# FACULTY OF SCIENCE, ENGINEERING AND COMPUTING

## School of *Computer Science & Mathematics*

## BSc DEGREE

## IN

***COMPUTER SCIENCE***

# PROJECT REPORT

Name: Gary Osman

ID Number: K1548873

Project Title: Neural Network Hyperparameter Tuning for Breast Cancer Histology Image Classification

Project Type: Research

Date:28/04/2019

Supervisor: Ahmed Shihab

Kingston University London

# Plagiarism Declaration

The following declaration should be signed and dated and inserted directly after the title page of your report:

**Declaration**

I have read and understood the University regulations on plagiarism and I understand the meaning of the word *plagiarism*. I declare that this report is entirely my own work. Any other sources are duly acknowledged and referenced according to the requirements of the School of Computer Science and Mathematics. All verbatim citations are indicated by double quotation marks ("…"). Neither in part nor in its entirety have I made use of another student's work and pretended that it is my own. I have not asked anybody to contribute to this project in the form of code, text or drawings. I did not allow and will not allow anyone to copy my work with the intention of presenting it as their own work.

Date 28/04/2019

Signature: *Gary Osman*

## Acknowledgements

The author wishes to thank Dr Ahmed Shihab for his help and support throughout the duration of this project.

A big thank you to family, friends and colleagues who have given me the freedom and encouragement to push myself outside of my comfort zone.

Finally, the author wishes to acknowledge the cancer research and medical communities for their efforts in helping patients to fight cancer.

## Abstract

This report explores the performance tuning of neural networks in the application of image classification for breast cancer histology images. Through the adjustment of neural network parameters and the analysis of various performance metrics, the author aimed to identify the relationships that these parameters have in optimising the ability of the neural network to classify the images accurately. The experiments conducted resulted in a neural network capable of classifying sections of microscope slide images of with an accuracy of 72%. The report also seeks to contribute to a wider discussion around the usage of neural networks within cancer diagnostics.

# Table of Contents

# CHAPTER 1: INTRODUCTION

## 1.1.1 Problem Domain

In 2015 there were approximately 55,000 incidences of invasive breast cancer in the UK, accounting for 15% of total cancer diagnoses, making it the most common cancer in the UK (Cancer Research UK, 2016). In a separate study by the American Society for Cancer it was found that Invasive/Infiltrating Ductal Carcinoma (IDC) makes up 80% of invasive breast cancer diagnoses (Breastcancer.org, 2019) .With such a high proportion of potential incidence rates, there is a clear need to ensure that diagnoses can be made quickly and accurately.

This research will investigate the possibility of using deep learning as a diagnostic aid for the identification of IDC. The focus will be in the usage of neural network architecture for image classification. Using a dataset of microscope images featuring evidence of IDC vs non-IDC cells, the research will explore a range of neural network parameters and evaluation heuristics and their application in image classification, to identify the highest accuracy in classifying slide images.

The report will introduce neural networks, looking into the evolution of this technique and challenges that were faced both technically and from the perspective of adoption in the research community. The methodology section of the report explores the concept of "Agile Science" (Hekler *et al.*, 2016), proposing a hybrid approach to research involving an iterative software development mindset. Finally, the report focuses on the execution and analysis of the experiments carried out to answer the research question, before offering research conclusions and a critical review of the process.

## 1.1.2 A view from history

Artificial Intelligence (AI) has captured the imaginations of thinkers and tinkerers of various disciplines long before the ability to put theory into practice was ever available.
In an essay for the Editor of The Press titled "Darwin Amongst the Machines" (Butler, 1863) considers the question:

"*What sort of creature man's next successor in the supremacy of the earth is likely to be*"?

In this essay Butler claims:

"*Day by day, however, the machines are gaining ground upon us; day by day we are becoming more subservient to them; more men are daily bound down as slaves to tend them, more men are daily devoting the energies of their whole lives to the development of mechanical life."*

This essay is a pre-cursor to the novelist's work "Erewhon" (Butler, 1999), in particular, the chapter: "Book of the Machines", in which the Butler further explores the rapid evolution of machines relative to other ecological systems, highlighting humanity's increasing dependency on machines.

Following the emergence of the Perceptron (Rosenblatt, 1958) (described in chapter 2) The New York Times (New York Times, 1958) published an article hailing the potentials of this new technology, describing them as:

*"The embryo of an electronic computer today that it expects will be able to walk, talk, see, write, reproduce itself, and be conscious of its existence".*

In the article the U.S Navy claims:

*"Later Perceptrons will be able to recognise people and call out their names and instantly translate speech in one language to speech or writing in another language".*

Fast-forward to today, and many of those predictions are starting to take shape in one form or another, as the "embryo" has developed and matured. Autonomous Robots (Murphy *et al.*, 2011) and Vehicles (Tesla, 2016) , Facial Recognition systems (Perry, 2018), and speech to speech translators (Hyman, 2014) are becoming more common place in mainstream technology.

Butler's essay reads as a warning, offering a pessimistic view on the evolution of machines, while The New York Times article's author revels in the potential applications. The author of this paper sees the two perspectives expressed in separate domains. The former situated in an ethical domain, the latter in the domain of scientific pursuit, unconstrained by ethical considerations.

The research presented in this report seeks to lie within a convergence of point of these two domains, permitting rigorous scientific pursuit, within an application to which people remain the beneficiary. The research will focus on one area of AI, namely image classification with deep learning and its potential for application in healthcare.

### 1.1.3 Future Challenges

The hype cycles published by Gartner (2016, 2017, 2018) have consistently identified machine learning and neural networks at the “peak of inflated expectation” on the curve for the last three years suggesting another potential phase of ”disillusionment” with the technology.

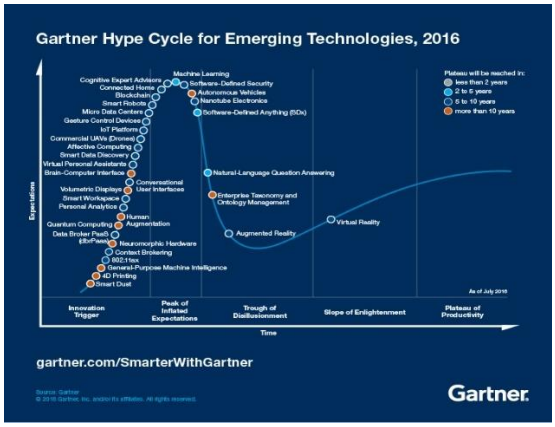



Figure 1: Hype Cycle 2016 (Gartner, 2016)

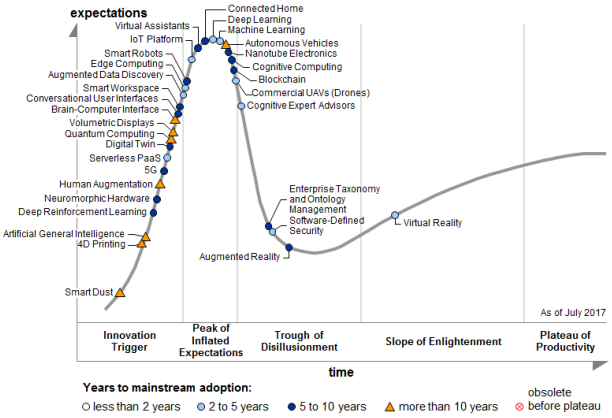



Figure 2: Hype Cycle 2017 (Gartner, 2017)

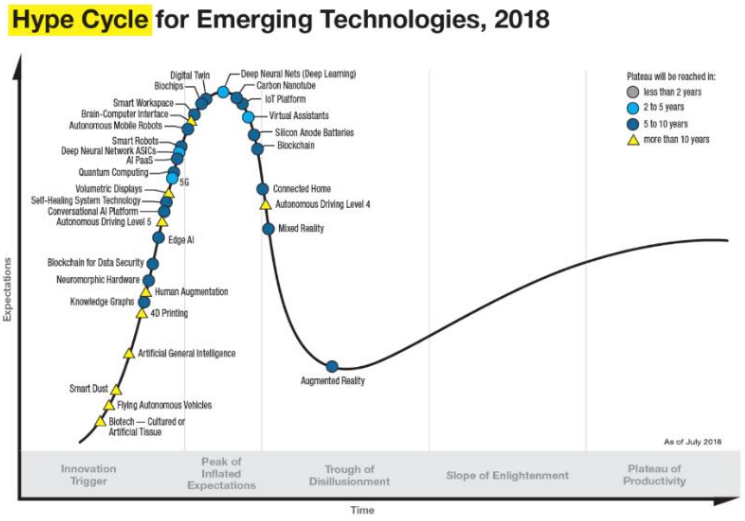



Figure 3: Hype Cycle 2018 (Gartner, 2018)

Without a suitable range of successful applications across the various sectors, the risk that the technology is overhyped can lead to a rejection in adoption, as was demonstrated by the criticisms levelled at Perceptrons (see section 2.4) (Hunt, Minsky and Papert, 1971), The threat of rejection could lead to another period of underfunding in this sector, as business value is sought from other emerging technologies. The hope is that this research area will contribute to topic of successful use-cases for this technology.

# 1.2 Aims and Objectives

## 1.2.1 Aims

The main aim of this project is to explore the relationship between neural network architecture parameters and the quality of detection of cancer in microscope slide images.

An additional aim is to contribute to the debate into the use of deep learning in breast cancer diagnostics.

## 1.2.2 Objectives

- To pre-process a sufficiently large (approx. 100k) dataset of images to be used as training and test data for consumption by a neural network before December.
- To demonstrate an image classification system with an accuracy of at least greater than 50% (better than a random choice) before the end of the project.
- To utilise the architecture of at least one deep learning design pattern (i.e. Convolutional Neural Network) before the early prototype demo.
- To provide a comparison in accuracy between at least two deep learning parameters before the end of the project.

# 1.3 Scope

As with any project it is necessary to identify a suitable scope so that the project team can maintain focus towards an end goal. The author is familiar with the negative impacts of changing scope, spreading the resources to thinly and reducing the time to achieve an acceptable output.

With approximately 6 months to complete the project, the author will be focusing on a single dataset to be used as a source for training any neural network architecture proposed. In terms of network architecture, it has been specified that at least two network parameters be tested. Therefore, within this architecture it needs to be possible to test different parameters in order to observe a change in output. This performance tuning must be limited to ensure the wider objective can be achieved in the time allowed, while still offering an insight into the nature of the effects.

The use of frameworks will assist the rapid development of a neural network (see section 2.12). With many variations available, for the purposes of this project, only one framework will be used to increase the likelihood that at least one end to end development of a functioning network can take place.

# CHAPTER 2: LITERATURE REVIEW

## 2.1 Neural Networks in Artificial Intelligence

Artificial Intelligence (AI) is a often used in public discourse, but to describe it as if it were a single entity is disingenuous. AI comes in many different forms depending on the application. This review will explore the evolution of neural networks (NN) as a concept, identifying cutting edge advancements in research and technology which helped to facilitate its evolution. The review will provide insights into various architectures with reference to peer-reviewed research. Any anecdotal views used within the scope of this research have been scrutinised to ensure that the subjects are respected within the research community.

Within AI two broad approaches are common:

**Weak/Narrow AI** – task specific intelligence. A specialised system that can perform a single task equal to or greater than that of human ability

**Strong/General AI** (referred to as AGI) – able to perform a range of tasks equal to or greater than the ability of a human.

When researching into the evaluation of such approaches (Hernández-Orallo, 2017) asserts that:

*"The intention of stressing this duality is that this should necessarily pervade the evaluation procedures in AI. Specialised AI systems should require a task-oriented evaluation, while general-purpose AI systems (also known as AGI systems, from the term 'artificial general intelligence') should require an ability-oriented evaluation. In practice, however, we see that some general-purpose AI systems are evaluated with a narrow set of tasks"*

It is important to acknowledge this in any research carried out. As the evaluation of a model should be appropriate for the goal of the project. Not just from a network performance perspective but also from the perspective of stakeholders as the ability of a network may be viewed subjectively, dependent on the respective parties' perspective.

## 2.2 Types of Learning

Within these approaches the notion of learning must be considered. How is the 'knowledge' to perform a task acquired by a system? Two possible approaches are: (Japkowicz, 2001)

**"Supervised Learning"** - The process relies on the training of a system using labelled data which the network processes. The output is then compared with the label associated to the data and if an error has occurred, modifications to the network are applied to reduce the error rate.

**"Unsupervised Learning"** - Here a system will attempt, often through a process of trial and error, to achieve a desired output. Examples may include brute forcing permutations of moves to achieve successful completion of a game or attempting to cluster information to gain insights into large unstructured datasets.

## 2.3 The Biological Link

Neural networks are an attempt to simulate the biological activity which enable animals to learn. A particular focus is paid to the activity of neurons in the brain. Neurons are able to take input (in the form of sensory signals) through dendrites which are in essence programmed to respond to particular features of the input. There is a threshold at which the neuron, given the correct sensory signals, will fire sending a pulse along the axon. This output is then propagated through a network of neurons that either fire or not based on the output pulse generated (Fitch, 1944) . The research into the receptive field and neural activity of cats when subjected to visual stimulus (Hubel and Wiesel, 1959) inspired

the structure of the Perceptron (Rosenblatt, 1958), a technological neuron capable of producing an output in response to an input with respect to an activation function over a suitable threshold.

## 2.4 Challenge to Perceptrons

The perceptron came under opposition by proponents of other computing paradigms such as serial computing (Von Neumann, 1993). The book titled "Perceptrons" (Hunt, Minsky and Papert, 1971) offered quite open criticism of the approaches by researchers in the field of Perceptrons.

"*We do not see that any good can come of experiments which pay no attention to limiting factors that will assert themselves as soon as the small model scaled up to a usable size"* (Hunt, Minsky and Papert, 1971). This publication is recognised as a pivotal contribution to an observable decline of research in this field between the 1960s and 80s (Olazaran, 1996). Later editions of "Perceptrons" (Hunt, Minsky and Papert, 1971) included a prologue where the author acknowledges this attribution but asserts the notion that the research served to re-focus the community and drive a more methodical approach, leading to innovations in the understanding of learning.

Hindsight shows that the divergence in research lead to the widespread distribution of serial/sequential computation, that modern society has come to rely on. It can be argued that the countless architectures and frameworks that lead to the advances and dissemination of knowledge of computation to the masses has allowed researchers to 'walk before running' into the complexities of parallel computation. In agreement with Minsky, the author of this report suggests that the limitations of serial computing are now well understood, highlighting use-cases where parallel computing is more suitable.

## 2.5 The Cognitron & Neocognitron

The work of Fukushima (1975, 1983) appeared to have reignited research into neural networks with an architecture for pattern recognition starting with the "Cognitron" (Fukushima, 1975) which was then improved upon with the creation of the "Neocognitron" (Fukushima, Miyake and Ito, 1983). The limitation of the cognitron was that it treated the same pattern in different regions of an image as different patterns. The neocognitron offered a solution to this problem.

## 2.6 Multi-Layer Perceptrons

In roughly the same period of time, Multi-Layer Perceptrons (Rumelhart, 1985) proposed a solution to the XOR problem identified (Hunt, Minsky and Papert, 1971). The "Backpropagation" of error has proven highly effective in reducing the error rates of outputs from neural networks as shown in Convolutional Neural Networks (Lecun *et al.*, 1989). All this was happening in and amongst a backdrop of computational adoption in wider society, increasing the demand for innovations in the technology field.

## 2.7 Enablers of Neural Networks

### 2.7.1 Processors

The Von Neumann architecture (Von Neumann, 1993), is by designed for sequential processing. The nature of this architecture limits the speed of a program's operation. With neural networks being a simulation of cognition, it is natural to consider the need for parallelism of processing. The plateau in the yield related to Moore's law (Moore, 1965), as the physical limits of single core central processing are being reached, has resulted in the emergence of multi-core CPUs (Esmaeilzadeh *et al.*, 2012). These are an important feature of modern computing to compensate for this limitation. However, as remarked in the same research, this requires the development of applications that are capable of parallel processing. This shift in mindset has itself, been running in parallel with research into neural networks.

The Graphics Processing Unit, built for rendering graphics, are inherently parallel. These processors possess hundreds of cores and have been instrumental in the successes seen in neural network development. The time taken to train neural nets have reduced as these technologies have improved and GPUs are now being built specifically for artificial intelligence applications such as autonomous vehicles (Tesla, 2016).

### 2.7.2 Data

This past decade has seen an increased focus in all areas related to data. Social media platforms have exposed a wealth of information about consumers, and an increase in internet connected devices offer sensor information. The combination of advancements in data generation and storage and "the expectation that data should be made freely available accessible to global research networks" (Leonelli, 2012a) has a profound impact in scientific research. Data lake companies have been established with the sole purpose of storing data for mining later. The power of data has led to new roles such as Data engineers and Data scientists tasked with the management and analysis of these vast sources.

## 2.8 Neural Network Applications

With computers firmly established in society the demand for automated processes would inevitably increase. As the imaginings of science fiction writers now seem possible, the applications were now starting to be realised. With different applications came different iterations of network architecture. A multitude of architectures are now available to researchers in different fields. The table below (Zhou *et al.,* 2018) provides an overview of different applications and the corresponding algorithms and network architectures that a researcher may wish to explore when attempting to innovate in the field.

