConvLSTM class, which implements extra features such as a number of layers and a number of classes. By implementing both, we can manipulate the hyperparameters of the number of cells in our structure.

Similar to the CNN, we have ensured GPU usage support in our ConvLSTM implementation by dynamically loading the images instead of pre-loading. The training, metric calculations, and evaluation processes are analogous to those of the CNN. We initially chose the Adam optimizer due to its efficiency in training when data is skewed to a certain class. However, we found that using only the Adam optimizer was not effective in ensuring balanced training of our architecture. To address this, we implemented additional measures to improve the training results, demonstrating our commitment to optimizing the ConvLSTM architecture for our task.

Particularly, we have implemented definitions to control the learning rate l which is set to 0.001 by default to adjust depending on the number of cancer images found per batch of learning. Through hyper-parameter tuning and managing the loss during iteration for

balance, we have found that the numbers in the code snippet below are reasonable:

```
    if cancer_ratio > 0.6:
        lr = lr * 3
    elif cancer_ratio > 0.5:
        lr = lr * 1.5
    elif cancer_ratio > 0.3:
        lr = lr * 0.8
    elif cancer_ratio > 0.08:
        lr = lr * 0.6
    else:
        lr = lr * 0.5
```

We have further added extra adjustments to ensure more balanced learning. Specifically, we have also implemented drop logic. The dropping logic undersamples the number of entirely healthy batches to provide more balanced learning. The batches are dropped probabilistically to provide different learning per epoch. The code for falling in the training loop is as follows:

```
    if (labels == 0).all():
                # Decide to skip this batch with a certain probability
                if random.random() < drop_probability:
                    #print(f"Skipping batch {batch_idx+1} due to being
                        fully clean.")
                    continue # Skip this iteration, thus not training on
                        this batch
            adjust_learning_rate(optimizer, epoch, base_lr=0.001,
                batch_labels=labels)
```

Using a similar methodology of tracking the loss after each batch and distribution of batches depending on the number of positive cases, we have determined that a suitable drop probability is 75% for this particular configuration.

To further ensure balanced learning, we have also modified the architecture structure to be complex. In particular, for our batch size $B = 8$ and the limitations of GPU, we have an input dimension of 1 matching the grayscale channelling. The hidden dimension is 8, and the output dimension is 2 for binary classification. These values were set in accordance with the limit of our vRAM. We further use average pooling to extract each feature across all time steps.

By utilising all the features we have discussed, we have attempted to create a balanced learning ConvLSTM for our task. We once more note that we have tested multiple values.

6 Results

6.1 CNN

6.1.1 Data

In this section we will be displaying the results obtained from training, along with the results of our chosen metrics that we will use for analysis. We note that when we display the obtained information, our number of epochs $E = 30$ and batch size $B = 16$. We have set $E = 30$ as we have noticed that $E > 30$ begins to significantly overfit. We begin with displaying figure 15, which shows the relationship between number of epochs and computed loss of training data and validation data:

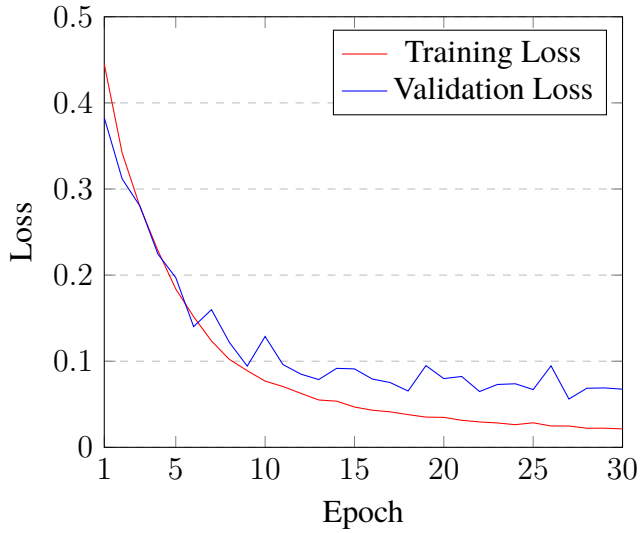


Figure 15: CNN: Epoch v Loss

Furthermore, we obtain data to find the relationship between accuracy and epoch using analogous method, as such we plot figure 16:

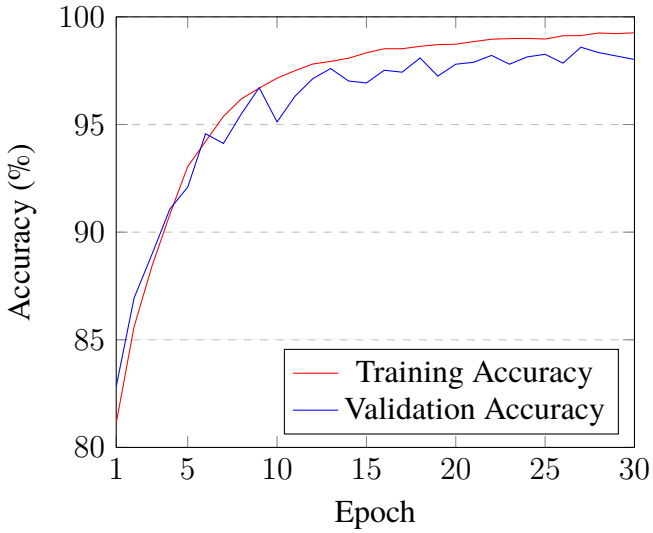


Figure 16: CNN: Epoch v Accuracy

Considering the fact that we have also tracked the number of false positives and false negatives per epoch on validation data, it is useful to plot the relationship on epoch vs false positives and false negatives to enhance our understanding in figure 17:

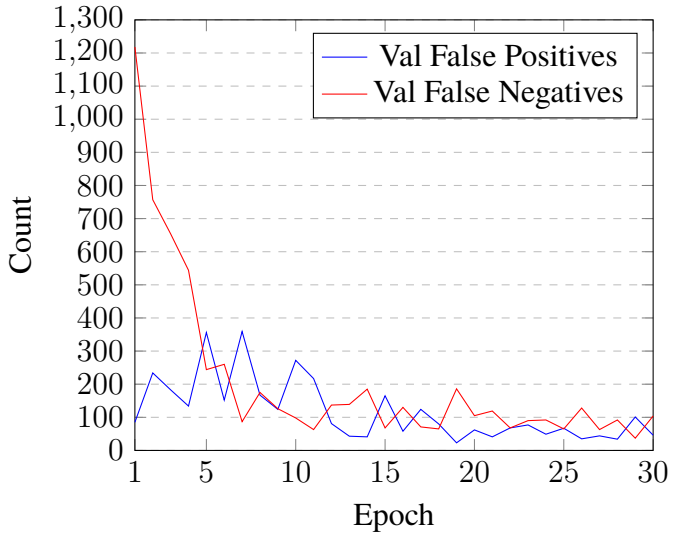


Figure 17: CNN: Epoch v FP and FN

Lastly, let us display the results we got for test data in the confusion matrix below:

	Positive	**Negative**
True	1851	8032
False	110	69

Table 1: CNN: Test data confusion matrix

6.1.2 Analysis

In this section we will consider analysing our result data by observing the obtained results. We will analyse sequentially, similar to how data is displayed in the previous section.