Applications of graph neural networks.

| Area | Application | Algorithm | Deep Learning Model |
|---|---|---|---|
| Text | Text classification | GCN | Graph Convolutional Network |
| | | GAT | Graph Attention Network |
| | | DGCNN | Graph Convolutional Network |
| | | Text GCN | Graph Convolutional Network |
| | | Sentence LSTM | Graph LSTM |
| | Sequence Labeling (POS, NER) | Sentence LSTM | Graph LSTM |
| | Sentiment classification | Tree LSTM | Graph LSTM |
| | Semantic role labeling | Syntactic GCN | Graph Convolutional Network |
| | Neural machine translation | Syntactic GCN | Graph Convolutional Network |
| | | GGNN | Gated Graph Neural Network |
| | Relation extraction | Tree LSTM | Graph LSTM |
| | | Graph LSTM | Graph LSTM |
| | | GCN | Graph Convolutional Network |
| | Event extraction | Syntactic GCN | Graph Convolutional Network |
| | AMR to text generation | Sentence LSTM | Graph LSTM |
| | | GGNN | Gated Graph Neural Network |
| | Multi-hop reading comprehension | Sentence LSTM | Graph LSTM |
| | Relational reasoning | RN | MLP |
| | | Recurrent RN | Recurrent Neural Network |
| | | IN | Graph Neural Network |
| Image | Social Relationship Understanding | GRM | Gated Graph Neural Network |
| | Image classification | GCN | Graph Convolutional Network |
| | | GGNN | Gated Graph Neural Network |
| | | ADGPM | Graph Convolutional Network |
| | | GSNN | Gated Graph Neural Network |
| | Visual Question Answering | GGNN | Gated Graph Neural Network |
| | Object Detection | RN | Graph Attention Network |
| | Interaction Detection | GPNN | Graph Neural Network |
| | | Structural-RNN | Graph Neural Network |
| | Region Classification | GCNN | Graph CNN |
| | Semantic Segmentation | Graph LSTM | Graph LSTM |
| | | GGNN | Gated Graph Neural Network |
| | | DGCNN | Graph CNN |
| | | 3DGNN | Graph Neural Network |
| Science | Physics Systems | IN | Graph Neural Network |
| | | VIN | Graph Neural Network |
| | | GN | Graph Networks |
| | Molecular Fingerprints | NGF | Graph Convolutional Network |
| | | GCN | Graph Convolutional Network |
| | Protein Interface Prediction | GCN | Graph Convolutional Network |
| | Side Effects Prediction | Decagon | Graph Convolutional Network |
| | Disease Classification | PPIN | Graph Convolutional Network |
| Knowledge Graph | KB Completion | GNN | Graph Neural Network |
| | KG Alignment | GCN | Graph Convolutional Network |
| Combinatorial Optimization | | structure2vec | Graph Convolutional Network |
| | | GNN | Graph Neural Network |
| | | GCN | Graph Convolutional Network |
| | | AM | Graph Attention Network |
| Graph Generation | | NetGAN | Long short-term memory |
| | | GraphRNN | Rucurrent Neural Network |
| | | Regularizing VAE | Variational Autoencoder |
| | | GCPN | Graph Convolutional Network |
| | | MolGAN | Relational-GCN |

Figure 4: Network Algorithms and Architectures by Application

(Zhou *et al.*, 2018)

With the many areas identified in Figure 4, a few notable deep learning models seem to own the space of research. However, across all applications, variations of Convolutional Neural Network (Lecun *et al.*, 1989) appears consistently.

# 2.9 Convolutional Neural Networks in detail

The creation of the first convolutional neural network (CNN) (Lecun *et al.*, 1989) is now commonly used in variety of domains as shown in Figure 4. The images in Figures 5 and 6 show the architecture and functionality within a convolutional neural network. Conceptual descriptions are provided below

**Figure 2. An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 290,400–186,624–64,896–64,896–43,264–4096–4096–1000.**

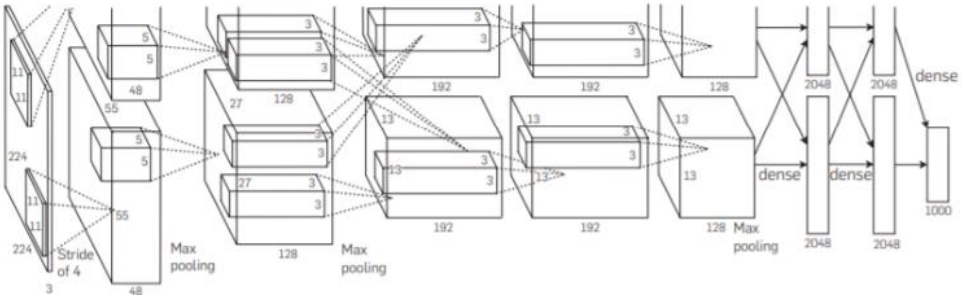



Figure 5: AlexNet - A competition winning example of a CNN (Krizhevsky, A., Sutskever and Hinton, 2017)

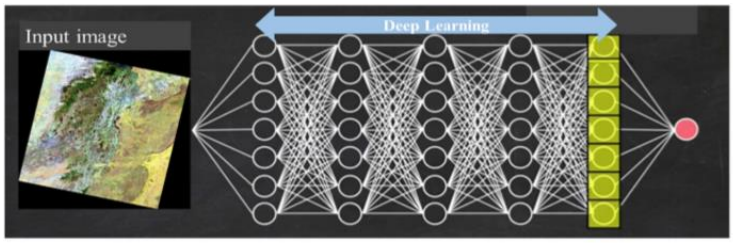



**Fig. 2.** Deep Learning Neural Network.

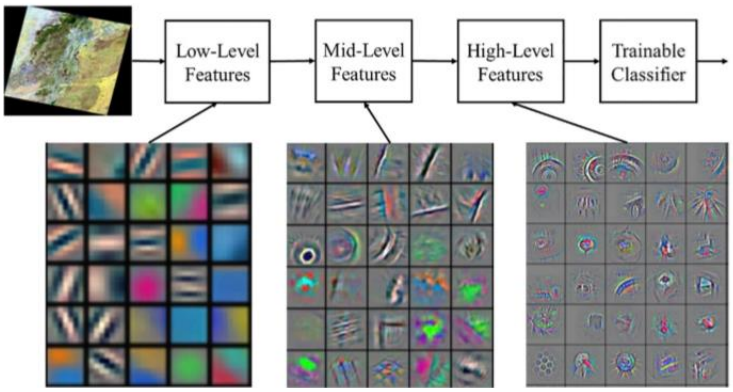



Figure 6: An example of feature extraction maps in hidden layers visualised (Traore, Kamsu-Foguem and Tangara, 2018)

## 2.9.1 Key Features – a conceptual view

Definitions are informed by the tutorials of the Microsoft Cognitive Toolkit (Seide, 2016)

#### *2.9.1.1 Sliding Window (Convolution/kernel)*

A technique which involves a matrix of nxn dimensions passing over the data set to capture a subset of that region of data. This subset is then passed to a function which reduces the region to a single piece of data to be fed to the next layer. This technique is contrasted with the initial input of the dataset flattened into a single vector. This gives the advantage of preserving features in the image for analysis. A flattened vector may lose features such as edge detection and shading that a biological neuron would utilise to distinguish notable features in an image.

#### *2.9.1.2 Stride*

The stride dictates the distance which the window slides in each convolution of the data set. For example, a stride = 1 would shift the sliding window horizontally 1 pixel in an image for the next subset to be taken.

#### *2.9.1.3 Padding*

Padding is a technique in which 'space' is added to the data. For example, in a square image, a border of x number of pixels can be applied around the edge to ensure that as the window slides across the image, detail which may be pertinent to classifying the image that lays on the edge is captured.

#### *2.9.1.4 Pooling*

A method for reducing the data in a 'window' to be fed to the next node in the network. Two common methods for achieving this are:

- Max Pooling – taking the maximum value within the window as the value to be fed forward
- Average Pooling – taking an average of the data captured and feeding that value forward to the next node.

#### *2.9.1.5 Activation*

The Activation Function is used to simulate the 'firing' of the neuron in response to the input. This firing takes place when features of the input are aligned with the features identified with the training data at a particular node. A threshold for the output probability is defined for the activation of a node. Mathematical functions are used to reduce the values from the outputs, typically by mapping them to a value between either 1 and 0 (Sigmoid) or -1 and 1 (Rectified Linear Unit - ReLu)

#### *2.9.1.6 Back propagation*

The concept of backpropagation has been crucial in solving problems such as the XOR problem (Hunt, Minsky and Papert, 1971). The basic principle is that during each cycle through the network the error is propagated backwards and used to amend the weighting associated with feature maps in the 'neuron'. The effect is to reduce the error and loss in the output neurons during each cycle. One function to help reduce this loss is Stochastic Gradient Descent, (Robbins and Monro, 1951) which is an optimisation algorithm designed to converge on a global minimum by randomly sampling the gradient. This can be described as a series of steps or jumps on a graph in an attempt to find a global minimum. Imagining a graph as a three-dimensional surface, from the perspective of the algorithm, it is moving in the direction of the gradient of a valley. The learning rate is a crucial part of this algorithm as it defines the size of step to take in each sample. Because the nature of the graph or "surface terrain" is unknown too big a step may result in missing the minimum and too small a step runs the risk that the optimiser never converges fully on the global minimum or gets trapped in a local minimum. The learning rate is a value between 0 and 1.

## 2.9.2 Generative Adversarial Networks

In an interview (Forbes, 2016) LeCun, the father of the Convolutional Neural Network, described "Generative Adversarial Networks" (Goodfellow *et al.*, 2014) as "*the most interesting idea in the last 10 years in ML*" The generative adversarial approach is an attempt to create synthetic data which is as close to the real data as possible. A common problem in the world of neural networks is the availability of data to train networks. This includes variations in data, for example an image is unlikely to represented in all visual dimensions. There is a need to be able classify the same objects rotated on different axis or in different positions within an image. The ability to generate this data synthetically could significantly reduce the effort required to train neural networks. This may explain the excitement around this particular approach.

#### *2.9.2.1 Structure*

A GAN consists of two networks referred to as the "Generative" and the "Discriminative" (Goodfellow *et al.*, 2014) . The "generative" is purposed with creating synthetic data based on the inputted data available to it. The "discriminative" (convolutional) network is then tasked with classifying whether or not the synthetic data is real or fake, based on the training data that has been supplied to it. The error can be backpropagated in order to improve overall performance of the network.

# 2.10 Performance Benchmarks

The notion that the performance of an artificial intelligence must be judged by comparison with a flesh and bone counterpart, can be traced back to the "Imitation Game", (Turing, 1950) exploring the question "Can machines think?". Several task-based challenges have been presented in the history of computing, that have attempted to compare the ability of an algorithm or piece of technology with that of a human expert in the same field.

## 2.10.1 IBM deep blue – The chess winning machine

During the discussion of the "imitation game" (Turing, 1950), the "interrogator" asks the machine a series of questions to identify whether the subject is indeed a machine. One such question is "Can you play chess?". This question was more than answered in 1997 when IBM Deep Blue (Feng-Hsiung Hsu, 1999) became the first world champion class chess computer defeating the current champion Garry Kasparov

## 2.10.2 ImageNet - AlexNet

The imageNet LSVRC- 2012 competition was one of the notable achievements in neural network applications. The "AlexNet" CNN beat other rivals in the image classification challenge by achieving an error rate of 15.3%, compared to 26.2% achieved by the second-best entry. (Krizhevsky, Alex, Sutskever and Hinton, 2017)

## 2.10.3 Cancer diagnostics image classification

In the realm of cancer diagnostics, there have been attempts to use neural networks image classification. The BACH Grand challenge (Aresta *et al.*, 2018) provides researchers with a dataset of breast cancer microscopy images for testing image classification models This challenge looks at multi-class classification (Normal, Benign, In-situ, Invasive). Here are some state-of-the-art results:

- "An average accuracy of 85% over the four classes and 93% for non-cancer (i.e. normal or benign) vs. malignant (in situ or invasive carcinoma)" (Golatkar, Anand and Sethi, 2018)
- "Patch and image-wise accuracy of 75.73% and 81.25% respectively on the validation set and image-wise accuracy of 57%" (Nawaz *et al.*, 2018)
- "The cross-validation accuracy, averaged over four classes, was achieved to be 87%" (Iesmantas and Alzbutas, 2018)
- "A multiclass accuracy of 87%" (Kwok, 2018)

# 2.11 Challenges for Neural Networks

## 2.11.1 Data Quality

Many current technological solutions were not designed with neural network analytics in mind. As a result, for data to be ingested into a neural network it must often first be pre-processed to ensure compatibility with any proposed network architecture. These data quality issues are a hurdle to be overcome. Data lakes may have been created, but the purity of the water will dictate the value in the long run.

## 2.11.2 The Black Box

Another challenge is that the 'decision making' that takes place within a neural network is often obscured from users. This is referred to as a "black box" approach (Sun and Medaglia, 2018). The complexity involved with training the neural network with potentially millions of parameters to be understood is an obstacle to adoption of the technology. If incorrect decisions are made, little can be done but to acknowledge that the technology has not learned sufficiently. If this decision were to, as an example, impact a patient's treatment pathway, the reputational damage would inevitably set back the usage of artificial intelligence in the field of diagnostics.

## 2.11.3 Model Selection

Model Selection describes the process of choosing the correct model or network topology as a potential challenge given the rapidity with which new designs are created (Zhang *et al.*, 2018a). In the same research the author remarks that:

*"It is difficult to set and adjust the hyperparameters, and very small changes in hyperparameters are likely to change the effect of the training"*

This goes back to a fundamental lack of understanding in how the hyperparameters interact in the hidden layers. This also speaks to the topic of "Black Box" thinking (above). Perhaps in future, as Artificial General Intelligence (AGI) improves, researchers may find that the choice of network topology is itself selected by some form of network, able to analyse the hyperparameter interactions with greater accuracy.

## 2.11.4 Performance Metrics

With the varieties of networks and algorithms available, there is a requirement to find sensible metrics with which neural networks can be evaluated. Research into various implementations of neural networks in forecasting (Adya and Collopy, 1998) suggested the following approach to assessing the performance of the network:

**"Convergence**: Convergence is concerned with the problem of whether the learning procedure is capable of learning the classification defined in a data set."

**"Generalization:** Generalization measures the ability of NNs to recognize patterns outside the training sample… If performance on a new sample is similar to that in the convergence phase, the NN is considered to have learned well."

**"Stability**: Stability is the consistency of results, during the validation phase, with different samples of data. This criterion, then, evaluates whether the NN configuration determined during the learning phase and the results of the generalization phase are consistent across different samples of test data."

When Designing the neural networks some notable criterion for evaluation (Adya and Collopy, 1998) suggested were:

**"Network architecture:** Several variables such as the number of hidden layers and nodes, weight interconnections, and bottom-up or top-down design can determine the most effective NN architecture

for a problem. We considered whether a study had done sensitivity analyses with the number of layers and nodes in the architecture. Evaluating the other features of network architecture proves difficult given the level of disclosure typical of these studies.

**Gradient descent**: Manipulation of learning rate during training has been shown to lead to more effective gradient descent into the error surface."

With a new implementation of a neural network the author agrees that these are crucial features which need to be accounted for, which frameworks may not have pre-packaged. Evaluation should take place in stages and various outputs should be created to ensure as much transparency of the network's inner workings as possible are available to any potential users of the neural network.

## 2.11.5 Quantitative methods

With classification problems, accuracy alone may not be the most useful statistic for, for example, medical diagnostics. It may be the case that the researcher would wish to capture the highest number of patients with a condition even at the cost of misdiagnosing the healthy, as the cost of misdiagnosing an unhealthy patient may be far greater. (Cantor *et al.*, 1999) To do this a researcher may define a different threshold value for classification based on the probability output, or alternatively, select a model which over predicts towards one value, for the purposes of classifying a greater number of cases correctly while defining an acceptable cost for predicting the opposing case incorrectly.

With binary classification problems (i.e. healthy vs diseased), the ideal scenario would be that 100% of the cases are correctly classified. However, the various challenges in creating accurate models often result in a balance between successful and unsuccessful classifications for each class. As visualised below.(Saito and Rehmsmeier, 2015)

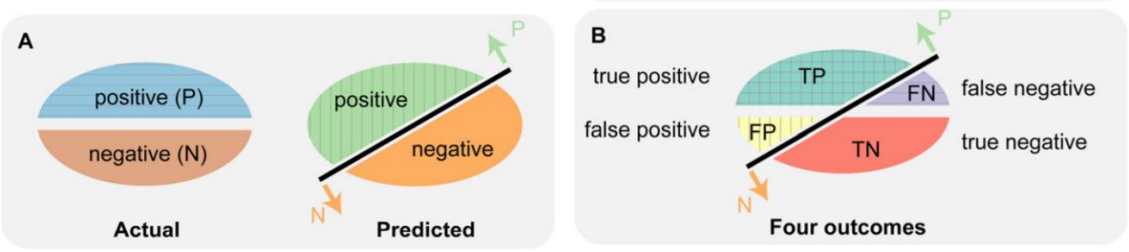



Figure 7: Binary classification terminology (Saito and Rehmsmeier, 2015)

With this scenario occurring frequently, several metrics have been established to identify the suitability of a model for various problem domains which a researcher may be asked to account for. These are described with reference to the confusion matrix (Kohavi, 1998)

**Table 8.2** Evaluation Measures

| Term | Definition | Calculation |
|---|---|---|
| Sensitivity | Ability to select what needs to be selected | TP/(TP+FN) |
| Specificity | Ability to reject what needs to be rejected | TN/(TN+FP) |
| Precision | Proportion of cases found that were relevant | TP/(TP+FP) |
| Recall | Proportion of all relevant cases that were found | TP/(TP+FN) |
| Accuracy | Aggregate measure of classifier performance | (TP+TN)/(TP+TN+FP+FN) |

Figure 8: Binary classifier and performance metrics and descriptions (Kotu, 2015)

Some definitions are interchangeable but may be used depending on the perspective of the research carried out. These interchangeable terms as well as other metrics are noted in the table provided below.

| Measure | Formula |
|---|---|
| ACC | (TP + TN) / (TP + TN + FN + FP) |
| ERR | (FP + FN) / (TP + TN + FN + FP) |
| SN, TPR, REC | TP / (TP + FN) |
| SP | TN / (TN + FP) |
| FPR | FP / (TN + FP) |
| PREC, PPV | TP / (TP + FP) |
| MCC | (TP * TN—FP * FN) / ((TP + FP)(TP + FN)(TN + FP)(TN + FN))$^{1/2}$ |
| $F_{0.5}$ | 1.5 * PREC * REC / (0.25 * PREC + REC) |
| $F_1$ | 2 * PREC * REC / (PREC + REC) |
| $F_2$ | 5 * PREC * REC / (4 * PREC + REC) |

ACC: accuracy; ERR: error rate; SN: sensitivity; TPR: true positive rate; REC: recall; SP: specificity; FPR: false positive rate; PREC: precision; PPV: positive predictive value; MCC: Matthews correlation coefficient; F: F score; TP: true positives; TN: true negatives; FP: false positives; FN: false negatives

Figure 9: binary classification performance metrics in detail (Saito and Rehmsmeier, 2015)

When discussing the ability to make these trade-offs by adjusting thresholds for classification whilst minimising the cost incurred for misclassification. Receiver Operating Characteristic curves (Swets, 1988) are used. These test the model at different thresholds to provide an insight into the ability of the model to achieve this aim.