Let us begin with analysing figure 15, which considers the relationship between the epoch number x and the loss l. As observed from the data, the decrease in loss over the epoch suggests that our model is learning well. We noticed that the validation loss graph is less smooth than the training loss graph. The smoothness of the training loss graph is likely related to the fact that our model learns from it; therefore, it slowly converges to

a value through calculations. However, in validation, there is likely information that the model does not know, and thus, during each epoch, it may forget important information that relates to validity set, causing bumps. Furthermore, we notice that the training loss and validation loss begin to deviate from each other at approximately the 10th epoch, suggesting overfitting. Despite the overfit, the validation data seems to improve gradually, though in increasingly smaller amounts.

Let us now consider figure 16, which considers the relationship between epoch number x and the accuracy a. Similar to the previous graph, we notice the steep learning curve during the initial launch of the data, with a decreasing rate of change, suggesting that the model is converging to a maximum value. Similarly, this figure indicates that overfitting occurs at approximately epoch 10, suggesting the reliability of our analysis. The perturbations in validation accuracy follow equally to that of the previous figure, further suggesting our hypothesis of forgetting important information in the validation set.

Using figure 17, we can further observe new information on the reason for possible perturbations in loss and accuracy. In particular, from epochs 1-10, we can observe a steep decrease in false negatives, suggesting that the model focuses on learning to identify cancer tumours correctly. As it focuses on identifying cancer tumours, we can see that it slightly loses its ability to identify clean image frames by increasing false positives. However, once both counts are somewhat more balanced, we notice a pattern that false positives and false negatives are inversely correlated. This balance between false positives and false negatives is a key aspect of the model's performance, as it indicates a high level of precision in its predictions. This further supports the initial hypothesis we postulated from the first figure.

Lastly, we consider the confusion matrix of our final test data. We calculate our metrics

as discussed previously to be the following:

$$\text{Precision} = 96.3\%$$

$$\text{Recall} = 94.4\%$$

$$\text{F1} = 95.3\%$$

$$\text{TNR} = 98.7\%$$

Our TNR demonstrates a high specificity with a TNR of approximately 98.7% of actual negative cases as correctly identifying negative. Additionally, the precision of 96.3% signifies that most of our model's positive case identifications were correct, while a recall of approximately 94.4% of actual positive cases was correctly classified. This balance between precision and recall is reflected in the F1 score, roughly 95.3%, signifying a robust overall performance.

6.2 ConvLSTM

6.2.1 Data

We note that despite any parameter changes, our results have not changed. As such, for this section, we will discuss a model that contains the highest overall accuracy. The implications of results not changing is discussed in the next section. The tuned value parameters for this model are as follows:

- Probability of dropping clean batches = 75%
- Learning rates l, factors of 1.8, 1.6, 0.9, 0.6 from highest cancer ratio to bottom, with base learning rate $l = 0.001$
- Number of cells $= 4$
- Loss function $\mathcal{L}$ Adam

- Batch size $B = 24$
- Order of data input = unshuffled and sequential

We will begin by plotting Epoch with respect to Loss in figure 18. We note that loss is constant throughout, and that training data loss compared to evaluation data loss share staggeringly similar values. We furthermore notice that loss is high throughout, and furthermore, due to overfitting or underfitting, varies highly with each epoch.

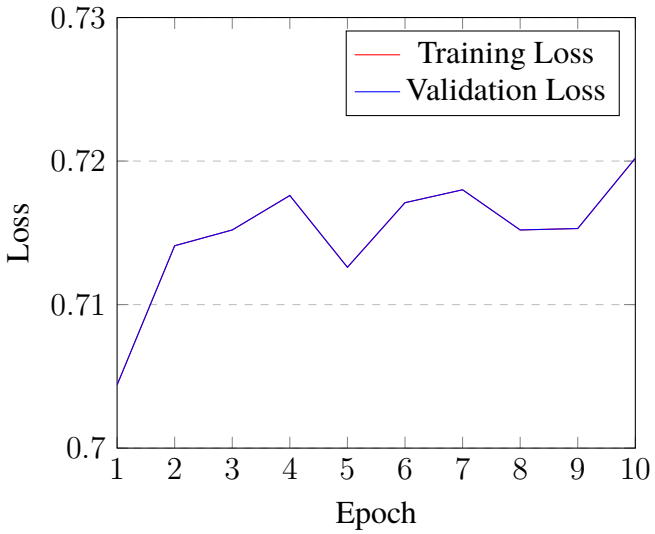


Figure 18: ConvLSTM: Epoch v Loss

Furthermore, the accuracy of our ConvLSTM architecture appears to be consistent, classifying all instances as negative, causing a fixed accuracy rate. We note the difference in validation accuracy to training accuracy, suggesting overfitting. This relationship is demonstrated in figure 19 below.

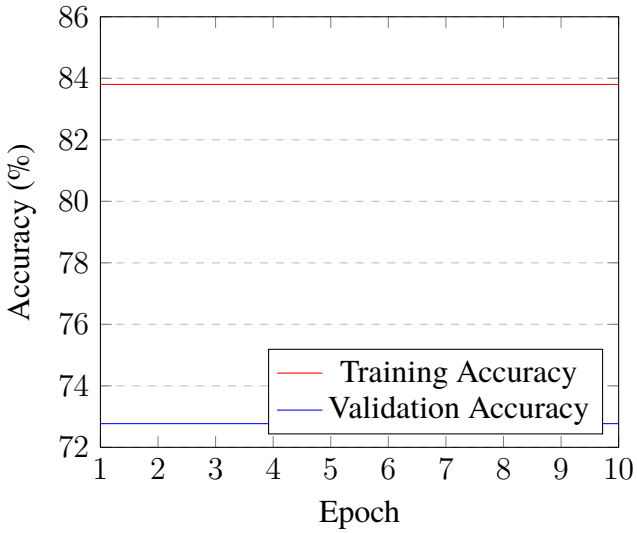


Figure 19: ConvLSTM: Epoch v Accuracy

Lastly, let us consider the number of false negatives and false positives in our validation data across epochs, demonstrated in figure 20:

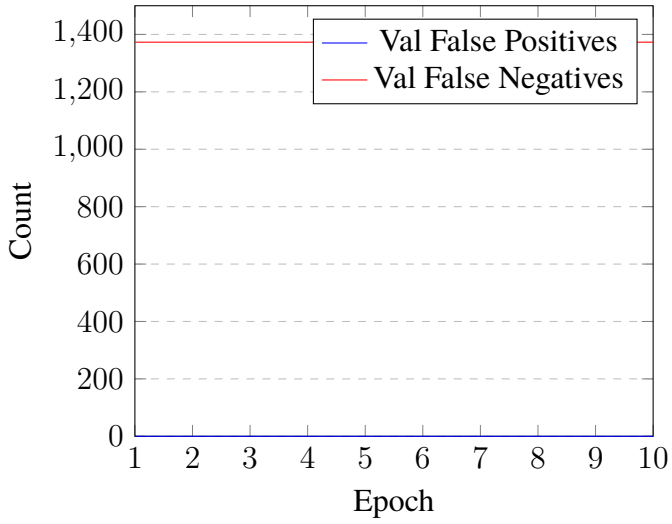


Figure 20: ConvLSTM: Epoch v FP and FN

Indeed, from this figure, the confusion matrix confirms our concerns and results as follows:

	Positive	**Negative**
True	0	7438
False	2649	0

Table 2: ConvLSTM: Test data confusion matrix

6.2.2 Analysis

We would like to begin this section be reiterating the fact that theoretically, the implementation of ConvLSTM has the basis of capturing temporal data, which theoretically should enable the prediction of cancer by analysing the development of tumour per frame. However, the analysis suggests otherwise, and in this section, we will explore why this might be the case by analysing our results in detail along with the steps we have taken to achieve them.

Figure 18 suggests that our data is either underfitting or overfitting by having exceptionally loss of approximately 0.715 across all epochs. Furthermore, we notice that validation loss and training loss have exceptionally similar values for each epoch, suggesting that the model may not be complex enough to uncover patterns for classification. Furthermore, figure 19 demonstrates a constant accuracy for both tests, suggesting that the architecture is unable to learn the required features for classification. Figure 20 confirms that the architecture is unable to learn further as it learns to classify every image as negative. We note, however, that the overall accuracy is deceiving, which can be demonstrated by our other metrics.

Specifically, let us compute the F1 score, recall, precision and TNR to demonstrate that

the accuracy is deceiving. The results are as follows:

$$\text{TNR} = 1.0$$

$$\text{Recall} = 0$$

$$\text{Precision} = \text{UNDEFINED}$$

$$\text{F1} = \text{UNDEFINED}$$

Therefore, we can deem that the current architecture is unable to capture any useful data to be used. However, we can reason the results considering our experiments during implementation and postulate the reason why they are what we have recorded.

On top of metric scores, we would like to reiterate the fact several steps were taken to tune the design. As mentioned previously, we have implemented additional measures to handle the data imbalance and overfitting. In the tuning stage, the following parameters were tuned individually to asses their impact, and in some cases, merged together. Our observation and tests consist of over 200 manual tests by adjusting the following parameters:

- Probability of dropping clean batches (undersampling)
- Learning rates l, including with and without the adjustment to the nature of the batch being used
- Number of cells
- Loss function $\mathcal{L}$
- Batch size B
- Order of data input
- Sample size

- Shuffling

Using the results we obtained, we concluded that the vanilla ConvLSTM model we implemented is not an effective method of analysing CT scans of the lungs to detect cancer. Furthermore, despite adjusting different hyper-parameters, we have deemed that even with a balanced dataset or smaller dataset, the ConvLSTM fails the ability to capture significant differences in the data despite increasing the number of cells. We have further decreased the batch size to 2 to explore this hypothesis, with the same outcome. The ConvLSTM fails to capture data differences effectively and always converges towards classifying all samples to specific classes no matter how many extra measures we have taken. However, memory constraints limited our architecture to a maximum of 10 total cells. Despite the changes in the number of cells, with a minimum of 3 to a maximum of 10, we have yet to notice any difference in the architecture learning ability, with the classification always overfitting or underfitting.

As such, we hypothesise and deem that the vanilla ConvLSTM as an architecture cannot classify images as we theorised, no matter the complexity, data balance or learning rates we have adjusted due to the structure of the cell's ability to capture data, making it overly sensitive to the smallest changes we have administered. Therefore, ConvLSTM is not an architecture effectively suited to solve our problem. The lack of existing literature also means we cannot compare and confirm our conclusion with research, however, this may be because most researchers have theorised that ConvLSTM architecture is unsuited for this task.

7 Conclusion and Evaluation

Our conclusion and evaluation of this dissertation is that CNNs are more likely the most suited architecture for our task problem. By exploring these different architectures, we have shown the high accuracy of 98% and the high metric scores of our CNN archi-

tecture. This accuracy is higher than our reviewed literatures. Furthermore, the ConvLSTM architecture sounded theoretically effective and could produce good results in our classification problem. Conversely, the vanilla ConvLSTM architecture failed to capture frame differences correctly and effectively, causing overfitting or underfitting. Despite our attempted corrections and modifications in the parameters to obtain results whilst keeping the architecture vanilla, through our results, we concluded that the vanilla ConvLSTM's are unsuitable for our task. This is because our taks is inherently possible more complex than ConvLSTM's capabilities. Furthermore, ConvLSTMs require a high amount of video memomry compared to CNNs, deeming the architecture inefficient in clinical practices. As such, current state-of-the-art and our custom model CNN should be the primary method of finding malignant tumours in patients.

7.1 Achievements

We succeeded in training our neural networks despite high data disparity. Training neural networks in the presence of such data disparity is a significant human achievement of our dissertation, enabled through effective training techniques like the Adam optimizer algorithm. We demonstrate that the negative effect of data dissimilarity can be reduced with solid optimization strategies applied so that the neural network can learn representative patterns and features across different datasets. This success thus underscores the critical relevance that methodological innovation holds in overcoming the inherent challenges posed to medical image analysis, hence contributing to the advancement of computational methodologies for better diagnostic precision.