## 2.12 Frameworks

The development of programming libraries and frameworks have increased the accessibility of artificial intelligence in both research and commercial environments. These frameworks allow users to stand on the shoulders of giants when it comes to the heavy lifting required to accomplish tasks such as memory management, scaling across multiple processors and more common tasks within the field such as network topology definition, mathematical processing and data pipelining. The two figures below (Zhang *et al.*, 2018b) identify core features of various frameworks as well as potential pros and cons of each framework's utilisation.

**Table 5.** Comparison of deep learning libraries

| Name | Caffe | MXNet | Torch | Deeplearning4j | Tensorflow | Theano | CNTK | Neon | Keras |
|---|---|---|---|---|---|---|---|---|---|
| Creator | UC Berkeley | CMU, UW and Microsot | Ronan Collobert et al. | Skymind | Google | Universite de Montreal | Microsoft | Nervana System | Franois Chollet |
| Interface | C++, Python, MATLAB | C++, R, Python, Scala, Matlab, JavaScript, Go, Julia | Lua, LuaJIT, C | Java, Scala, Clojure | C++, Python, GO, Java | Python | NDL, C++, Python | Python | Python |
| Suitabe model | CNN, RNN | CNN, RNN | DNN, CNN, RNN | DNN, CNN, RNN | DNN, CNN, RNN | DNN, CNN, RNN | CNN, RNN | DNN, CNN, RNN | DNN, CNN, RNN |
| OS | Linux, Win, OSX, Andr. | Linux, Win, OSX, Andr. | Linux, Win, OSX, Andr., iOS | Linux, Win, OSX, Andr. | Linux, OSX, Win | Linux, OSX, Win | Linux, OSX, Win | OSX, Linux | Linux, Win, OSX |
| Stars in github | 20212 | 11170 | 7279 | 7203 | 68800 | 6914 | 12396 | 3200 | 19589 |
| Multi-GPU | ✓ | ✓ | ✓ | ✓ | ✓ | × | ✓ | ✓ | × |
| Distributed | × | ✓ | × | ✓ | ✓ | × | ✓ | ✓ | × |
| Cloud copmuting | × | ✓ | × | × | ✓ | × | × | ✓ | × |

*Notes: In this table, while the various frameworks support different deep learning models, each framework is good for different models. In terms of CNN modeling capabilities, Caffe is the best. In terms of RNN modeling capabilities, CNTK is the best.*

Figure 10: Deep learning framework descriptions and features (Zhang *et al.*, 2018)

**Table 6.** Advantages and disadvantages of some deep learning frameworks

| Name | Advantages | Disadvantages |
|---|---|---|
| TensorFlow | 1. Flexible portability;<br>2. Fast compilation speed;<br>3. Powerful supporting software, such as TensorFlow Serving;<br>4. Strong technical support services;<br>5. Excellent overall architecture. | 1. The documentation and interfaces are insufficiently clear;<br>2. Debugging difficulties;<br>3. High memory footprint. |
| Caffe | 1. Easy to use;<br>2. Training speed is sufficiently fast;<br>3. Highly modular of components. | 1. Support for RNN is not quite adequate;<br>2. Different versions of the interface are not compatible;<br>3. No support for distributed training. |
| Keras | 1. Documentation is complete;<br>2. Easy to learn and easy to use;<br>3. Updates quickly;<br>4. Highly modular of components. | 1. Insufficiently flexible;<br>2. Cannot directly use a multi-GPU. |
| Theano | 1. High level of flexibility;<br>2. Low cost of API interface learning;<br>3. Good computational stability. | 1. It is difficult to learn how to use it;<br>2. It does not have the underlying C++ interface;<br>3. The model is very inconvenient to deploy;<br>4. The debugging error message is very difficult to understand. |
| Torch | 1. Easy to use;<br>2. Highly modular of components;<br>3. It is convenient for use with GPUs;<br>4. The high efficiency of the Lua programming language. | 1. The Lua programming language is not commonly used;<br>2. Torch's data file format is special and needs conversion. |
| MXNet | 1. Good performance;<br>2. High level of flexibility;<br>3. Saves memory;<br>4. Supports many language packages. | 1. Has poor API documentation;<br>2. It is difficult to learn how to use it. |
| CNTK | 1. Very good performance;<br>2. Good scalability. | 1. It is difficult to install;<br>2. It has less learning materials than other frameworks. |

Figure 11: Deep Learning framework advantages and disadvantages (Zhang *et al.*, 2018)

Many of the major players in the Neural Network space such as Tensorflow (Abadi *et al.*, 2016) Microsoft Cognitive Toolkit (formerly CNTK) (Seide, 2016) in the figures above support the Open Neural Network Exchange (ONNX) format created in partnership with Facebook (Singh, 2017), enabling portability of networks for use in other frameworks. In Figure 11 it is stated that CNTK does not support cloud computing. This is inaccurate at the time of writing this report as Microsoft cloud environments (Microsoft, 2018) support the deployment of models and have specific environments which are optimized for these services.

# CHAPTER 3: METHODOLOGY

## 3.1 The Scientific Method: an epistemic view

This section of the report is dedicated to defining the underlying methodology that was implemented to support the creation and execution of experiments designed to explore the research question:

*"*What is the relationship between neural network architecture parameters and the quality of detection of cancer in microscope slide images?*"*

Throughout this section, the author considers the need for a hybrid method featuring elements of scientific and computational experimentation, and software development practices.

## 3.2 Justification

### 3.2.1 Scientific Method – an epistemic view

#### *3.2.1.1 Convenience Experimentation*

Two different approaches to a scientific method leading to the further proposal of "Convenience Experimentation" (Krohs, 2012) are described as experimentation justified by the simple fact that it is convenient to carry out. With the abundance of data and technological solutions for data analytics, the author agrees that this rationale is difficult to disagree with. However, it must be noted that for the research to remain respected by the community. It must continue to adhere to principles and practices which are proven in the field.

#### *3.2.1.2 Hypothesis Driven Research*

When talking about the scientific method, hypothesis driven research (or "hypothetico/deductive reasoning") is what often is being described. This is the process of forming a hypothesis followed by the design and execution of experiments seeking to prove or disprove the hypothesis. A succinct description compares "hypothetico reasoning" as moving from "Ideas to Data", to "Data-driven research" (defined below) that moving from "Data to Ideas" (Kell and Oliver, 2004) . The author of this report recognises that within the application of neural networks, both research methods could be used depending on the learning method and network topology.

#### *3.2.1.3 Data Driven Research*

With reference to "big data" and the impact on scientific research, two key features under the definition of data driven research are identified.

> "*one is the intuition that induction from existing data is being vindicated as a crucial form of scientific inference, which can guide and inform experimental research; and the other is the central role of machines, and thus of automated reasoning, in extracting meaningful patterns from data*" (Leonelli, 2012b)

In the same paper (Leonelli, 2012), when discussing inferences made from data-driven research, highlights that inferences made from data often rely on the solid theoretical principles and therefore should not be considered "hypothesis-free". The proposition that automated reasoning should not "*be seen as a substitute to human judgement based on specific expertise and laboratory experience*" was made. The author agrees with this in principle, particularly from a medical diagnostics point of view, as it is often the case that a diagnosis relies on multiple factors (symptoms/biochemistry) interacting before a decision is reached.

### 3.2.2 Agile Software Development

In 2001 Beck et al, gathered together in a ski-lodge in Utah to discuss the current state of the software development environment. Each member had seen flaws with traditional software development approaches such as Waterfall (Alshamrani and Bahattab, 2015) and were attempting to innovate in the field. The idea behind the gathering was to bring together the best practices from their respective

innovations to try and establish a new foundational approach to subject. The result was the "Agile Manifesto" (Beck, 2001). This consisted of 4 values and 12 simple principles which have impacted software development hugely. The manifesto represented a shift from document heavy, rigid and opaque development practices, which lead to software that didn't represent the true vision of the user's requirements, to a collaborative, user-centric and product focussed methodology, which has the ability to adapt to changing requirements at its heart.

The authors of the manifesto represented a range of frameworks and practices which have since become a family of methodologies under the agile umbrella. With the increasingly technology dependent global economy, the last decade has seen widespread adoption of Agile Methodologies in software houses. However, there are still barriers to adoption. The authors own experience of introducing agile practices in large organisations was one of resistance, as managers with a mentality reliant on documentation and stepwise development processes have struggled to realise the benefits of product focussed iterative design.

#### *3.2.2.1 SCRUM*

Scrum is a development methodology which describes the mentality of the development practice and it's application to complex software development.

"*The heart of Scrum lies in the iteration. The team takes a look at the requirements, considers the available technology, and evaluates its own skills and capabilities. It then collectively determines how to build the functionality, modifying its approach daily as it encounters new complexities, difficulties, and surprises. The team figures out what needs to be done and selects the best way to do it*." (Schwaber, 1995)

The iterations are referred to as "sprints" which provide isolated periods of time for the team to work on specific tasks set by, in the development context, the product owner. At the end of each sprint a review takes place to evaluate the product and provide feedback into the next iteration.

Within a scientific context and experimental approach, particularly with reference to hypothesis driven research (section 3.2.1.2), the process can be perceived as a series of "sprints" with the goal of evolving an experimental design, the result of which may prove or disprove a hypothesis. The "complexity" (Schwaber, 1995) discussed, refers to the aspects of a project that are un-knowable at the start. The scientific discipline operates in the same mentality and it is for this reason that the author believes an iterative approach with, continuous review and modification, should be considered applicable in this context.

#### *3.2.2.2 KANBAN*

Kanban is another methodology under the Agile umbrella which focuses on the prioritisation and execution of tasks in a way that is organised and allows team members to quickly identify bottleneck in the project. The methodology emerged from Japanese manufacturing companies to align manufacturing output with consumption. In the software development world Kanban is often used to match the capacity of tasks to the resources available. (Ohno, 1988)

The use of a virtual Kanban board will assist in prioritising day to day tasks, offering a view of bottlenecks in the project at a glance. This virtual board can also be shared with stakeholders to the project that have an interest in the project progress.

As the author is the only active team member on this project the design of the Kanban board can be tailored to feature task columns allocated to research and queries. This will help to mitigate blocking points in the development lifecycle.

## 3.2.3 Deep Learning Stages

The deep learning process requires the formation of a pipeline through which inputs are fed and outputs are received. There are several intermediate steps which are necessary to ensure that the pipeline is not impeded in any way and that the outputs are as accurate as possible.

It is worth noting at this point that all steps are required for a successful implementation, however some steps may have been simplified or completed by other parties either in the preparation of a data set or in the creation of API libraries within frameworks.

The Cognitive Toolkit by Microsoft (Seide, 2016) has several tutorials made available to the research community designed to encourage use of the toolkit. Within the tutorials, broad categories of the intermediate steps needed to create the pipeline have been outlined. The phases align with the more comprehensive methodological steps (Zhang *et al.*, 2018) and are a sensible methodology for the experiment. These are as follows:

#### *3.2.3.1 Data Acquisition*

This stage focuses on how the data is obtained and fed into the network. It is important to understand the way in which data is ingested to ensure that adequate hardware resources are provisioned and that adequate steps have been taken to limit errors in the data transfer between systems.

#### *3.2.3.2 Data Pre-processing*

##### *3.2.3.2.(a)* "**Data cleaning**"

As mentioned above, one challenge is data quality. Errors in the dataset or missing values can affect the quality of training that takes place. This phase focuses on removing these defects to minimise the number of variables to evaluate when reflecting on the performance of a network.

##### *3.2.3.2.(b)* **"Data Augmentation**"

This stage is designed to address two challenges:
Data volumes - Often datasets are too small to allow for accurate training to take place and so it is necessary to expand the dataset by creating fake data for example. An image can be rotated or transformed to add more variety to the training process ensuring models are better equipped to deal with new data.

##### *3.2.3.2.(c)* **"Normalisation"**

Normalisation is described as limiting or reducing the data in such way that it fits a certain model. Limiting could be by excluding outliers to fit a distribution and reduction refers to the dimensionality of the data. Removing dimensionality can help to focus the model training to certain features within a dataset.

##### *3.2.3.2.(d)* "**Encoding"**

This stage involves encoding the dataset with a simplified label that is recognised by a neural network. An example of label encoding is 'One hot encoding'. This involves providing a label consisting of zeros and ones, which represents one of the labels to be predicted. An example where a network is training to identify a 'cat', 'dog' or 'rabbit' would have the following encodings:
'cat' = 001
'dog' = 010
'rabbit' = 100
This simplifies the output required by the neural network. The length of the encoding is therefore equal to the total number of unique labels to be identified.

#### *3.2.3.3 Model Selection*

It is essential to select the correct type of network topology to train on the dataset. As outlined in the literature review. There is an abundance of network designs but not all designs should be applied to all applications. It is important to understand the different domains within which each network can operate.

#### *3.2.3.4 Model Training*

This is the process whereby the labelled dataset is passed into the neural network to train the individual nodes on features of the data. This will involve weightings being assigned to the nodes as the data passes through, defining activation on the node the next time it is exposed to similar data.

#### *3.2.3.5 Performance Evaluation*

With a functioning network now trained on the dataset, performance evaluation can now take place. This process looks at how well the network is responding to new data and if it is not, query why this might be the case. The label now acts as comparator from which performance metrics are created. A series of options and visualisations have been created to assist in evaluating performance.

#### *3.2.3.5.(a)* Review the prediction outputs

This data provides clues regarding the model's ability to produce any form of prediction. This model is a binary classifier and when subjected to new data, the model, if not trained correctly may decide that every sample is cancerous or none of them are. If the dataset is evenly split between cancer and normal tissue images this could result in a success rate of 50% simply because 50% of the dataset happens to be one of these classes.

#### *3.2.3.5.(b)* Review the images and associated predictions

This information would produce an insight into the dataset itself. The researcher would be able to view the images for data quality issues. This is useful for both training and testing. With a large dataset it is not always possible to review every image separately and if sampled at random it is possible that the model has been trained with poor quality images. With the example of microscope slides, it is possible that whitespace (created by the backlight of the microscope passing through the glass slide) could be identified as a significant feature during training. If it is present in enough of the training images. This could be a feature in either of the classes so may confuse a model leading to inaccurate predictions. This process could be broken down further to look at images of each class that were either successfully or unsuccessfully classified. So that the researcher can review a batch of images for similar features.

#### *3.2.3.5.(c)* Review the confidence of the neural network's prediction

Every prediction made comes with a percentage probability showing how 'confident' the model was in the prediction made compared to the alternative class. This can be displayed as a distribution across the entire dataset, a subset of classification or when reviewing the individual images. Images which are either classified correctly or incorrectly with a low confidence can be reviewed to see features which may be common to both, offering an insight into the network's decision making.

#### *3.2.3.5.(d)* Review the proportions of classifications taking place

This shows how well each class has been learned by the network. It may be the case that the network has managed to learn about more features of one class than the other and as a result copes better with predictions for that class. By reviewing the proportions, a researcher may be able to address this issue by reviewing the balance of the number of samples from each class that are being used to train the network.

#### *3.2.3.5.(e)* Breaking into the black box

This involves getting under the hood of the neural network, looking at the feature maps produced by the convolution kernels at different layers. The output of each feature map highlights the activity at different nodes. This assists the researcher in further understanding what the network might be focusing on during decision making.

#### *3.2.3.5.(f)* Attention/Focus heatmaps

By using the feature maps in the hidden nodes, it is possible to create a heatmap of the pixels that the network is focusing on. When this is used as an overlay for the original image it can be a useful tool

in identifying the features of an image that appear to be important to the neural network at a particular layer.(Jetley *et al.,* 2018) (Liu *et al.,* 2016)

#### *3.2.3.5.(g)* Quantitative methods

In line with other research areas regarding binary classification performance (Saito and Rehmsmeier, 2015) several classification statistics can be derived (see section 2.11.5) from the confusion matrix data to provide information regarding the performance of particular classes during classification. This data has featured in other research regarding image classification ((Fraz *et al.,* 2012) ,(Welikala *et al.,* 2017), (Angel Cruz-Roa, 2014)).

## 3.3 Description and implementation plan

As previously stated, the author considers a hybrid approach of software development and scientific method to be essential to the rapid development and execution of experiments to support the research question. The term "Agile Science" is not new. Artefacts of such a method have been described in the context of behavioural change (Hekler *et al.,* 2016). This research project aspires to be an example of "Agile Science" in action by utilising iterative evolutionary cycles, with continuous review and feedback by stakeholders, enabling progression towards the aims and objectives outlined above.

The research was planned out as a series of "sprints" (Schwaber, 2000) .Every sprint had the same format:

- Research – allowing time to gain a further understanding of the theme
- Write Report – Research and Analyses needs to be maintained and updated throughout the project.
- Code – This strikes the balance between documentation and a working product necessary for successful implementation
- Test – Each step must be tested to ensure that the implementation is progressing.
- Review – A critical review must be undertaken to ensure that the project is still progressing towards achieving the aims and objectives. Feedback is built in to inform the next sprint or iteration.

The early sprints are designed to be shorter and more intensive, aimed around the setup of the environment and exploration of the data. Including tasks that reduce complexity and allow for quick wins early on helps to embed the researcher in their work instilling a good work ethic and motivation for longer sprints. The deep learning stages involving data were ideal candidates for these shorts sprints as they were specific, involve the researcher with the domain, and pave the way for the later themes.

The longer sprints or "epics" (Schwaber, 2000) are designed for the iterative development/experimentation cycles. It is important to recognise that research of this nature does not simply involve the analysis of variables. It is also the design and construction of the apparatus and performance metrics for experimental evaluation. This adds a level of complexity to the sprints which requires an iterative approach, as the researcher takes each factor from an 'unknown' to a 'known' state. The "epic" decomposes into a series of sprints iterating around a single theme, whether it is the development of the network design or parameter tuning within that architecture.

The sprints have been organised into themes. The theme of each sprint is in-line with the phases outlined by the CNTK tutorial modules (Seide, 2016). These broad phases align with, or aggregate, the deep learning stages defined above (Zhang *et al.,* 2018) (section 3.2.3). These are suitable for project organisation and are easily consumable for stakeholders not fully familiar with data science/neural network parlance.

As the project progressed, more requirements became evident and as a result, it was necessary to quickly prioritise the new tasks to achieve them. This is where Kanban is strongest as it facilitates the arrangement and visualisation of requirements and task progression.

## 3.3.1 Data Set

This research project used a freely available open dataset of IDC Microscope slide images that has been prepared and provided by a Kaggle user (Mooney, 2018). Kaggle is a data scientist community website, which allows users to create challenges, demonstrate their projects and share knowledge between each other. The data set had originally formed part of a study also exploring the use of deep learning (Angel Cruz-Roa, 2014) . Each image is a 50x50 pixel section of a Haematoxylin and Eosin (H&E) stained slide. The images have been separated by the presence/absence of Breast Cancer IDC. (Note: Patient details have been anonymised prior to the release of the data to the public). The original images were 100x100 so some compression for the published dataset has taken place which may result in a reduction in performance output.

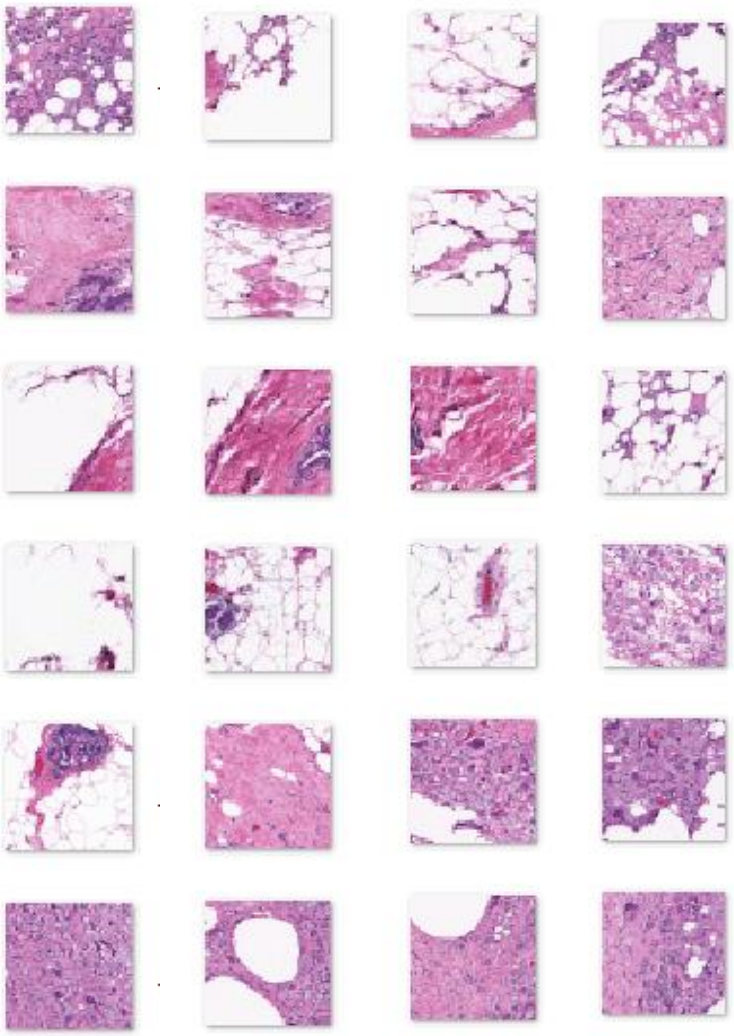


Figure 12: Examples of slide images

#### *3.3.1.1 Dataset Testing Accuracy*

The original dataset obtained an F-score of 71% and Balance Accuracy of 82% (Angel Cruz-Roa, 2014).The compressed version of the dataset obtained >90% accuracy during testing (Mooney, 2018). The focus of this report aims to achieve a high degree of accuracy but must bare these benchmarks in mind when considering the performance of the network relative to the field.

## 3.3.2 Technology

#### *3.3.2.1 Jupyter Notebook*

A tool used by data scientists to design and conduct experiments involving data analytics and artificial intelligence. The interface allows a user to write reports with embedded code which can be executed by a built-in interpreter. This ensures that experiments can be re-created, and the results observed in real-time by readers of the report.

#### *3.3.2.2 Python*

Python is used in many frameworks aimed at data pre-processing and construction of deep learning models. Python is one of the primary languages used by data scientists due to the availability of dedicated libraries for data manipulation and visualisation.

#### *3.3.2.3 Microsoft Cognitive Toolkit (CNTK)*

This library has been developed by Microsoft to assist data scientists with the development of AI models. The APIs allow for communication with lower level frameworks, providing a level of abstraction which increases the accessibility to artificial intelligence to those who are new to the discipline.

#### *3.3.2.4 GPU*

During the early stages of the project, experiments created will be run on machines with CPUs, as the budget for the project is limited. If positive results are observed, it could be possible to move to a GPU architecture. This is a budget/efficiency trade-off which will be reviewed as the research progresses.

# CHAPTER 4: EXPERIMENTS

## 4.1 Experimentation Plan

A Convolutional Neural Network (LeCun, 1992) design was selected for the purposes of image classification. Using an iterative approach to network parameter tuning, the experiment aimed to identify the presence or absence of invasive ductal carcinoma as accurately as possible. The focus of the analysis lies within the parameter tuning itself to attempt to establish which parameters have the greatest impact on the accuracy.

### 4.1.1 Experimental Stages

#### *4.1.1.1 Stage 1 – Iterative Cycle - Getting the network to learn*

By iterating through sample sizes and learning rates the author attempted to find a combination which resulted in a network performance greater than random change to show that the model was in fact detecting features. This forms part of the optimisation of the "convergence" and "gradient descent" design criterion (Adya and Collopy, 1998)

The adjustments were made with increasing granularity where positive results were observed. This method allowed for faster convergence around parameters which produced better results. As this stage was dealing in a lot of unknowns the performance evaluation was less fined grained. For each iteration, the moving average of the training and test data was retained and plotted on a graph to show performance relative to other iterations. A confusion matrix and bitmap showing class prediction were used to provide an at-a-glance view of the performance at each iteration. This allowed the researcher to quickly identify models which were overpredicting to certain classes.

#### *4.1.1.2 Stage 2 – Predictions In depth*

The best performing parameters were retested with more detailed performance evaluation features to provide insights into the 'decision making' process of the network. The confidence of the network predictions and visualisations relating the network's attention were used. If some parameter combinations were found to be more reliable, other values in the same region were tested to ensure that further opportunities for optimisation were not being missed. This has been performed to test the "generalisation" evaluation criterion (Adya and Collopy, 1998)

#### *4.1.1.3 Stage 3 – Evaluation with a larger dataset*

With the network options further narrowed down, the best performing models were tested with a larger dataset to further validate the network performance. This stage was performed to test the "generalisation" and "stability" evaluation criterion (Adya and Collopy, 1998)

#### *4.1.1.4 Stage 4 – Network model parameters*

This stage involved adjusting network model parameters such as the convolution kernel dimensions and number of kernel filters to see the effects on the accuracy and outputs in the performance evaluation. This stage conforms to the "network architecture" design criterion (Adya and Collopy, 1998).

### 4.1.2 Data Feed

Sample Size – This was included in the iterative cycle observe the impact on the accuracy of the classification. The issue of overfitting may be highlighted by this parameter.

For stage one, sample sizes between 100 and 9100 were used for training, 5000 for testing and another 5000 for evaluation. No samples from any set featured in another set.

For stage two, a selection of the best sample sizes and learning rate from stage 1 were used for training. 25000 samples were used for testing to increase the validation of the model, and the same 5000 for evaluation were used to ensure consistency in the further analysis.

For stage three, the best performing sample sizes were carried forward from stage two for training. The training set featured 25000 samples and the evaluation was expanded to 62000. This value was suitable for the memory resources.

For stage four, the evaluation set initially used 5000 samples but where higher performing models were identified the larger evaluation set of 62000 samples was used.

## 4.1.3 Hyperparameter Tuning

#### *4.1.3.1 Learning rate*

This will be included in the iterative cycle in stage one as it is essential for establishing a model that is capable of generalising across multiple datasets. This will be used in combination with varying sampling sizes to find effective combinations.

#### *4.1.3.2 Assessing the Performance*

- Percentage Error - Used throughout the experiment
- Graphing performance – Used throughout the experiment
- Visualising image data and predictions - Included throughout the experiment
- Confusion Matrix – Included throughout the experiment
- Prediction confidence - Only used in stages 3 and beyond
- Attention heatmaps – Only used in stages 3 and beyond. 5 samples selected at random by the researcher. The same samples were used to show comparative results across different parameter tunings.
- Quantitative metrics – Used from stage 2 onwards

#### *4.1.3.3 Iterating over the network topology*

- Feature Maps – Tested in stage four
- Convolution dimensions – Tested in stage four
- Number of Hidden Layers – fixed for this experiment
- Activation functions – fixed for this experiment
- Pooling Functions – fixed for this experiment

Note: parameters that were fixed for this experiment were chosen based on CNTK tutorials provided as part of the installation and certain parameters such as convolution dimensions were based on the work of the publisher of the dataset (Mooney, 2018).

# 4.2 Experimental Design

## 4.2.1 Environment Setup

The following code shows the initial imports for python and CNTK. A device variable is set in case the Jupyter Notebook is deployed in Microsoft Azure Notebooks (Microsoft, 2018) to detect the network settings. A fixed random seed has also been set for the testing of our model to ensure consistent sampling between experiments. Note: A lot of the code regarding construction of the network and data ingestion has been informed by CNTK tutorials and documentation (CNTK, n.d.) made available on installation.

```
from __future__ import print_function # Use a function definition from future version (say 3.x from 2.7 interpreter)
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
import os
import sys
import time

import cntk as C
import cntk.tests.test_utils
cntk.tests.test_utils.set_device_from_pytest_env() # (only needed for our build system)
C.cntk_py.set_fixed_random_seed(1) # fix a random seed for CNTK components

%matplotlib inline
```

Figure 13: initial imports for environment setup

## 4.2.2 Data Acquisition & Pre-processing

The code below deals with the reading of the dataset into the neural network. The create_reader function takes a text file with links to the dataset in the following format:

[full path to File][filename] \<Tab> [Category Label]

In this function, any transformations or augmentation of the data are also defined. For the purposes of this experiment the only transformation present ensures that the data that is streamed is scaled to fit the training model. The images are all 50x50 pixels RGB images and so the transformation matches these dimensions. The num_classes variable represents the number of classes that can be predicted and these are one hot encoded by CNTK.

The remainder of the function takes the file and prepares the data and the number of output classes. These are passed to the CNTK streaming function which creates the features array (the dataset) and the labels in the correct format for the neural network.

```
# model dimensions
image_height = 50
image_width  = 50
num_channels = 3
num_classes  = 2

import cntk.io.transforms as xforms
#
# Define the reader for both training and evaluation action.
#
def create_reader(map_file, train):
    print("Reading map file:", map_file)

    # transformation pipeline for the features when training
    transforms = []

    transforms += [
        xforms.scale(width=image_width, height=image_height, channels=num_channels)

    ]
    # deserializer
    return C.io.MinibatchSource(C.io.ImageDeserializer(map_file, C.io.StreamDefs(
        features = C.io.StreamDef(field='image', transforms=transforms), # first column in map file is referred to as 'image'
        labels   = C.io.StreamDef(field='label', shape=num_classes)      # and second as 'label'
    )))
```

Figure 14: Reader and data transformation definition

The code in Figure 15 ensures that the datasets are available in the specified directories, returning an error if the files are not found. It is necessary to ensure that the data is accessible in the correct directory specified.

```
# Ensure the training and test data is available.


data_found=False # A flag to indicate if train/test data found in local cache
for data_dir in [os.path.join("..", "Tutorials", "data")]:

    train_file=os.path.join(data_dir, "train_mapv3.txt")
    test_file=os.path.join(data_dir, "test_mapv3.txt")
    eval_file=os.path.join(data_dir, "eval_test_mapv3.txt")


    if os.path.isfile(train_file) and os.path.isfile(test_file):
        data_found=True
        break

if not data_found:
    raise ValueError("Please generate the data")

print("Data directory is {0}".format(data_dir))
```

```
Data directory is ..\Tutorials\data
```

Figure 15: Accessing the image data for training, testing and evaluation

## 4.2.3 Model Selection

The next step deals with the model definition. The model that has been selected is a convolutional neural network (LeCun, 1992). The code below defines the layers of the network and the parameters which dictate the behaviour of the layer.
The high-level layout of the network is:

- Input
- Convolutional Layer - defines the kernels which will the features of initial input
- Max Pooling - a compression layer taking the max value of the feature maps created
- Convolutional Layer - a second layer of feature extraction
- Max Pooling - a second layer of compression
- Dense Layer - handling the output of the predicted class probabilities

```
x = C.input_variable(input_dim_model)
y = C.input_variable(num_output_classes)
```

Slide Type -

```
def create_model(features):
    with C.layers.default_options(init = C.layers.glorot_uniform(), activation = C.relu):
            h = features

            h = C.layers.Convolution2D(filter_shape=(3,3),
                                       num_filters=8,
                                       strides=(1,1),
                                       pad=True, name='first_conv')(h)
            h = C.layers.MaxPooling(filter_shape=(2,2),
                                    strides=(1,1), name="first_max")(h)
            h = C.layers.Convolution2D(filter_shape=(3,3),
                                       num_filters=24,
                                       strides=(1,1),
                                       pad=True, name='second_conv')(h)
            h = C.layers.MaxPooling(filter_shape=(2,2),
                                    strides=(1,1), name="second_max")(h)
            r = C.layers.Dense(num_output_classes, activation = None, name="classify")(h)
            return r
```

Figure 16: CNN layer definition

To ensure that there are no errors in the configuration the shape of the first layer, output biases of the last layer of network and the number of parameters are printed. Two biases are displayed which at this point are both zero as no training has yet taken place. The network can also be visualised as highlighted in figure 18:

```
# Create the model
z = create_model(x)

# Print the output shapes / parameters of different components
print("Output Shape of the first convolution layer:", z.first_conv.shape)
print("Bias value of the last dense layer:", z.classify.b.value)
```

```
Output Shape of the first convolution layer: (8, 50, 50)
Bias value of the last dense layer: [0. 0.]
```

Slide Type -

```
# Number of parameters in the network
C.logging.log_number_of_parameters(z)
```

```
Training 112570 parameters in 6 parameter tensors.
```

Figure 17: verifying the model structure and output

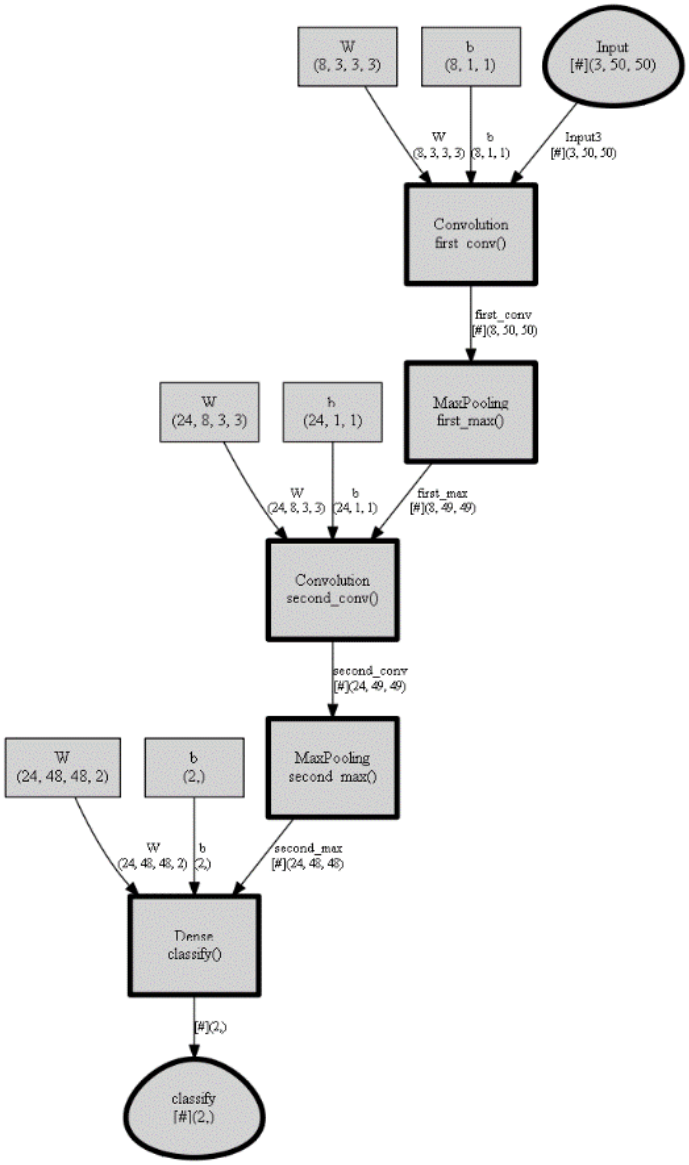



Figure 18: Neural network visualisation

The following code contains helper functions such as defining the loss and error calculations for the network, a moving average function, and displaying feedback on training progress.

```
def create_criterion_function(model, labels):
    loss = C.cross_entropy_with_softmax(model, labels)
    errs = C.classification_error(model, labels)
    return loss, errs # (model, labels) -> (loss, error metric)
```

Slide Type -

```
# Define a utility function to compute the moving average sum.
def moving_average(a, w=5):
    if len(a) < w:
        return a[:]    # Need to send a copy of the array
    return [val if idx < w else sum(a[(idx-w):idx])/w for idx, val in enumerate(a)]


# Defines a utility that prints the training progress
def print_training_progress(trainer, mb, frequency, verbose=1):
    training_loss = "NA"
    eval_error = "NA"

    if mb%frequency == 0:
        training_loss = trainer.previous_minibatch_loss_average
        eval_error = trainer.previous_minibatch_evaluation_average
        if verbose:
            print ("Minibatch: {0}, Loss: {1:.4f}, Error: {2:.2f}%".format(mb, training_loss, eval_error*100))

    return mb, training_loss, eval_error
```

Figure 19: Helper functions

### 4.2.4 Configure Model Training

with the model now configured, the training parameters can be defined. The function below takes the output from the reader function for a training and test dataset, passes the data through the model. Some important features in the code are:

- num_sweeps_to_train_with - how many passes over the same dataset are taken.
- loss and label_error - tracks the progress of the training
- lr_schedule - the learning rate hyperparameter defining the co-efficient which dictates how large a step to take during gradient descent.
- learner - identifies the learning optimisation function. In this case it is stochastic gradient descent built into the CNTK library.
- trainer - builds the trainer with the parameters specified.
- minibatch_size - defines how many data samples to pass to the trainer at a time.
- num_samples_per_sweep - defines how many samples to draw from
- input_map - formatting the features and labels created from the reader for ingestion into the trainer
- A for loop to run the trainer for the model for each minibatch. Output of error and moving average for evaluation
- A for loop for testing with output of error and moving average for evaluation
- A plotting function to create a visualisation of the error in prediction relative to a random choice.