The accuracy level of our CNN Architecture is truly an outstanding achievement, an excellent one, concerning the ability of our designed CNN model to achieve a 98% accuracy level. Achieving this level of accuracy demonstrates that the model put in place is practical and robust in discriminating cancerous from non-cancerous lung nodules. It

can further be observed that the model achieved good True Negative Rate (TNR) and F1 scores, which confirmed the diagnostic ability of the model. This further substantiates the effectiveness of our CNN architecture in lung cancer detection and induces confidence that it might be possible to enable accurate and reliable lung cancer detection, adding to the further development of computational methodologies of medical image analysis.

Furthermore, the successful fulfilment of our project management objectives is a testament to our efficient execution and completion of the dissertation. This involved a systematic performance comparison between two neural architectures, achieving a target accuracy threshold of over 95% in a single model, and successfully implementing a DICOM to PNG conversion pipeline. Our adherence to good project management practices, such as the development and execution of research towards predefined goals and milestones, underscores our planning, execution, and evaluation rigour. This success reflects our structured project management practices in achieving research objectives and delivering tangible outcomes.

The inherent difficulty must be realized in fine-tuning the convolutional LSTM model for our specific task. Even if this is our best effort, hindrances are telling us how far possible it is for us to achieve optimal performance within this context. However, this journey has been immense in its value regarding the lessons learned. The first goes to the unbalanced data and learning many small details during medical image analysis. As we take these challenges head-on, our domain knowledge has grown over time, and we have become proficient in preprocessing data, model development, and evaluation exercises. All in all, while the ultimate goal of achieving high accuracy with ConvLSTM might be elusive, the focus placed on learning and adaptation turned out to be an enriching exercise that gave more impetus to our growth as researchers in medical imaging.

7.2 Limitations

The data from a single model of the CT scan device is limited in such a way that the use of data from a single model of the CT scan device will be a significant source of bias and further introduces potential variability into the dataset. CT scans' models will vary, and the images will have noise, diverse artefacts that will limit model generalisation across different imaging devices. This limitation restricts the generalisation of the trained model's applicability to any data collected from the used CT scan model, thus reducing the effectiveness in real-world applications where data from different sources are being met. Moreover, total reliance on one CT scan model would mean losing out on critical differences in image characteristics that may be present in images taken by other models and loss of opportunities for better accuracy and improved diagnostic performance.

We also have low sample size compared to most literature works: this, therefore, means our research offers a huge limitation regarding the robustness and generalisability of our findings. These "big" and "diverse" datasets are of paramount importance, in particular, deep learning models used in tasks such as medical imaging, are to derive some meaningful understanding of the intricate patterns and features that may indicate pathological conditions. When the sample size is limited, the model may perform poorly in real life scenarios; hence, it fails to capture all the variations in lung cancer imaging, and thus the predictions are unreliable. On the other hand, using small sample sizes only increases the risk of overfitting, where the model learns how to over-memorize the training data instead of generalising to unseen data. The result is that the performance metrics become inflated and, therefore, a false sense of efficacy is engendered.

The low accuracy observed during classification by the designed ConvLSTM architecture for lung cancer is a critical issue that needs to be addressed. This inefficacy may

be due to sub-optimal network architecture, poor training data at scale, and poorly optimized strategies. It's crucial to address these underlying reasons for the low accuracy to boost the model's performance and build confidence in the diagnoses it can make. This will facilitate the adoption of the model as a new and valuable tool in lung cancer detection.

Furthermore, our problem is oversimplified. That is, it is oversimplified through the binarisation of cancer or non-cancer, the complexity and heterogeneity of lung diseases get lost. This model does not take into account benign tumours or, in fact, any other non-malignant conditions of the lung; this, therefore, serves to limit its diagnostic capability and clinical applicability. Misclassifying the benign nodules for malignant or vice versa has far-reaching consequences on the management of the patients such as unnecessary invasive procedures are undertaken. More importantly, this indicates that classification schemes need to be extended further, considering the ample spectrum of pulmonary disorders met in clinical practice, with insufficiency in distinguishing lung cancer from some other lung pathologies.

Lastly, our computational resources restricted to the RTX 4080: the constraint of computational resources allows only RTX 4080 in the provided machines, a real impediment in researching and further developing other deep learning architectures. Models such as deep learning models on top of complex convolutional and recurrent neural networks usually require a lot of computation in the process of training and optimization. Moreover, the limitation of a single GPU would inhibit the opportunity to explore and train large-scale, more complex models that would be feasible in carrying out extensive exploration over much more advanced architectural techniques on lung cancer detection, which is expected to outperform in our analysed architectures. Therefore, this confines the scope in which the research can harness new advances in deep learning techniques and probably the failure to fully capitalise on the advancement of computational method-

ologies in pursuit of better diagnostic accuracies.

7.3 Future Work

7.3.1 Data Combination

One way to expand the existing work we have achieved in this dissertation is to combine our existing knowledge with other types of patient data. Specifically, this type of extension of work refers to the amalgamating of various forms of medical data that a patient may have with CT scans of the lungs. Such extension aims to increase the accuracy and reliability of our work in diagnosing a patient with cancer, as well as its exact location in the lungs. This section will discuss which data forms may help facilitate the extension.

An example of data combination that we could utilise is the fusion of multimodal imaging data. That is, in addition to CT scans of lungs, we could utilise the existence of other devices such as magnetic resonance imaging (MRI), positron emission tomography (PET), and X-rays. Using different images can provide information to our network on things such as tumour morphology, metabolism and tissue composition. Through the combination of these different imaging techniques, we would be able to extract more detailed and informative representations of frame data, facilitating more accurate diagnosis (Jintao Ren & Korreman, 2021).

However, it is not only imaging data that we can combine our CT scan images with. We can extract other insightful data from genetic markers, biomarker levels, patient demographics, and clinical history. All such data can provide valuable information in training a CNN and an LSTM. It may be possible to use clinical history as a part of LSTM to create a predictive function on the probability of cancer. Other data, such as genetic mutations, can suggest lung cancer susceptibility, and biomarkers can indicate tumour spread. All of this data can be used to improve the detection of cancer.