With all these parameters established it was now possible to proceed with training the model on the dataset followed by performance evaluation and begin our experimentation with iterative parameter tuning.

```
def train_test(train_reader, test_reader, model_func, lr , sz, num_sweeps_to_train_with=1):

    # Instantiate the model function; x is the input (feature) variable
    # We will scale the input image pixels within 0-1 range by dividing all input value by 255.
    model = model_func(x/255)

    # Instantiate the loss and error function
    loss, label_error = create_criterion_function(model, y)

    # Instantiate the trainer object to drive the model training
    learning_rate = lr
    lr_schedule = C.learning_parameter_schedule(learning_rate)
    learner = C.sgd(z.parameters, lr_schedule)
    trainer = C.Trainer(z, (loss, label_error), [learner])

    # Initialize the parameters for the trainer
    minibatch_size = 50 #50 is a good minibatch size
    num_samples_per_sweep = sz
    num_minibatches_to_train = (num_samples_per_sweep * num_sweeps_to_train_with) / minibatch_size

    # Map the data streams to the input and labels.
    input_map={
        y  : train_reader.streams.labels,
        x  : train_reader.streams.features
    }

    # Uncomment below for more detailed logging
    training_progress_output_freq = 100

    # Start a timer
    start = time.time()
    global minib
    minib=[]
    evalerr=[]
    global iter_error_list_train
    for i in range(0, int(num_minibatches_to_train)):
        # Read a mini batch from the training data file
        data=train_reader.next_minibatch(minibatch_size, input_map=input_map)
        trainer.train_minibatch(data)
        print_training_progress(trainer, i, training_progress_output_freq, verbose=1)
        minib+=[i]
        evalerr+=[trainer.previous_minibatch_evaluation_average]

    movingAv = moving_average(evalerr, w=5)
    iter_error_list_train += [[movingAv]]

    # Print training time
    print("Training took {:.1f} sec".format(time.time() - start))
```

Figure 20: training and testing function definition

```
# Test the model
test_input_map = {
    y  : test_reader.streams.labels,
    x  : test_reader.streams.features
}

# Test data for trained model
test_minibatch_size = 50
num_samples = 25000
num_minibatches_to_test = num_samples // test_minibatch_size

test_result = 0.0

minib2=[]
evalerr2=[]

for i in range(num_minibatches_to_test):
    minibat=i
    # We are loading test data in batches specified by test_minibatch_size
    data = test_reader.next_minibatch(test_minibatch_size, input_map=test_input_map)
    eval_error = trainer.test_minibatch(data)
    test_result = test_result + eval_error
    minib2+=[i]
    evalerr2+=[eval_error]

movingAv2 = moving_average(evalerr2, w=5)


# Average of evaluation errors of all test minibatches
print("Average test error: {0:.2f}%".format(test_result*100 / num_minibatches_to_test))

def plot_progress(minib, evalerr, movingAv, num_minibatches, title):
    plt.plot(minib, evalerr, color='orange', label='Progress')
    plt.plot(minib, movingAv, markevery=3, color='blue', label='Moving Average')
    plt.plot([0, num_minibatches], [0.5, 0.5], color='darkred', linestyle='--', label='Random Choice')
    plt.xlabel('Minibatch')
    plt.ylabel('Evaluation Error')
    plt.title(title)
    plt.legend()
    plt.show()
    plt.pause(0.5)


#print(minib)
#print(evalerr2)

plot_progress(minib,evalerr, movingAv, num_minibatches_to_train, 'Prediction Error During Training')
plot_progress(minib2,evalerr2, movingAv2, num_minibatches_to_test, 'Prediction Error During Testing')
```

Figure 21: function definition with progress visualisation

## 4.2.5 Model Training

The function below takes reader function outputs of the training and test files and passes them to the train_test function to run the dataset through the neural network. The output shows the training progress at the predefined minibatch frequency. The average test error across the test dataset and graphs showing the error rate during both training and testing are displayed. Whilst these are important to understand the performance of the network, further work is required in order to understand why these results are being observed. This is the rationale behind the performance evaluation step taken.

```
def do_train_test():
    global z
    z = create_model(x)
    reader_train = create_reader(train_file, True)
    reader_test = create_reader(test_file, False)
    train_test(reader_train, reader_test, z)

do_train_test()
```

Figure 22: model training and testing initialisation

Printing the bias values in the last layer allow the user to see whether the training outputs are working correctly. The outputs from the model are then passed into a softmax function to bring the prediction values between 0 and 1 to be used as an output for our dataset.

```
print("Bias value of the last dense layer:", z.classify.b.value)
```

```
Bias value of the last dense layer: [-0.01090398  0.01090398]
```

```

```

```
out = C.softmax(z)
```

Figure 23: confirmation of trained outputs and softmax function

### *4.2.5.1 Stage 1: Iterative Cycle code adaptation*

During this stage it was necessary to create the functionality to quickly cycle through many different values for the sample size and learning rate parameters. To facilitate the this, the test_train and do_test_train functions were amended to accept parameters to the function. These were then generated in for-loops to iterate through various permutations.

```
def train_test(train_reader, test_reader, model_func, lr , sz, num_sweeps_to_train_with=1):

    # Instantiate the model function; x is the input (feature) variable
    # We will scale the input image pixels within 0-1 range by dividing all input value by 255.
    model = model_func(x/255)

    # Instantiate the loss and error function
    loss, label_error = create_criterion_function(model, y)

    # Instantiate the trainer object to drive the model training
    learning_rate = lr
    lr_schedule = C.learning_parameter_schedule(learning_rate)
    learner = C.sgd(z.parameters, lr_schedule)
    trainer = C.Trainer(z, (loss, label_error), [learner])

    # Initialize the parameters for the trainer
    minibatch_size = 50 #50 is a good minibatch size
    num_samples_per_sweep = sz
    num_minibatches_to_train = (num_samples_per_sweep * num_sweeps_to_train_with) / minibatch_size
```

Figure 24: Modification of the train/test function for iterative testing

Global variables were used within the function so that data could be retained for comparisons with other iterations. In this example the minibatch numbers, evaluation errors and moving averages of the error rate for, in this case, evaluation of the test data. The same process was used to output a value for the training data. This data could then be plotted relative to other iterations featuring different permutations of sample size and learning rate.

```
global minib2

evalerr2=[]
minib2=[]
for i in range(num_minibatches_to_test):
    minibat=i
    # We are loading test data in batches specified by test_minibatch_size
    data = test_reader.next_minibatch(test_minibatch_size, input_map=test_input_map)
    eval_error = trainer.test_minibatch(data)
    test_result = test_result + eval_error
    minib2+=[i]
    evalerr2+=[eval_error]
global iter_error_list
movingAv2 = moving_average(evalerr2, w=5)
iter_error_list += [[movingAv2]]
```

Figure 25: global variables created for model evaluation

The amended do_triain_test function handles the iteration permutations of sample size and learning which are defined by the researcher and passed to the train_test function to initialise and test the deep learning model. For each sample size the error rates and ROC curves for each learning rate are plotted using the functions defined. Global variables are initialised for data capture.

```
def do_train_test(lr, sz):
    global z
    z = create_model(x)
    reader_train = create_reader(train_file, True)
    reader_test = create_reader(test_file, False)
    train_test(reader_train, reader_test, z, lr, sz)

eval_table = []
eval_index = []

for n in range(8100, 8200,200): #Sample size range
    iter_error_list = []
    iter_error_list_train = []
    iter_i = []
    iter_n = []
    minib2 = []
    minib = []
    FalsePR = []
    TruePR=[]
    Gthresholds = []
    successes=[]
    successesCancer=[]
    successesNormal=[]
    failures=[]
    failuresCancer=[]
    failuresNormal=[]
    predicted_label_prob=[]
    gtlabel=[]
    pred=[]
    img_data=[]
    print("-----------------------------------------------")
    print("------------Sample Size: " , n, "-----------")
    print("-----------------------------------------------")
    for i in range(22, 23, 5): #Learning rate range
        sz = n
        iter_i += [i/10000]
        iter_n += [n]

        #Initialise the training and testing
        do_train_test(i/10000, n)
        print("Bias value of the last dense layer:", z.classify.b.value)
        out = C.softmax(z)
        evaluationBitmapAndConfusion(i/10000,n)

    #plot_all_errors(minib, iter_error_list_train, iter_n, iter_i, "Training for sample size: " + str(n))
    #plot_all_errors(minib2, iter_error_list, iter_n, iter_i, "Testing for sample size: " + str(n))
    plot_roc(iter_n, iter_i)
```

Figure 26: code for iterative testing and model evaluation

```
def plot_all_errors(minib, iter_err_list, iter_n, iter_i, title):
    # Calculate chart area size
    leftmargin = 6 # inches
    rightmargin = 6 # inches

    figwidth = leftmargin + rightmargin
    plt.figure(figsize=(figwidth, figwidth))
    for p in range(len(np.array(iter_err_list))):
        plt.plot(minib, iter_err_list[p][0], markevery=3, label=str(iter_n[p]) + " : " + str(iter_i[p]))
    plt.plot([0, len(minib)], [0.5, 0.5], color='darkred', linestyle='--', label='Random Choice')
    plt.xlabel('Minibatch')
    plt.ylabel('Evaluation Error')
    plt.title(title)
    plt.legend()
    plt.show()
    plt.pause(0.5)

def plot_roc(iter_n, iter_i):
    global FalsePR
    global TruePR
    global Gthresholds

    leftmargin = 4 # inches
    rightmargin = 4 # inches

    figwidth = leftmargin + rightmargin
    plt.figure(figsize=(figwidth, figwidth))
    plt.plot([0, 1], [0, 1], linestyle='--', label='Random Choice')
    # plot the roc curve for the model
    for i in range(len(FalsePR)):
        plt.plot(FalsePR[i][0], TruePR[i][0], marker='.', label=str(iter_n[i]) + " : " + str(iter_i[i]))
    # show the plot
    plt.legend()
    plt.title("ROC Curve")
    plt.show()
```

Figure 27: plotting error rates and ROC curves for iterative testing

## 4.3 Performance Evaluation

As the experiment progressed, new methods of performance evaluation were developed and deployed at different stages depending on the successes of the previous evaluation. As an example, a network that was overpredicting was not appropriate for reviewing images individually. However, with all methods deployed, seeing more advanced tools such as attention heatmaps applied to potentially overfitted training sets could be useful, so experiments may have been re-run to include these evaluation methods.

The code below tests the neural network on a new dataset. As before, a reader is created for the data. The data is then formatted as an object output which contains the labels and features of the data. The image data is reshaped so that it can be plotted, displaying the image. This img_data variable is then passed to the neural network for evaluation returning the predictions for each image.

```
# Read the data for evaluation
reader_eval=create_reader(test_file, False)

eval_minibatch_size = 5000
eval_input_map = {x:reader_eval.streams.features, y:reader_eval.streams.labels}

data = reader_eval.next_minibatch(eval_minibatch_size, input_map=eval_input_map)

img_label = data[y].asarray()
img_data = data[x].asarray()

# reshape img_data to: M x 3 x 50 x 50 to be compatible with model
img_data = np.reshape(img_data, (eval_minibatch_size, 3, 50, 50))

predicted_label_prob = [out.eval(img_data[i]) for i in range(len(img_data))]
#print(predicted_label_prob)
```

```
Reading map file: ..\Tutorials\data\test_mapv3.txt
```

Figure 28: passing the evaluation dataset through the model for prediction

The data below extracts the prediction and associated label for each image based on the highest probability of the output for each class with each image.

```
# Find the index with the maximum value for both predicted as well as the ground truth
pred = [np.argmax(predicted_label_prob[i]) for i in range(len(predicted_label_prob))]
gtlabel = [np.argmax(img_label[i]) for i in range(len(img_label))]
```

Figure 29: image label and prediction output for comparison

### 4.3.1 Review the prediction outputs

Three visualisations have been produced by the code below to provide an insight into the predictions made by the network. The code creates a series of lists of sample id's based on:

- Number of successful Cancer Diagnoses
- Number of successful Normal Diagnoses
- Number of unsuccessful Cancer Diagnoses
- Number of unsuccessful Normal Diagnoses

A separate calculation produces a percentage of correct answers. The first visualisation is a bitmap showing the spread of class predictions across the evaluation dataset. This provides a view of how well the neural network generalises. This does not give any view on the success of the predictions, simply whether the model is attempting to make predictions.

```
import math
from sklearn.metrics import confusion_matrix

a=5000

np_pred= np.array(pred)

np_pred = np_pred.reshape(50,100)
#print(np_pred.shape)

np_pred = np.multiply(np_pred, 255)
plt.figure(0)
plt.title('Predictions as a bitmap showing density of classes')
plt.axis('off')
plt.imshow(np_pred, cmap="gray")
plt.pause(0.5)


#print("Label    :", gtlabel[:a])
#print("Predicted:", pred[:a])
successes=[]
successesCancer=[]
successesNormal=[]
failures=[]
failuresCancer=[]
failuresNormal=[]
overall=[]
sample=0
test_result = 0
for i in range(len(pred)):
    sample=i
    if (gtlabel[i] == pred[i]):
        test_result+=1
        successes+=[sample]
        if(pred[i] == 1):
            successesCancer+=[sample]
            overall+=[255]
        else:
            successesNormal+=[sample]
            overall+=[150]
    else:
        failures+=[sample]

        if(pred[i] == 1):
            failuresNormal+=[sample]
            overall+=[75]
        else:
            failuresCancer+=[sample]
            overall+=[0]
```

Figure 30: Performance evaluation metrics

## 4.3.2 Review the proportions of classifications taking place

The second visualisation is a bitmap showing different colour values based on the successes or failures of predictions with each class. Each row in the image is a minibatch with each pixel mapped to a prediction. This shows the density of successful predictions.
The third visualisation is a confusion matrix showing the proportions of the successful and unsuccessful predictions for each class. The colour coding corresponds to the colour coding in the second bitmap visualisation, offering a simplified view of the accuracy of the model. Each of these class predictions are also described in the printout of the categories mentioned above.

```
np_overall= np.array(overall)

np_overall = np_overall.reshape(50,100)
#print(np_overall.shape)

plt.figure(1)
plt.title('Predictions as a bitmap of successes and failures')
plt.axis('off')
plt.imshow(np_overall, cmap="cool")
plt.pause(0.5)

print("Number Of Successful Cancer Diagnoses: ", len(successesCancer))
print("Number Of Successful Normal Diagnoses: ", len(successesNormal))
print("Number Of unsuccessful Cancer Diagnoses: ", len(failuresCancer))
print("Number Of unsuccessful Normal Diagnoses: ", len(failuresNormal))

print("percentage of correct answers: {0:.2f}%".format(test_result/len(pred)*100))

cm = confusion_matrix(gtlabel,pred)

# Calculate chart area size
leftmargin = 1 # inches
rightmargin = 1 # inches
categorysize = 1 # inches
classNames = ['Normal','Cancer']

figwidth = leftmargin + rightmargin + (len(classNames) * categorysize)
plt.figure(figsize=(figwidth, figwidth))
plt.imshow(cm, interpolation='nearest', cmap='cool')

plt.title('Cancer Diagnosis Confusion Matrix - Eval Data')
plt.ylabel('True label')
plt.xlabel('Predicted label')
tick_marks = np.arange(len(classNames))
plt.xticks(tick_marks, classNames, rotation=45)
plt.yticks(tick_marks, classNames)
s = [['TN','FP'], ['FN', 'TP']]
for i in range(2):
    for j in range(2):
        plt.text(j,i, str(s[i][j])+" = "+str(cm[i][j]),horizontalalignment="center",)
plt.show()
```

Figure 31: creating a confusion matrix

### *4.3.2.1 Stage 1: Iterative Cycle code adaptation*

The code above was combined under performance evaluation and turned into a function to be called after each iteration to perform the evaluation.

```
import math
from sklearn.metrics import confusion_matrix

def evaluationBitmapAndConfusion(lr, sz):
    # Read the data for evaluation
    reader_eval=create_reader(test_file, False)

    eval_minibatch_size = 5000
    eval_input_map = {x:reader_eval.streams.features, y:reader_eval.streams.labels}

    data = reader_eval.next_minibatch(eval_minibatch_size, input_map=eval_input_map)

    img_label = data[y].asarray()
    img_data = data[x].asarray()

    # reshape img_data to: M x 3 x 50 x 50 to be compatible with model
    img_data = np.reshape(img_data, (eval_minibatch_size, 3, 50, 50))

    predicted_label_prob = [out.eval(img_data[i]) for i in range(len(img_data))]
    #print(predicted_label_prob)

    # Find the index with the maximum value for both predicted as well as the ground truth
    pred = [np.argmax(predicted_label_prob[i]) for i in range(len(predicted_label_prob))]
    gtlabel = [np.argmax(img_label[i]) for i in range(len(img_label))]

    np_pred= np.array(pred)

    np_pred = np_pred.reshape(50,100)
    #print(np_pred.shape)
```