Lastly, it is possible to combine data with longitudinal studies. That is, by considering multiple time points of the patient's data, it is possible to track potential changes. It might be possible to gain insights into progression of cancer or other novel diseases such as tuberculosis. In fact, as mentioned, it might be possible to include the longitudinal data in a modified ConvLSTM model as we have utilised in this dissertation. As discussed previously in background, ConvLSTM's architecture allows to predict using temporal differences.

In summary, data combination offers a promising approach to improving lung cancer detection and patient diagnosis. By leveraging diverse sources of information gathered by different devices and histories, it is possible to create a more accurate, robust, and clinically relevant model for detecting and possibly treating lung cancer.

7.3.2 Different Data

It is furthermore possible to extend this project by including other various organs. That is, we have the potential to expand our dissertation project to include other organ-specific research. Exploring other organs would provide us a fresh and new perspective to the current focus on lung cancer imaging. Furthermore, by exploring the complexity of different organs and their specific tumor characteristics, we can expand our model's understanding of cancer and refine detection to other specific areas of the body, potentially through the use of transfer learning. In this section, we explore how organ-specific research can enhance our knowledge and contribute to the advancement of cancer imaging.

One way we could approach solving patterns for different carcinomas is to expand our knowledge on specific organs that can be scanned through CT. Such organs include, but are not limited to, the heart, liver, pancreas, bladder, kidneys, etc. We can learn how and where carcinoma tumours can grow by studying other organs individually. Furthermore,

we may gain insights into the nature of carcinoma. That is, we can investigate general and effective approaches for cancer detection in CT scans, which may be used clinically in the future. Investigating different cancers may also allow us to create a supermodel that can detect various cancers.

Furthermore, analogous yet different to the combination of data, we may also explore different carcinomas using different devices. In particular, it is known that different devices are more efficient at detecting cancer in various organs, such as x-rays being typically used in mammography (NHS, 2024). By utilising different machines to postulate conclusions on a patient, we might gain the ability to create more robust and accurate systems for detecting carcinoma within different parts of the body.

In conclusion, by studying different organs individually, we can deepen our understanding of the formation of carcinoma in other bodies, possibly allowing us to extend our learned knowledge in neural networks into different organs through transfer learning. Furthermore, it is possible to gain better insight into specific types of carcinoma using various devices, suggesting that it is possible to train better deep-learning models in cancer detection other than using CT scans as the central methodology for different organs.

7.3.3 Software Development

One of the most innovative extensions to this dissertation would be to create a fully-fledged software that utilises our current trained data to create a product that is to be used in the real world. Using software development, we would gain the ability to streamline the process of cancer image analysis in a doctor's report. By doing this, we hope to increase patient outcomes by reducing the time taken to analyse data and promote frequent cancer screening by making it efficient, accessible and inexpensive.

Specifically, the aim of this software, powered by advanced AI technology, would be to facilitate further training, assisting clinicians and radiologists. It would also enable preliminary tests on the possibility of cancer through screening, without the need for a clinician. The central hub of the software would serve as a repository for storing and managing imaging data, clinical records, and diagnostic reports. Moreover, integrating deep learning solutions would deploy suggestions and diagnostics for clinicians and radiologists in the diagnosis of a patient, with the final potential to create diagnostic reports on its own.

However, the design of such software is not easy and requires robust UI design. To increase the efficiency of the patient diagnosis as planned, the software must be intuitive and easy to use. In particular, our software development efforts would aim to enhance the user interface (UI) and user experience (UX) to ensure ease of use and accessibility for healthcare providers. Intuitive interfaces, interactive visualisation tools, and decision support features empower clinicians to interpret imaging data effectively and make informed decisions.

In summary, the creation of a fully-fledged software using our current trained data represents a groundbreaking extension to our dissertation work in cancer imaging analysis. By harnessing the power of advanced AI technology, this software would aim to revolutionise the process of cancer image analysis in clinical settings, ultimately improving patient outcomes and promoting frequent cancer screening.

7.3.4 Classification

Our project focused on classifying malignant tumours within lung computer tomography scans of patients. However, our approach cannot classify benign tumours, and it cannot classify any other possible lung diseases, such as COVID-19 or tuberculosis. This limitation underscores the need to expand our classification framework to encompass a

wider spectrum of conditions.

Expanding classification capabilities of our research presents us with an exciting opportunity. By developing classifiers that can differentiate between malignant, benign and other diseases in patient CT lung data, we will gain the ability to provide more comprehensive diagnoses to clinicians and radiologists. That is, we would be able to guide and suggest clinicians and radiologists on possible cases of lung abnormalities, inflammations, viruses, etc., of the patient they are diagnosing.

However, achieving broader classifications in our models would require challenging to obtain data. Not only would we have to be presented with lungs of several diseases at high amounts, but we would also be required to obtain doctor reports to ensure supervised learning of our model. Nevertheless, it is possible to achieve this extension if we consider sponsors. A partnership with a diagnostics centre, assuming their interest in such a project, would be extremely useful in achieving such an extension.

In conclusion, even though our project has successfully classified the difference between a malignant tumour frame and a clean frame with high accuracy, we cannot classify benign tumours or other lung diseases. As such, it is fundamental that we develop and extend our classifier to give it the ability to differentiate between malignancy and benignancy, along with other novel diseases such as COVID-19. Achieving such an extension, however, would require a variety of well-labelled data, which could be addressed through collaborations with diagnostic centres and sponsorships.

7.3.5 3D Architecture

Other than the 2D analysis of our data, 3D architectures exist that can act similarly to those of RNNs. In other words, 3D architecture, such as 3D-CNNs, can capture volumetric features of input data. As we have seen in Section, we can achieve a more accurate

classifying neural network for our problem by utilising volumetric features (Alakwaa et al., 2017).

Using 3D architectures opens new avenues and brings with it the potential to incorporate advanced techniques such as volumetric segmentation and reconstruction. By reconstructing 3D models of lung CT scans, we can generate detailed representations of tumour morphology and spatial distribution. This, in turn, can significantly enhance the accuracy of diagnosis when classifying and further aid treatment planning to clinicians and radiologists post-diagnosis, marking a significant step forward in our understanding and management of lung cancer.

However, it's essential to acknowledge the significant challenges associated with implementing 3D deep neural network architectures. One of the most daunting hurdles is the increased computational complexity, as the number of computations grows exponentially. Additionally, the architecture necessitates larger datasets due to the computations and details required to create a 3D model. This extension, while feasible, would be a complex undertaking given our current circumstances.

In conclusion, integrating 3D architectures like 3D CNNs gives us the potential to improve the accuracy of our lung cancer imaging classification. These architectures capture volumetric features similar to that of RNNs. Techniques such as volumetric segmentation enable detailed representations of lung CT scans, aiding diagnosis and treatment planning. However, challenges like increased computational complexity and the need for larger datasets arise. Despite these hurdles, exploring 3D architectures holds potential for future research in lung cancer imaging analysis, as demonstrated by Alakwaa et al.

References

Ahmed, T., Parvin, M. S., Haque, M. R., & Uddin, M. S. (2020). Lung cancer detection using ct image based on 3d convolutional neural network. *Journal of Computer and Communications*, *08*(03), 35–42. Retrieved from http://dx.doi.org/10.4236/jcc.2020.83004 doi: 10.4236/jcc.2020.83004

Alakwaa, W., Nassef, M., & Badr, A. (2017). Lung cancer detection and classification with 3d convolutional neural network (3d-cnn). *International Journal of Advanced Computer Science and Applications*, *8*(8).

Al-Yasriy, H. F., AL-Husieny, M. S., Mohsen, F. Y., Khalil, E. A., & Hassan, Z. S. (2020, nov). Diagnosis of lung cancer based on ct scans using cnn. *IOP Conference Series: Materials Science and Engineering*, *928*(2), 022035. Retrieved from https://dx.doi.org/10.1088/1757-899X/928/2/022035 doi: 10.1088/1757-899X/928/2/022035

Basodi, S., Ji, C., Zhang, H., & Pan, Y. (2020). Gradient amplification: An efficient way to train deep neural networks. *Big Data Mining and Analytics*, *3*(3), 196-207. doi: 10.26599/BDMA.2020.9020004

Cancer Research UK. (2022, May). Retrieved 8 October 2023, from https://www.cancerresearchuk.org/health-professional/cancer-statistics/mortality/common-cancers-compared#heading-Zero

Chen, W., Li, Z., Liu, C., & yi, a. (2021, 05). A deep learning model with conv-lstm networks for subway passenger congestion delay prediction. *Journal of Advanced Transportation*, *2021*, 1-10. doi: 10.1155/2021/6645214

Ganesh, S., & Nachimuthu, M. (2023, October). Improving cancer classification using deep reinforcement learning with convolutional lstm networks. *Revue d'Intelligence Artificielle*, *37*(5), 1367–1376. Retrieved from http://dx.doi.org/10.18280/ria.370530 doi: 10.18280/ria.370530

Gharraf, H. S., Mehana, S. M., & ElNagar, M. A. (2020, Sep 17). Role of ct in dif-

ferentiation between subtypes of lung cancer; is it possible? *The Egyptian Journal of Bronchology*, *14*(1), 28. Retrieved from https://doi.org/10.1186/s43168-020-00027-w doi: 10.1186/s43168-020-00027-w

Jintao Ren, J. N., Jesper Grau Eriksen, & Korreman, S. S. (2021). Comparing different ct, pet and mri multi-modality image combinations for deep learning-based head and neck tumor segmentation. *Acta Oncologica*, *60*(11), 1399–1406. Retrieved from https://doi.org/10.1080/0284186X.2021.1949034 (PMID: 34264157) doi: 10.1080/0284186X.2021.1949034

Kang, L., Zhou, Z., Huang, J., & Han, W. (2022). Renal tumors segmentation in abdomen ct images using 3d-cnn and convlstm. *Biomedical Signal Processing and Control*, *72*, 103334. Retrieved from https://www.sciencedirect.com/science/article/pii/S1746809421009319 doi: https://doi.org/10.1016/j.bspc.2021.103334

Mhaske, D., Rajeswari, K., & Tekade, R. (2019). Deep learning algorithm for classification and prediction of lung cancer using ct scan images. In *2019 5th international conference on computing, communication, control and automation (iccubea)* (p. 1-5). doi: 10.1109/ICCUBEA47591.2019.9128479

Nakrani, M. G., Sable, G. S., & Shinde, U. B. (2020). Resnet based lung nodules detection from computed tomography images. *Int J Innov Technol Exploring Eng*, 1711–4.

NHS. (2024, Sep). *What happens at your breast screening appointment.* Author. Retrieved from https://www.nhs.uk/conditions/breast-screening-mammogram/what-happens-at-your-breast-screening-appointment/

Nooreldeen, R., & Bach, H. (2021, Aug). Current and future development in lung cancer diagnosis. *International Journal of Molecular Sciences*, *22*(16), 8661. doi: 10.3390/ijms22168661

Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., kin Wong, W., & chun Woo, W. (2015). *Convolutional lstm network: A machine learning approach for precipitation now-*

casting.

World Health Organization. (2023, Jun). World Health Organization. Retrieved 8 October 2023, from https://www.who.int/news-room/fact-sheets/detail/lung-cancer

Yu, J., Yang, B., Wang, J., Leader, J. K., Wilson, D. O., & Pu, J. (2020, October). 2d cnn versus 3d cnn for false-positive reduction in lung cancer screening. *Journal of medical imaging (Bellingham, Wash.)*, *7*(5), 051202–. Retrieved from https://lens.org/041-586-834-140-199 (https://pubmed.ncbi.nlm.nih.gov/33062802/ ; https://www.spiedigitallibrary.org/journals/journal-of-medical-imaging/volume-7/issue-5/051202/2D-CNN-versus-3D-CNN-for-false-positive-reduction-in/10.1117/1.JMI.7.5.051202.pdf ; https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7550796 ; https://www.spiedigitallibrary.org/journals/journal-of-medical-imaging/volume-7/issue-05/051202/2D-CNN-versus-3D-CNN-for-false-positive-reduction-in/10.1117/1.JMI.7.5.051202.full) doi: 10.1117/1.jmi.7.5.051202