Figure 32: Modification for iterative testing

Global variables were created to retain data from each iteration so that the permutations could be compared find the highest performers. The example below outputs the various performance metrics for each iteration. This was then converted into a table for comparison. The table was then exported to a csv file to be manipulated and analysed externally.

```
if(TP != 0 and FN != 0 and FP !=0 and TN != 0):
    #Recall: Out of all the positive classes, how many instances were identified correctly.
    Recall = TP / (TP + FN)
    #Precision: Out of all the predicted positive instances, how many were predicted correctly.
    Precision = TP / (TP + FP)

    #F-Score: From Precision and Recall. F-score is the harmonic mean of Precision and Recall.
    FScore = (2 * Recall * Precision) / (Recall + Precision)

    #True Positive Rate = True Positives / (True Positives + False Negatives)
    #Sensitivity = True Positives / (True Positives + False Negatives)
    TPR = TP/(TP + FN)

    #False Positive Rate = False Positives / (False Positives + True Negatives)
    FPR = FP/(FP + TN)

    #Specificity = True Negatives / (True Negatives + False Positives)
    Spec = TN/(TN + FP)

    FPR2 = 1 - Spec

probs=[]
for i in range(len(pred)):
    probs+=[predicted_label_prob[i][0][pred[i]]]

# calculate AUC
auc = roc_auc_score(gtlabel, np.array(probs).astype('float'))
print('AUC: %.3f' % auc)

# calculate roc curve
fpr, tpr, thresholds = roc_curve(gtlabel, probs)

global FalsePR
global TruePR
global Gthresholds

FalsePR += [[fpr]]
TruePR += [[tpr]]
Gthresholds += [[thresholds]]

global eval_table
global eval_index
eval_index += [[lr,sz]]
eval_table += [[len(successesCancer),len(successesNormal),len(failuresCancer),len(failuresNormal),
                test_result/len(pred)*100, Recall, Precision, FScore, TPR, FPR, Spec, auc]]
```

Figure 33: Performance metric calculation and collation

```
import numpy as np
import pandas as pd
# Creating a 2 dimensional numpy array
data= np.array(eval_table).astype(float)

ind = []

for g in range(len(eval_index)):
    eval_ind = str(eval_index[g][1]) + " - LR = " + str(eval_index[g][0])
    ind += [eval_ind]

#Creating pandas dataframe from numpy array
dataset = pd.DataFrame(data[:,[0,1,2,3,4,5,6,7,8,9,10,11]].astype(float), index = ind[:], columns = ['True Cancer',
                       'True Normal',
                       'False Cancer',
                       'False Normal',
                       'Accuracy',
                       'Recall',
                       'Precision',
                       'F-Score', 'TPR/Sensitivity', 'FPR', 'Specificity','AUC'])
print(dataset)

dataset2 = pd.DataFrame(data[:,4],index = ind[:], columns = ['% Correct Answers'])
print(dataset2)

export_csv = dataset.to_csv(r'C:\Users\g_osm\CNTK-Samples-2-6\Tutorials\Eval_Matrix.csv')
export_csv = dataset2.to_csv(r'C:\Users\g_osm\CNTK-Samples-2-6\Tutorials\AccuracyBySampleSizeAndLR.csv')
```

Figure 34: Metrics stored in a dataframe and output to a csv file for external analysis

## Review the images and associated predictions

The code below allows a user to review the images and the associated predictions in the evaluation dataset. a series of functions are defined to output different results based on user selection of successful and unsuccessful predictions or a sample range for review. A confidence threshold can also be defined. The makeImage function reconstructs the original image array by re-organising the pixel values from the img_data variable. This is then converted to a numpy array and plotted to provide the image as an output.

```
# construct an image
def makeImage(sample_number):
    nparr = np.array(img_data[sample_number])
    nparr3=[]
    nparr4=[]
    nparr5=[]
    for i in range(50):
        nparr4=[]
        for j in range(50):
            nparr4 += [[nparr[0][i][j],nparr[1][i][j],nparr[2][i][j]]]
        nparr3+=[nparr4]
    return nparr3

def showRange(sample_lb, sample_ub, confidence=100):
    print("-------------------------------------------------------------------------------------------------------------------")
    print("-------------------------------------------------------------------------------------------------------------------")
    print("Showing Samples: ", sample_lb," to ", sample_ub-1, " with confidence <= ", confidence, "%")
    for b in range(sample_lb, sample_ub):
        if(predicted_label_prob[b][0][np.argmax(predicted_label_prob[b])]*100 <= confidence):
            nparrn = np.array(makeImage(b))
            showSlide(nparrn, b)

def showSuccesses(sample_lb, sample_ub, confidence=100):
    print("-------------------------------------------------------------------------------------------------------------------")
    print("-------------------------------------------------------------------------------------------------------------------")
    print("Showing Successes: ", sample_lb," to ", sample_ub-1, " with confidence <= ", confidence, "%")
    for b in successes[sample_lb:sample_ub]:
        if(predicted_label_prob[b][0][np.argmax(predicted_label_prob[b])]*100 <= confidence):
            nparrn = np.array(makeImage(b))
            showSlide(nparrn, b)

def showFailures(sample_lb, sample_ub, confidence=100):
    print("-------------------------------------------------------------------------------------------------------------------")
    print("-------------------------------------------------------------------------------------------------------------------")
    print("Showing Failures: ", sample_lb," to ", sample_ub-1, " with confidence <= ", confidence, "%")
    for b in failures[sample_lb:sample_ub]:
        if(predicted_label_prob[b][0][np.argmax(predicted_label_prob[b])]*100 <= confidence):
            nparrn = np.array(makeImage(b))
            showSlide(nparrn, b)


def showSuccessesCancer(sample_lb, sample_ub, confidence=100):
    print("-------------------------------------------------------------------------------------------------------------------")
    print("-------------------------------------------------------------------------------------------------------------------")
    print("Showing successes for cancer: ", sample_lb," to ", sample_ub-1, " with confidence <= ", confidence, "%")
    for b in successesCancer[sample_lb:sample_ub]:
        if(predicted_label_prob[b][0][np.argmax(predicted_label_prob[b])]*100 <= confidence):
            nparrn = np.array(makeImage(b))
            showSlide(nparrn, b)
```

Figure 35: Displaying evaluated images and predictions

### 4.3.3 Review the confidence of the neural network's prediction

Following the plotting of images for review, a series of graphs have been created that show the distribution of probabilities representing the 'confidence' of the predictions. This shows a researcher the volume of the dataset that is not easily differentiable. With this a researcher can gain a better understanding of the dataset in terms of quality as well as the accuracy of the model

```
def showFailuresNormal(sample_lb, sample_ub, confidence=100):
    print("-----------------------------------------------------------------------------------------------------------------------")
    print("-----------------------------------------------------------------------------------------------------------------------")
    print("Showing failures for normal: ", sample_lb," to ", sample_ub-1, " with confidence <= ", confidence, "%")
    for b in failuresNormal[sample_lb:sample_ub]:
        if(predicted_label_prob[b][0][np.argmax(predicted_label_prob[b])]*100 <= confidence):
            nparrn = np.array(makeImage(b))
            showSlide(nparrn, b)


def showSlide(nparrn, b):
    print("-----------------------------------------------------------------------------------------------------------------------")
    print("-----------------------------------------------------------------------------------------------------------------------")
    print("Image Sample Number: ", b)
    plt.figure(b)
    plt.imshow(nparrn.astype(np.uint8))
    plt.show
    plt.pause(0.5)
    img_gt, img_pred = gtlabel[b], pred[b]
    print("Image Label: ", img_gt)
    print("Image Prediction: ", img_pred)
    print("Confidence of Prediction: ", round(predicted_label_prob[b][0][np.argmax(predicted_label_prob[b])]*100,2),"%")
    print("Confidence of Alternative: ", round(predicted_label_prob[b][0][np.argmin(predicted_label_prob[b])]*100, 2),"%")


#user defines sample upper and lower bounds and the maximum confidence of probability to be listed.
sample_lb = 0
sample_ub = 50
confidence = 100 #find less confident values

#The commented functions below are to be used as a switch for users to display different images.

#showSuccesses(sample_lb, sample_ub, confidence)
#showSuccessesCancer(sample_lb, sample_ub, confidence)
#showSuccessesNormal(sample_lb, sample_ub, confidence)

#showFailures(sample_lb, sample_ub, confidence)
#showFailuresCancer(sample_lb, sample_ub, confidence)
#showFailuresNormal(sample_lb, sample_ub, confidence)

#showRange(sample_lb, sample_ub, confidence)
```

Figure 36: Image display function and switches for the image output functions

```
histplotSC = []
histplotSN = []
histplotFC = []
histplotFN = []

for p in successesCancer:
    histplotSC+=[round(predicted_label_prob[p][0][np.argmax(predicted_label_prob[p])]*100,2)]

for q in successesNormal:
    histplotSN+=[round(predicted_label_prob[q][0][np.argmax(predicted_label_prob[q])]*100,2)]

for r in failuresCancer:
    histplotFC+=[round(predicted_label_prob[r][0][np.argmax(predicted_label_prob[r])]*100,2)]

for s in failuresNormal:
    histplotFN+=[round(predicted_label_prob[s][0][np.argmax(predicted_label_prob[s])]*100,2)]

np_histplotSC = np.array(histplotSC)
np_histplotSN = np.array(histplotSN)
np_histplotFC = np.array(histplotFC)
np_histplotFN = np.array(histplotFN)

#print(np_histplotSC.shape)
#print(np_histplotSN.shape)
#print(np_histplotFC.shape)
#print(np_histplotFN.shape)

plt.figure(0)
plt.hist(np_histplotSC,bins=5)
plt.title('Successful Cancer Prediction Confidence Distribution')
plt.show
plt.figure(1)
plt.hist(np_histplotSN,bins=5)
plt.title('Successful Normal Prediction Confidence Distribution')
plt.show
plt.figure(2)
plt.hist(np_histplotFC,bins=5)
plt.title('Unsuccessful Cancer Prediction Confidence Distribution')
plt.show
plt.figure(3)
plt.hist(np_histplotFN,bins=5)
plt.title('Unsuccessful Normal Prediction Confidence Distribution')
plt.show
```

Figure 37: distributions of prediction probabilities for the evaluation dataset

### 4.3.4 Neural Network model visualisation

The cell below deals with visualising the network layers. The numbers of weights and biases are also present in the network diagram.

```
import pydot_ng

stuff = C.logging.graph.plot(z, 'model.png')
```

```
from IPython.display import Image

display(Image(filename="model.png"))
```

Figure 38: code to create the network topology visualisation

### 4.3.5 Breaking into the black box

The code below defines functions which allow a user to pass an image through the network for classification. As this happens, the feature maps in the two convolutional hidden layers are extracted so that a researcher can see what features are being focused on. This is performed by the eval_and_write function. The name of the node is taken as a parameter so that different layers can be extracted. The function testIndividualImage passes the image to be tested and stitches the output feature maps (a series of matrices of pixels) together to be displayed as a single image. These are plotted with the pixels scaled as a greyscale image.

```
from cntk import load_model, combine
import cntk.io.transforms as xforms
from cntk.logging import graph
from cntk.logging.graph import get_node_outputs

def eval_and_write(node_name, output_file, minibatch_source, num_objects):
    # Load model and pick desired node as output

    node_in_graph = z.find_by_name(node_name)
    output_nodes  = combine([node_in_graph.owner])

    # evaluate model and get desired node output
    #print("Evaluating model for output node %s" % node_name)
    features_si = minibatch_source['features']

    for i in range(0, num_objects):
        mb = minibatch_source.next_minibatch(1)
        output = output_nodes.eval(mb[features_si])

    return output

def makeImage2(img_dat,sample_number):
    nparr = np.array(img_dat[sample_number])
    nparr3=[]
    nparr4=[]
    nparr5=[]
    for i in range(50):
        nparr4=[]
        for j in range(50):
            nparr4 += [[nparr[0][i][j],nparr[1][i][j],nparr[2][i][j]]]
        nparr3+=[nparr4]
    return nparr3
```

Figure 39: outputting the feature maps for the convolutional layer

```
def TestIndividualImage(image):
    reader_eval2=create_reader(image, False)

    eval_minibatch_size = 1
    eval_input_map = {x:reader_eval2.streams.features, y:reader_eval2.streams.labels}

    data = reader_eval2.next_minibatch(eval_minibatch_size, input_map=eval_input_map)

    img_lab = data[y].asarray()
    img_dat = data[x].asarray()

    # reshape img_data to: M x 3 x 50 x 50 to be compatible with model
    img_dat = np.reshape(img_dat, (eval_minibatch_size, 3, 50, 50))

    predicted_lab_prob = [out.eval(img_dat[i]) for i in range(len(img_dat))]
    #print(predicted_label_prob)

    # Find the index with the maximum value for both predicted as well as the ground truth
    predi = [np.argmax(predicted_lab_prob[i]) for i in range(len(predicted_lab_prob))]
    gtlab = [np.argmax(img_lab[i]) for i in range(len(img_lab))]

    print("-----------------------------------------------------------------------------------------------------------------------")
    print("-----------------------------------------------------------------------------------------------------------------------")
    print("Showing Samples:", image)
    nparrn = np.array(makeImage2(img_dat, 0))
    print("-----------------------------------------------------------------------------------------------------------------------")
    print("-----------------------------------------------------------------------------------------------------------------------")
    print("Image Sample: ", image)
    plt.figure(0)
    plt.title('Original Image')
    plt.imshow(nparrn.astype(np.uint8))
    plt.show
    plt.pause(0.5)
    img_gt, img_pred = gtlab[0], predi[0]
    print("Image Label: ", img_gt)
    print("Image Prediction: ", img_pred)
    print("Confidence of Prediction: ", round(predicted_lab_prob[0][0][np.argmax(predicted_lab_prob[0])]*100,2),"%")
    print("Confidence of Alternative: ", predicted_lab_prob[0][0][np.argmin(predicted_lab_prob[0])]*100,"%")

    image_height = 50
    image_width  = 50
    num_channels = 3

    # use this to get the features of a layer
    node_name = "second_conv"
    node_name2 = "first_conv"
    output_file = ""#os.path.join(base_folder, "layerOutput.txt")

    # evaluate model and write out the desired layer output
    minibatch_source = create_reader(image, False)
    lfarr = eval_and_write(node_name, output_file, minibatch_source, num_objects=1)
    lfarr2= eval_and_write(node_name2, output_file, minibatch_source, num_objects=1)
```

Figure 40: Testing an image to produce feature maps

```
#conv2 layer feature maps
nplfarr = np.array(lfarr).astype(np.uint8)
#print(nplfarr[0][0].shape)

#stitching the feature maps together to form a 4x6 (RowsxCols) set of images
newlfarr= nplfarr[0][0]
for i in range(1,6):
        newlfarr= np.concatenate((newlfarr, nplfarr[0][i]),axis=1)
#print(newlfarr.shape)

newlfarr2= nplfarr[0][6]
for i in range(7,12):
       newlfarr2= np.concatenate((newlfarr2, nplfarr[0][i]),axis=1)
#print(newlfarr2.shape)

newlfarr3= nplfarr[0][12]
for i in range(13,18):
        newlfarr3= np.concatenate((newlfarr3, nplfarr[0][i]),axis=1)
#print(newlfarr3.shape)

newlfarr4= nplfarr[0][18]
for i in range(19,24):
        newlfarr4= np.concatenate((newlfarr4, nplfarr[0][i]),axis=1)
#print(newlfarr4.shape)

newlfarr5=np.concatenate((newlfarr,newlfarr2,newlfarr3,newlfarr4))
#print(newlfarr5.shape)

#conv1 layer feature maps
nplfarr2 = np.array(lfarr2).astype(np.uint8)
#print(nplfarr2[0][0].shape)
#print(nplfarr2.shape)

newlfarr6= nplfarr2[0][0]
for i in range(1,4):
    newlfarr6= np.concatenate((newlfarr6, nplfarr2[0][i]),axis=1)
#print("here",newlfarr6.shape)

newlfarr7= nplfarr2[0][4]
for i in range(5,8):
    newlfarr7= np.concatenate((newlfarr7, nplfarr2[0][i]),axis=1)
#print(newlfarr7.shape)

#conv1 layer feature maps plotted
newlfarr8=np.concatenate((newlfarr6,newlfarr7))
plt.figure(2)
plt.title('conv1 layer feature maps plotted')
plt.imshow(newlfarr8, cmap="gray")
plt.show
plt.pause(0.5)
```

Figure 41: stitching the feature maps together to create a single image

### 4.3.6 Attention/Focus heatmaps

The final part of the testIndividualImage function performs element-wise addition of the matrices of the 24 feature maps to form a heatmap, representing regions of focus for the image. The heatmaps are plotted and assigned a colour intensity range to assist the researcher in identifying these regions. The original image being tested is then used as an overlay with a transparency to allow the heatmaps to be seen in context with the image.

```
#conv2 layer feature maps plotted
plt.figure(1)
plt.title('conv2 layer feature maps plotted')
plt.imshow(newlfarr5, cmap="gray")
plt.show
plt.pause(0.5)

#create heatmap by adding up all feature maps from conv2 layer
newlfarr9= nplfarr[0][0]
for i in range(1,24):
        newlfarr9= newlfarr9 + nplfarr[0][i]


#feature heatmap one
plt.figure(3)
plt.title('feature heatmap one')
plt.imshow(newlfarr9, cmap="hot")
plt.colorbar()
plt.show
plt.pause(0.5)

#feature heatmap two
plt.figure(4)
plt.title('feature heatmap two')
plt.imshow(newlfarr9, cmap="winter")
plt.colorbar()
plt.show
plt.pause(0.5)

#original image
plt.figure(5)
plt.title('original image')
plt.imshow(nparrn.astype(np.uint8))
plt.show
plt.pause(0.5)
img_gt, img_pred = gtlab[0], predi[0]

#heatmap with image overlay
plt.figure(6)
plt.title('heatmap one with image overlay')
img3 = plt.imshow(newlfarr9,cmap='winter')
img2 = plt.imshow(nparrn.astype(np.uint8), alpha=0.75)
plt.show
plt.pause(0.5)
img_gt, img_pred = gtlab[0], predi[0]

#heatmap two with image overlay
plt.figure(7)
plt.title('heatmap two with image overlay')
img3 = plt.imshow(newlfarr9,cmap='hot')
img2 = plt.imshow(nparrn.astype(np.uint8) ,alpha=0.75)
plt.show
plt.pause(0.5)
img_gt, img_pred = gtlab[0], predi[0]
```