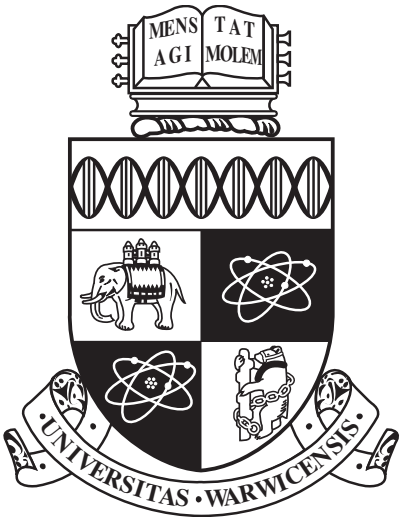
Appendices

A Specification

University of Warwick
Department of Computer Science
CS310

Developing AI-based tumour detection techniques for Lungs of CT scans

Project Specification



2108182
2023-10-11

Contents

1 Project Problem

Lung cancer is the leading cause of cancer death in the UK, accounting 21% of all cancer deaths in 2017-2019 (Cancer Research UK, 2022). Lung cancer is often diagnosed at advanced stages, while early detection dramatically improves survival rates (World Health Organization, 2023). As such, it is important that techniques of early detection are refined with high accuracy to maximise survival rates. Published research suggests computer tomography (CT) scans are now routinely and commonly used to screen for lung cancer, despite the existence of other methods due to efficiency of CT (Nooreldeen & Bach, 2021). Newly published research suggest the existence of artificial intelligence (AI) methods to detect tumours early with high success rate. However, the research area of AI methods for detection of tumours on new technology such as computer tomography scanning is recent: progress on both areas is developing rapidly and upcoming. A consequence of its recency is that extensive research is still being developed, and there are still many methods that have yet to be attempted to achieve success or failure. The purpose of research is not to just find successful methods, but also to report of methods that fail to succeed or hinder success. As such, we seek to create, modify or combine methods with the hope of enhancing or improving tumour detection in CT Lung scans. Furthermore, all information published thus far is research specific, meaning that there does not exist a software available to aid doctors diagnosing patients today which can be thought of a possible extension to the project.

2 Objectives

2.1 Must Have

- Make software compatible with data format submitted using DICOM
- Must be able to sort DICOM data to ensure correct learning
- $\geq 75\%$ success rate in detection of all tumours in a CT-scan from at least 1 method used

2.2 Could Have

- The usage, analysis and comparison of 2 different methods.
- $\geq 85\%$ success rate in detection of all tumours in a CT-scan from 1 method
- $< 50\%$ false positive rate from sample of all failures

2.3 Should Have

- The usage, analysis and comparison of ≥ 3 different methods.
- $\geq 95\%$ success rate in detection of all tumours in a CT-scan
- Easy to use UI to feed a CT scan with most successful method
- $< 35\%$ false positive rate from sample of all failures
- Malignancy/Benign detection with $\geq 75\%$ success.

3 Methodology

The project will be delivered by mainly utilising the agile method. There will be sprints, for which in each sprint:

1. Each sprint will begin with an initial research. My project contains content that has not been taught fully yet by any of the modules, therefore, I will be learning as I progress through. As a result of this, it is crucial that I have initial research of the next step I am taking. This can range anywhere from theory of statistics to machine learning.
2. At the end of each research, a day or two will be spent on noting down everything that I have learned so it stays in my mind throughout. Furthermore, I will be able to refer back to these notes later in the future.

3. After noting, a phase of development begins. In this phase I will be developing the artificial intelligence framework. The developed task in this phase can range anywhere from developing a method of converting data into something more readable to training the artificial intelligence model.

4. At the end of each run, I will write a short report of what has been achieved and how. These write ups will aid me on concluding the result of this sprint, how it was done, what needs to be done next, and which will aid me writing the final report and the progress report.

The above describes methodology at which the project will take place. However, it is also useful that we describe currently planned steps:

1. Initial research on existing methodologies and papers will be conducted. Initial theory around machine learning, deep learning and neural networks will be done.

2. A script that sorts DICOM data into readable data such as .png will be programmed.

3. Further research done on initial steps of AI development. At this point the first framework for the AI will be pinpointed.

4. The development of initial AI framework begins until it is finished.

5. Data is separated in 2 ways: data used for training and data used for testing (and possibly validating). After separation, data is translated and learned by the AI.

6. The learning will be tracked: there will be an attempt to find the best parameters and hyper-parameters possible for the data I have been given. As such, training error and test error will be tracked, and methods such as k-fold validation will be used.

7. Accuracy is tested of the AI framework with data that has been kept in reserve for this exact purpose only. Accuracy will be measured in specific ways, e.g., overall accuracy, false positives etc.

8. The method is either modified, or a new method is proposed.

9. If a new method is proposed, steps 3 to 5 are repeated. However, it is likely that initial AI will not be perfect, and therefore slight changes to parameters or methods are likely before proposal of a new method.

It is also to note that methods can range anywhere from brand new methods that I propose, or will be methods that are modifications of already existing AI frameworks (with credit). The exact number of methods I will be able to do is currently not well known, due to the nature of the project (I.e., some methods may require extra time to obtain high accuracy, other methods may be naturally high in accuracy but may have other problems e.g., high false-positive rate. Attempting to fix issues may cause me to run out of time). The aim is to do one method per term, although this may be adjusted as I progress. In either case, used AI methodology will likely be modified (either methodologically or parametrically) to suit the nature of my data with the aim of achieving better accuracy. Findings as I modify the AI framework will be written, noted and included in the final report. There will be a strong consideration and effect of possible bias throughout the research e.g., biased secondary data. Furthermore, testing methodology is prone to change as it is possible I may find better fairer methods of testing data correctness.

4 Timeline

4.1 Weekly Timetable

Table 1: Term 1 Timetable

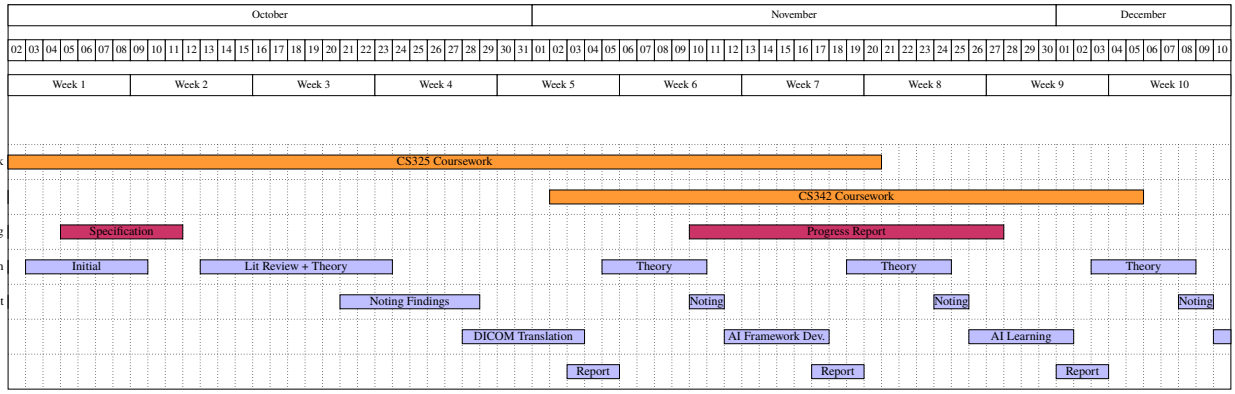
Time	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
9:00	Free	Free	Lecture	Free	Free	Free	Free
10:00	Lecture	Free	Seminar	Lecture	Free	Free	Free
11:00	Free	Free	Lecture	Free	Free	Free	Free
12:00	Free	Free	Free	Free	Free	Free	Free
13:00	Lecture	Lecture	Free	Free	Free	Free	Free
14:00	Free	Free	Free	Seminar	Free	Free	Free
15:00	Lecture	Free	Free	Free	Workshop	Free	Free
16:00	Lecture	Free	Free	Free	Free	Free	Free
17:00+	Free	Free	Free	Free	Free	Free	Free

Table 2: Term 2 Timetable

Time	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
9:00	Free	Free	Free	Free	Free	Free	Free
10:00	Free	Lecture	Free	Free	Free	Free	Free
11:00	Lecture	Free	Free	Free	Lecture	Free	Free
12:00	Free	Free	Free	Free	Free	Free	Free
13:00	Free	Lecture	Free	Free	Free	Free	Free
14:00	Free	Free	Free	Free	Lecture	Free	Free
15:00	Free	Free	Free	Free	Free	Free	Free
16:00	Lecture	Free	Free	Free	Free	Free	Free
17:00	Free	Free	Free	Lecture	Free	Free	Free
18:00+	Free	Free	Free	Free	Free	Free	Free

- Yellow - indicates times that I will study for either the project OR another coursework
- Green - indicates times that I will be taking a break
- Red - indicates times that I am confirmed busy according to university timetable.
- For term 1, I am free after 17:00. I will work every weekday from 16:00-18:00, therefore 17:00+ is highlighted yellow.
- For term 2, I am free after 18:00. I will work every weekday from 18:00 - 20:00, therefore it is highlighted yellow.
- Term 2 timetable is subject to change as seminars and workshop timetables have not been announced yet.
- For both terms, it amounts to approximately $7 \times 6 = 42$ hours of extra work per week.
- I plan to have minimum of ≥ 25 hours allocated each week for this project.

4.2 Gantt Chart



5 Resources and Risks

The nature of this project requires a strong graphics processing unit computer to learn data as fast as possible. Therefore, it is possible that the project will use the University of Warwick graphics processing unit batch computer system. Furthermore, my computer at home has an available RTX 3090 for use, which is a very suitable computer for learning in case the batch computer system fails to work.

Furthermore, even though the data I receive is fully anonymised, there exists a risk of leaking the data somehow. For example, it is possible that during the use of batch computing system, someone may access the computer with the data in it. To mitigate such circumstances, I plan to double check to ensure that data is fully anonymous, and further, will include encrypt any data that is stored within my devices using password protection.

The number of patient data I receive is not currently known until the project starts. Indeed, higher data will output better results. However, it is important to note that this project is very time restricted, and therefore, the time it takes to process data will be tested before requesting it. This will ensure that I do not have more resources that I need.

Furthermore, a problem may occur when transferring data to my computer. It follows that each patient is approximately 1 GB of data, therefore, mass transferring may be difficult. To mitigate this, we may transfer the data in batches over time. As such, it is important that I come in contact with them as soon as possible to begin the process and minimise any delay.

It was decided that the framework for the AI will be written in Python. Python has an abundance of AI-related packages that prove to be useful for my project, such as PyToch 2.0. Furthermore, Python has existing packages that can aid me in developing the data translation using the package PyDicom. The project code, along with all written reports, will be stored in my own private GitHub repository to ensure that I can work from anywhere.

6 Legal, Social, Ethical and Professional Issues and Considerations

The data used for this project is fully anonymised and provided by a hospital located outside of United Kingdom. Potential need for ethical review has been considered due to the nature of required data. Upon further contacting dissertation supervisor (Dr. Ligeng He) and support administrator for potential requirement of an ethical review (Dr. Mike Joy) it was concluded that ethical review is not required. There are no other issues to be considered for this project.

References

Cancer Research UK. (2022, May). Retrieved 8 October 2023, from `https://www.cancerresearchuk.org/health-professional/cancer-statistics/mortality/common-cancers-compared#heading-Zero`

Nooreldeen, R., & Bach, H. (2021, Aug). Current and future development in lung cancer diagnosis. *International Journal of Molecular Sciences*, *22*(16), 8661. doi: 10.3390/ijms22168661

World Health Organization. (2023, Jun). World Health Organization. Retrieved 8 October 2023, from `https://www.who.int/news-room/fact-sheets/detail/lung-cancer`

B Image Converter

```
import cv2
import numpy as np
import pydicom
import os
import shutil

input_dir = "/home/xein/Downloads/DATA/Lung-PET-CT-Dx"
output_dir = "/home/xein/Downloads/DATA/Lung-PET-CT-Dx_PNG"

# Function to find the folder containing .dcm files
def find_dcm_folder(directory):
    for root, dirs, files in os.walk(directory):
        for file in files:
            if file.endswith(".dcm"):
                return root
    return None

# Create output directory if it doesn't exist
if not os.path.exists(output_dir):
    os.makedirs(output_dir)

# Loop through patient folders
for patient_folder in os.listdir(input_dir):
    patient_dir = os.path.join(input_dir, patient_folder)
    dcm_folder = find_dcm_folder(patient_dir)
    if dcm_folder:
        output_patient_dir = os.path.join(output_dir, patient_folder)
        os.makedirs(output_patient_dir, exist_ok=True)
```

```
29          for root, dirs, files in os.walk(dcm_folder):
30              for file in files:
31                  if file.endswith(".dcm"):
32                      dcm_path = os.path.join(root, file)
33                      ds = pydicom.dcmread(dcm_path)
34                      img = ds.pixel_array
35                      # img = (img - img.min()) / (img.max() - img.min())*
                            255.0
36                      # img = (img / img.max()) * 255.0
37                      img = cv2.normalize(img, None, 0, 255, cv2.NORM_MINMAX
                            , dtype=cv2.CV_8U)
38                      # img = img_as_ubyte(exposure.rescale_intensity(img))
39                      output_path = os.path.join(output_patient_dir, file +
                            ".png")
40                      cv2.imwrite(output_path, img)
```