Figure 42: creation and displaying of attention heatmaps

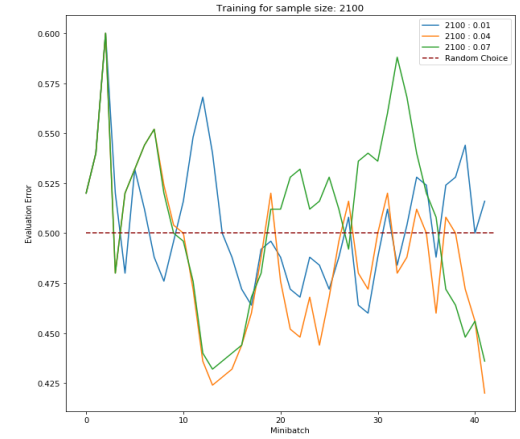
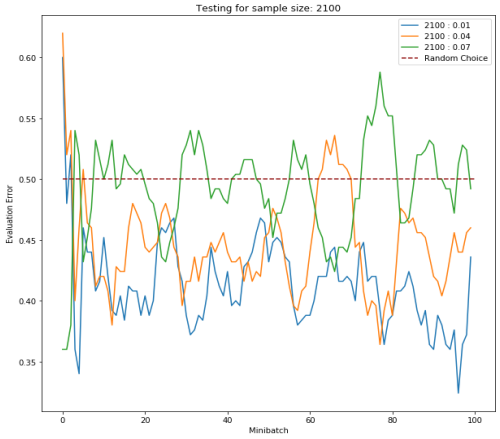
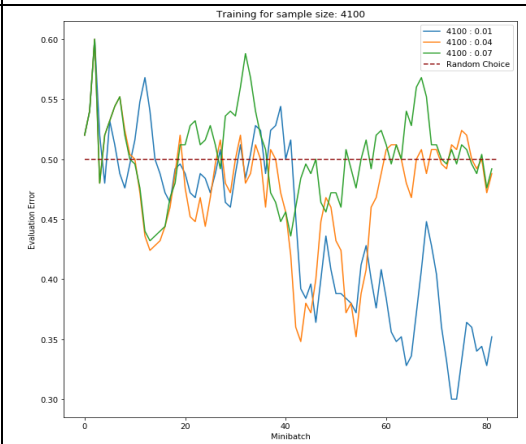
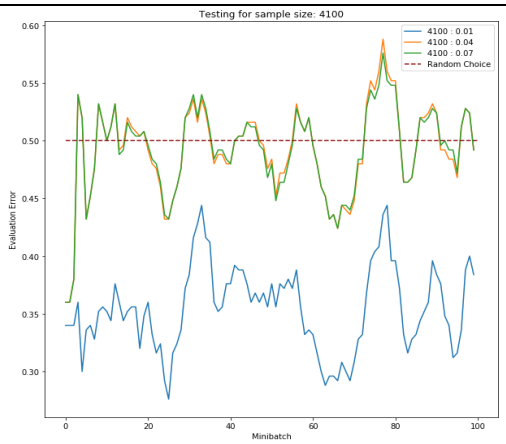
# 4.4 Experimentation Results

## 4.4.1 Stage 1

Through Iterating across a wide range of sample sizes and learning rates, graphs were created to show the performance of the training and test data at different iterations. The plots are moving averages of the error (not the accuracy) to show the performance across different minibatches of data input. The training graphs aim to show a decline in the error where the testing aims to show consistency in the error for validation of the stability of the model. If the tests frequently jump into the range of random choice (50% error) then there is little evidence to suggest that the network is attempting to classify correctly.

Figure: shows the training and across a broad range sample sizes with the learning rate values fixed at 0.01, 0.04 and 0.07. The sample sizes are then repeated at a lower order of magnitude with learning rate values fixed 0.01, 0.04, 0.07 to see if and how the performance changes. A broader range of tests is available for review in Appendix I (section 8.1).

### *4.4.1.1 Iterative Cycle 1 – Sample sizes used in training at different Learning Rates in the x10^-2 range*

| Sample Size | Moving Average Error Rate -Training | Moving Average Error Rate -Testing |
|---|---|---|
| 2100 |  |  |
| 4100 |  |  |

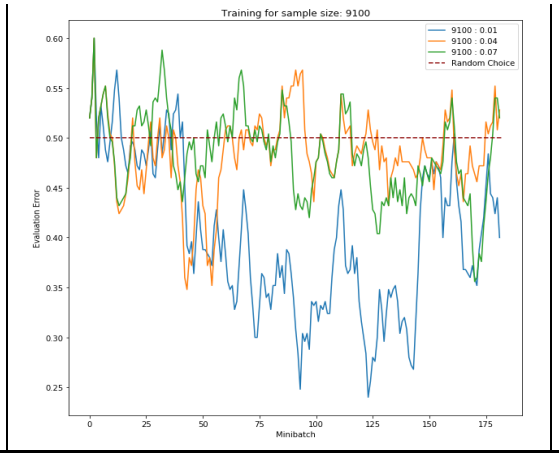
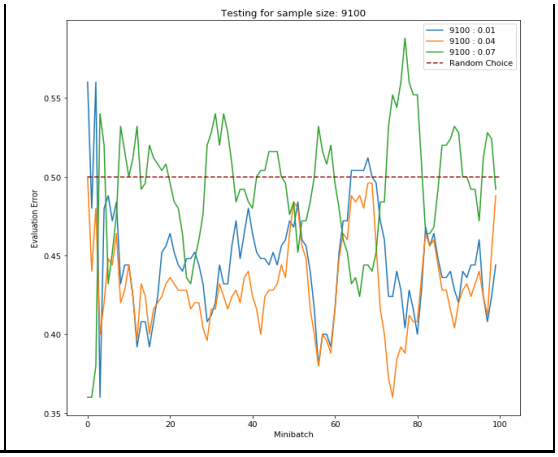
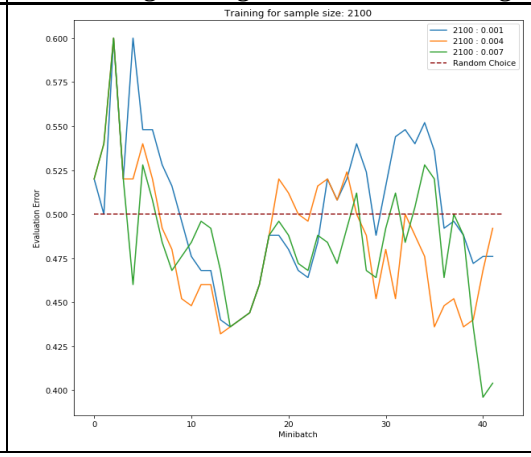
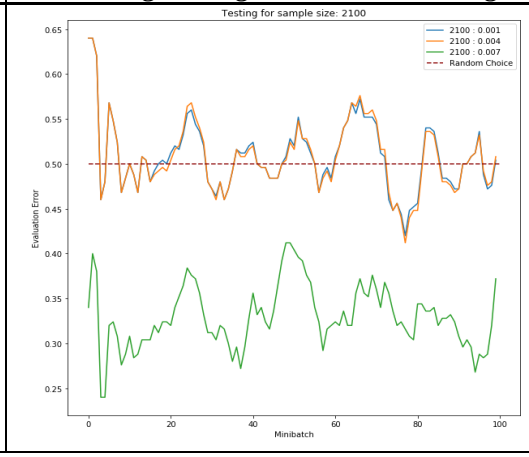
| | | |
|---|---|---|
| 9100 |  |  |

Table 1: Table 1: Iterative cycle testing of sample size and learning rate 10^-2

#### *4.4.1.1.(a)* analysis

The data above shows fairly poor performance at the learning rates 0.04 and 0.07 (green and orange) in terms of training and testing with frequent intercepts with random chance. These do not appear to provide meaningful training. However, at 4100 samples and a learning rate of 0.01 there is a distinct drop in the error rate as the training progresses. During testing, the average error is consistently below random chance suggesting there is some form of 'learning' taking place. Much larger reactions in the error are observed at the extreme ranges of sample sizes suggesting that there is a balance to be struck in this regard.

### *4.4.1.2 Iterative Cycle 2 – Sample sizes used in training at different Learning Rates in the x10^-3 range*

| Sample Size | Moving Average Error Rate -Training | Moving Average Error Rate -Testing |
|---|---|---|
| 2100 |  |  |

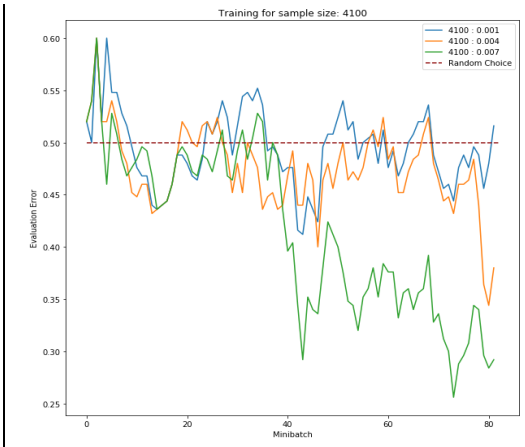
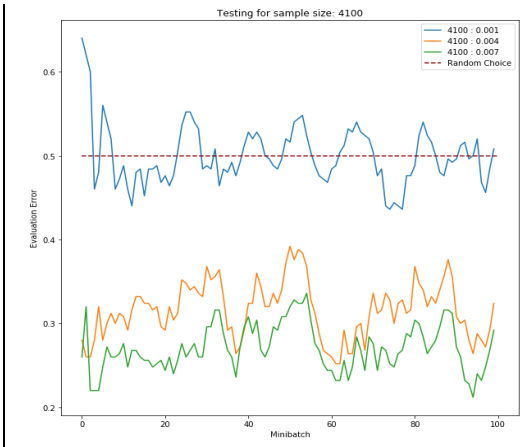
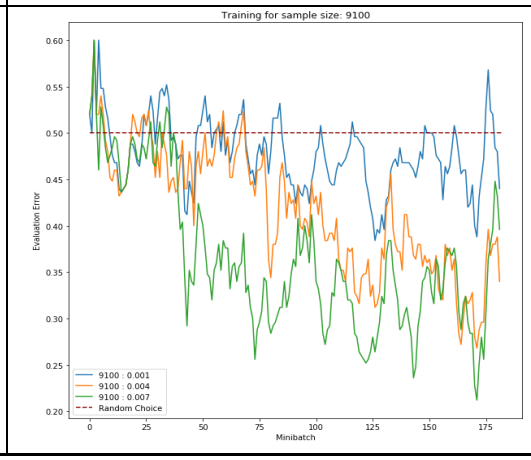
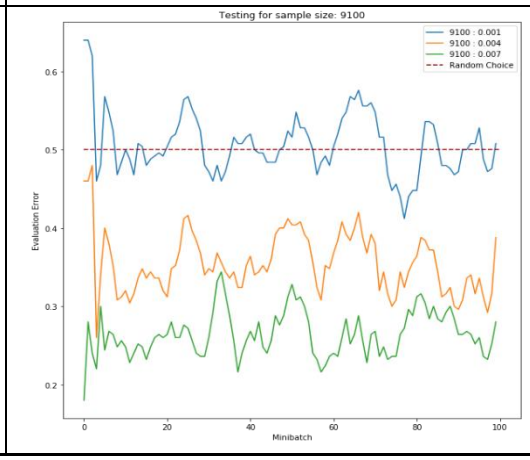
| | | |
|---|---|---|
| 4100 |  |  |
| 9100 |  |  |

Table 2: Iterative cycle testing of sample size and learning rate (10^-3)

#### *4.4.1.2.(a)* Analysis

Broadly speaking, the lower order of magnitude (10^-3) of learning rate seems to have a more positive effect on the training of this dataset than the previous magnitude (10^-2). However, it must be noted that at LR = 0.001 (Blue) poor training and test results are apparent as the average bounces around the region of random choice. Between sample sizes it was again noteworthy that the extremities showed less stable performance compared to the middle ground (size = 4100). This demonstrates the need to for both the learning rate and sample size to be tested rigorously to identify a balance.

It is important to understand that these graphs simply illustrate that the model is learning to classify images without explaining how the model is performing these classifications.

#### *4.4.1.3 Stage 1 macro analysis*

During each iteration the model was evaluated against a separate dataset to see how well it was able to generalise when classifying new data. The performance accuracy and distributions of classifications were captured and plotted to highlight combinations of sample size and learning rate that was obtained. Figure 43 shows each of the iterations plotted against the percentage of correct answers. The raw data analysis can be found in Appendix II (section 8.2)

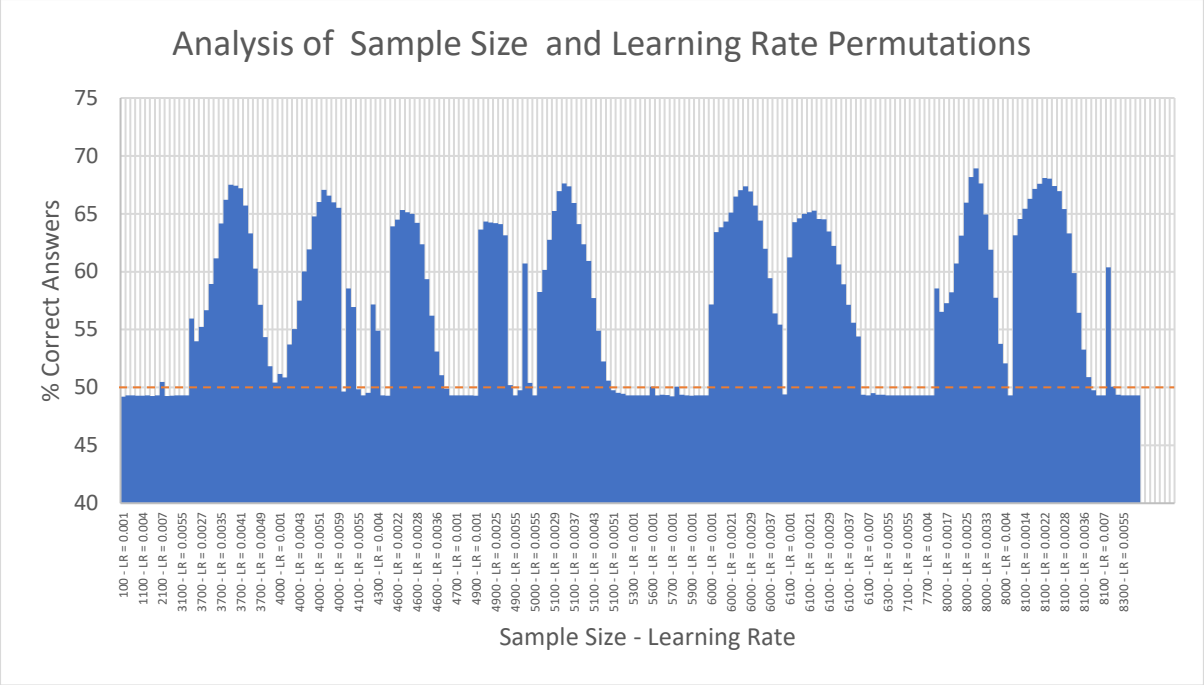



Figure 43: Graph to show accuracies of different sample sizes and learning rates for comparison

There are 9 distinct peaks on the graph showing areas where the training has converged on a model capable of generalising beyond a random choice when subjected to new data for classification. To ensure that these peaks are truly representative of a good prediction, the data was then plotted as a stacked bar chart showing the proportions of successful and unsuccessful classifications. This provided some insight into whether the model was generalising successfully across both classes and not just due to a model with a bias towards a particular class. The results are visualised in Figure 44.

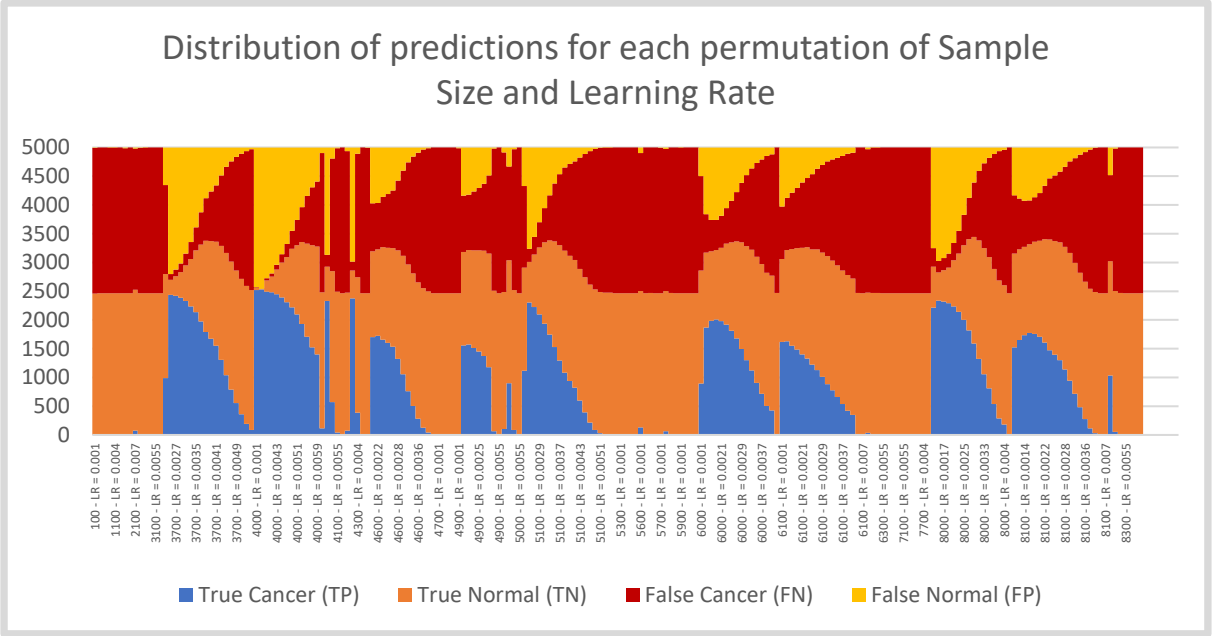



Figure 44: distribution of successful and unsuccessful for different sample sizes and learning rates

The objective in this visualisation is to minimise unsuccessful regions (red and yellow regions) whilst ensuring that there is a balance between the successful regions (blue and orange regions). This shows that the model is attempting to evaluate correctly across the dataset and, as a result, proves that the 9

peaks observed in Figure 44 show regions where the unsuccessful categorisations are being minimised by the model. This visualisation is effectively a method of representing a series of confusion matrices in a novel way proposed by the author.

With these regions identified the next stage was to look at some of the more successful models in greater detail, to identify any potential issues with the way in which it was 'deciding' which classification to assign to an image. As a starting point, the sample size/learning rate values for the models with greatest accuracy across the 9 peaks were identified for testing. If an evaluated model is identified as outperforming other models in different heuristics, other values within the peak could be tested.

The maximum accuracies across the 9 peaks are presented below:

| Sample Size - Learning Rate | Avg Error - Initial Testing (%) | Evaluation Accuracy (%) |
|---|---|---|
| 8000 - LR = 0.0029 | 39.66 | 68.92 |
| 8100 - LR = 0.0022 | 36.28 | 68.1 |
| 5100 - LR = 0.0033 | 36.69 | 67.62 |
| 3700 - LR = 0.0039 | 35.94 | 67.5 |
| 6000 - LR = 0.0027 | 36.09 | 67.36 |
| 4000 - LR = 0.0053 | 39.57 | 67.06 |
| 4600 - LR = 0.0024 | 35.86 | 65.32 |
| 6100 - LR = 0.0023 | 33.00 | 65.28 |
| 4900 - LR = 0.0022 | 35.28 | 64.34 |

Table 3: Models with the highest accuracy identified in the iterative cycle testing

## 4.4.2 Stage 2

This stage took the 9 maximum values identified in stage 1 for closer examination. For each test case the code for granular analysis of the model's performance was executed. Creating visualisations such as a confusion matrix, class density bitmap and feature attention heatmaps, the researcher's goal at this stage was to use these heuristics to rank the 9 cases according to the performance in the criteria specified.

### *4.4.2.1 Training vs Testing performance*

When reviewing the moving average error for the different test cases during training it was necessary to take into account whether the model was deviating from the line of a random choice and where this was the case, how stable the model was when exposed to new data. If the line is too close to random choice it would suggest that when tested over a large number of samples the accuracy may tend towards 50%, Examples are presented below, and the complete range is available in Appendix III (section 8.3):

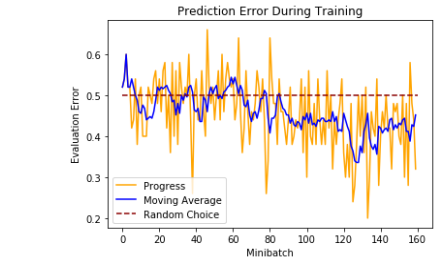
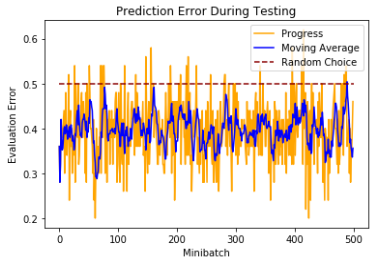
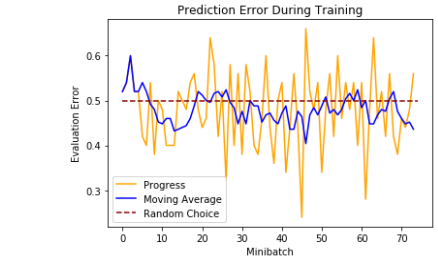
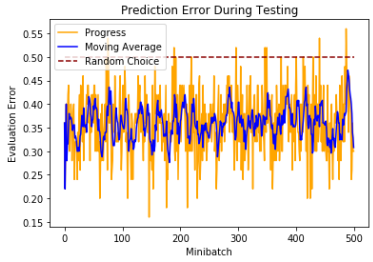
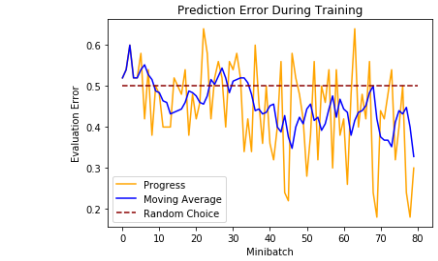
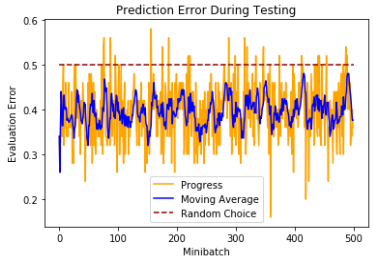
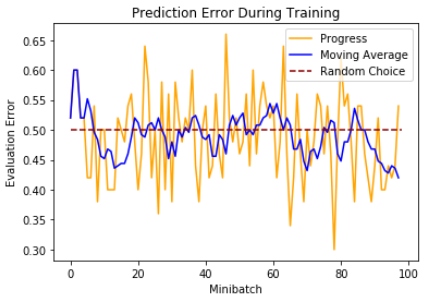
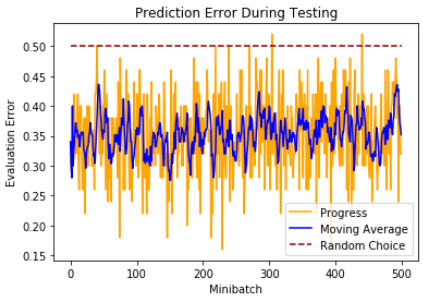
| Training Sample Size and Learning Rate | Training | Testing (25000 Samples) |
|---|---|---|
| 8000 - LR = 0.0029 |   |   |
| 3700 - LR = 0.0039 |   |   |
| 4000 - LR = 0.0053 |   |   |
| 4900 - LR = 0.0022 |   |   |

Figure 45: Examples of minibatch error rates during training and testing

#### *4.4.2.1.(a)* Analysis

Reviewing the moving average allowed the researcher to quickly identify models which performed better than others during training and testing. It was worth noting that models which did not appear to train well still obtained results better than a coin flip. This may be evidence to suggest that a larger testing set is required to see if the model accuracy holds or if it eventually tends towards random choice.

### *4.4.2.2 Evaluation of Model Confusion Matrix and Class Density Bitmap (5000 samples)*

For the confusion matrices, the true positive rate (recall), true negative rate, precision and F-Score were calculated. The difference between the recall and true negative rate was then obtained in order to rank the test cases. A small difference between the two classes was evidence of a stronger ability to differentiate between the two. This was an indication that the model was not overpredicting for one particular class. This data can be viewed qualitatively from the bitmap and confusion matrix colour intensities. As illustrated below:

**Over predicting to normal class:**

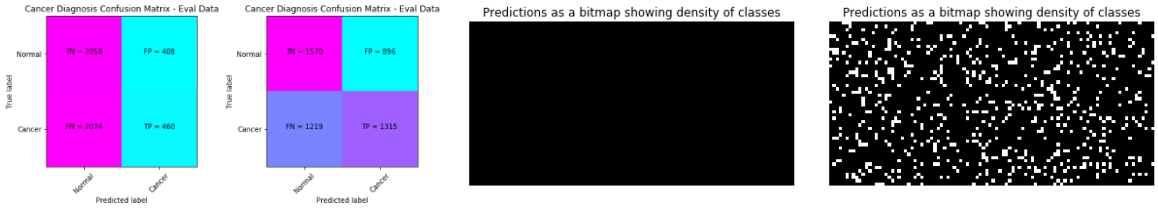



Figure 46: Confusion matrices and bitmaps for overpredicting Normal cases

**Differentiating between the two classes:**

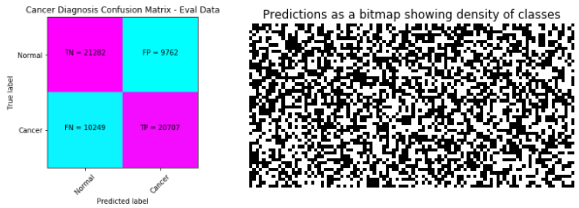



Figure 47: confusion matrix and bitmap for differentiated predictions of both classes

**Over predicting to Cancer class:**


Figure 48: Confusion matrices and bitmaps for overpredicting Cancer cases

The confusion matrices for each of the 9 models are available in Appendix IV (section 8.4)

#### *4.4.2.2.(a)* Analysis

While the accuracy of each model is only in the 64- 69% range, most of the successful models can be seen to differentiate between the two classes with relatively equal proportions. This is with two exceptions which seem to show overprediction. However, other statistical analyses must take place to observe the relationships between these results. It is also important to understand that is possible that none of these models are identifying the correct features when forming a prediction. These visualisations and corresponding data should be used in conjunction with other heuristics such as attention mapping to see whether the correct features are being focused on when these predictions are made.

#### *4.4.2.3 Feature Attention Heatmaps*

It is possible to extract the feature maps at different layers of the neural network to see regions of images that are being 'focused' on when forming the classification. Each map has a different area of focus and the combination of these maps can produce an effective heatmap of attention. These heatmaps describe the important features of an image, which can help to improve overall training. Recent research has attempted to quantify the importance of these images ((Liu *et al.*, 2016) ,(Jetley *et*

*al.*, 2018) ) either during or after training, creating an additional layer of supervision to the learning process. This stage of the experiment utilises this aspect by eliminating models that appear to pay too much attention to incorrect regions of the images. The author acknowledges that this is a qualitative assessment so chose the simplest differentiation (whitespace or tissue focus) to ensure that this could be easily identified as an error in the model for any reader of the report. The following are examples of an evaluation.

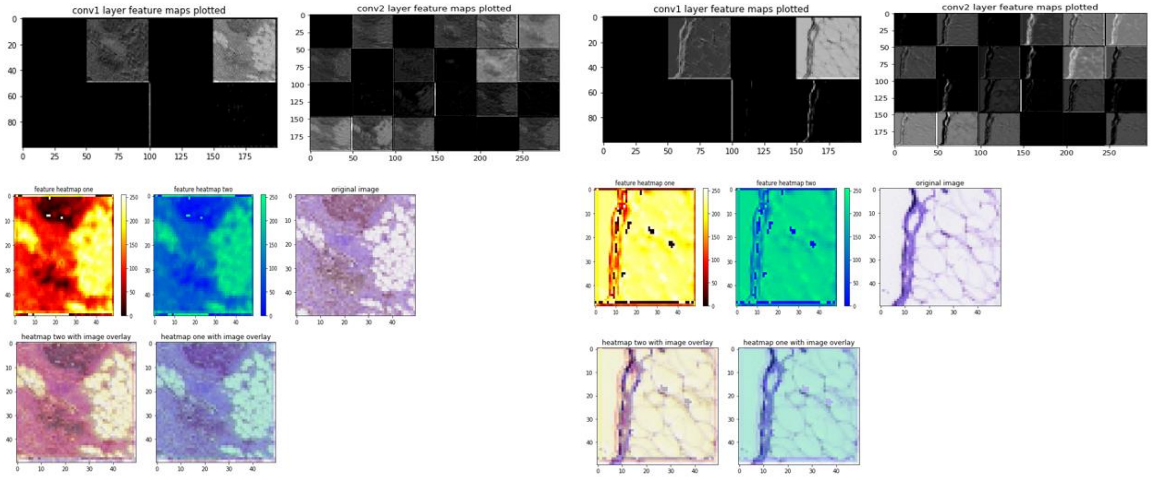



Figure 49: A poor model – too much attention paid to whitespace

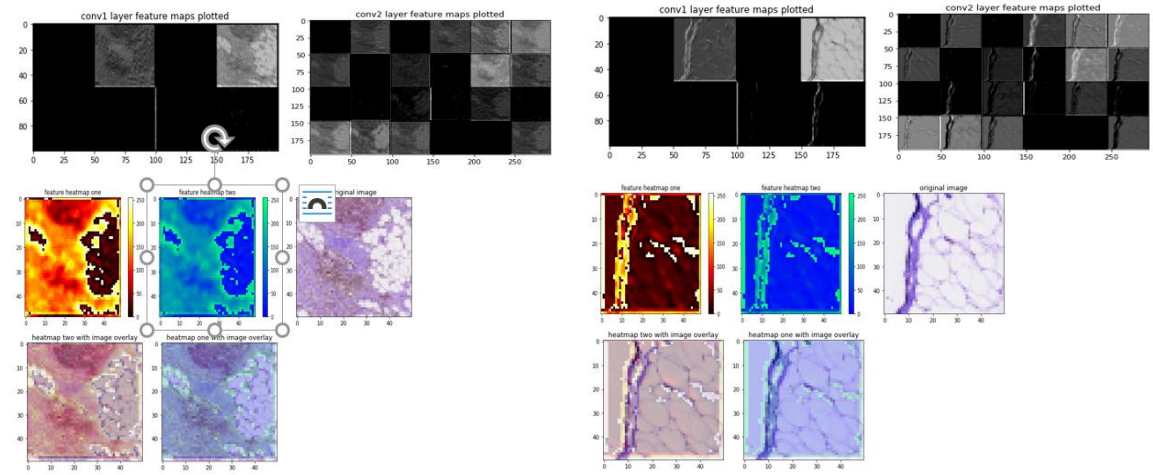



Figure 50: A good model – no/little attention paid to whitespace

Attention heatmaps for each sample/learning rate test case have been provided in Appendix V (section 8.5)

#### *4.4.2.3.(a)* Analysis

Heuristics such as the attention heatmaps required a stronger bias value in ranking, as it became clear that certain trained model had been focusing too much on the whitespace in the microscope image when predicting the class. Where this was found to be the case, a poor scoring (9 for a bad score vs 1 for a good score) was assigned to that test case. The final ranking can be found below

#### *4.4.2.4 Quantitative methods (classification performance statistics)*

The data below presents various statistics. For the purposes of this report the intention is it to minimise the difference in accuracy between predicting cancer correctly (Recall, TPR) and predicting Normal tissue correctly (TNR), whilst achieving the highest rates of accurate predictions for both classes. This decision is supported by the original research where the F-score and balanced accuracy

were used to measure "The trade-off that occurs when simultaneously minimizing FP and FN" (Angel Cruz-Roa, 2014).

| Numeric Index | Test Case (Sample Size and Learning Rate) | True Cancer (TP) | True Normal (TN) | False Cancer (FN) | False Normal (FP) | Recall (TPR) | Precision | F-Score | TNR/ Specificity | Difference( TPR, TNR) | % Accuracy | Balanced Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 196 | 8100 - LR = 0.0022 | 1606 | 1799 | 928 | 667 | 0.63 | 0.71 | 0.67 | 0.73 | 0.0957 | 68.1 | 0.68 |
| 132 | 6000 - LR = 0.0027 | 1677 | 1691 | 857 | 775 | 0.66 | 0.68 | 0.67 | 0.69 | 0.0239 | 67.36 | 0.67 |
| 43 | 4000 - LR = 0.0053 | 1927 | 1426 | 607 | 1040 | 0.76 | 0.65 | 0.70 | 0.58 | 0.1822 | 67.06 | 0.67 |
| 60 | 4600 - LR = 0.0024 | 1657 | 1609 | 877 | 857 | 0.65 | 0.66 | 0.66 | 0.65 | 0.0014 | 65.32 | 0.65 |
| 94 | 5100 - LR = 0.0033 | 1745 | 1636 | 789 | 830 | 0.69 | 0.68 | 0.68 | 0.66 | 0.0252 | 67.62 | 0.68 |
| 181 | 8000 - LR = 0.0029 | 1588 | 1858 | 946 | 608 | 0.63 | 0.72 | 0.67 | 0.75 | 0.1268 | 68.92 | 0.69 |
| 77 | 4900 - LR = 0.0022 | 1571 | 1646 | 963 | 820 | 0.62 | 0.66 | 0.64 | 0.67 | 0.0475 | 64.34 | 0.64 |
| 24 | 3700 - LR = 0.0039 | 1791 | 1584 | 743 | 882 | 0.71 | 0.67 | 0.69 | 0.64 | 0.0645 | 67.5 | 0.67 |
| 146 | 6100 - LR = 0.0023 | 1328 | 1936 | 1206 | 530 | 0.52 | 0.71 | 0.60 | 0.79 | 0.2610 | 65.28 | 0.65 |

Table 4: Performance metrics for each sample size and learning test case

#### *4.4.2.4.(a)* Analysis

The statistical methods above offer the researcher the ability to see at a glance, the performance of each of the test cases broken down by each class prediction. If a trade-off were required, to capture a higher number of one class, this information is incredibly useful to select a higher performing model. However, this data must still be viewed in context. Some models with similar levels of accuracy statistically were found to be predicting based on the presence of whitespace in the image. This speaks not only to data quality issues skewing the performance but also the ability to rely solely quantitative methods as a form of evaluation. The F-Score obtained in the original study for these images (Angel Cruz-Roa, 2014) was 71.8 a model within the dataset (8000 - LR = 0.0029) achieved an F-Score of 69 but was eliminated in the ranking with other metrics because of attention to whitespace.

#### *4.4.2.5 Final Ranking*

Following the assessment across each of the different performance evaluation criteria. The models were ranked relative to the scoring. Some ranks had the same scoring for different models (e.g whitespace attention). These were qualitatively ranked compared with the ranking by more quantitative methods.

| Test Case (sample size and Learning rate) | Rank for Evaluation Accuracy | Rank for Difference(TPR, TNR) | Rank for Whitespace Attention | Rank for Training Error | Scoring |
|---|---|---|---|---|---|
| 8100 - LR = 0.0022 | 2 | 6 | 1 | 2 | 11 |
| 6000 - LR = 0.0027 | 5 | 2 | 1 | 3 | 11 |
| 4000 - LR = 0.0053 | 6 | 8 | 1 | 1 | 16 |
| 4600 - LR = 0.0024 | 7 | 1 | 1 | 7 | 16 |
| 5100 - LR = 0.0033 | 3 | 3 | 9 | 3 | 18 |
| 8000 - LR = 0.0029 | 1 | 7 | 9 | 2 | 19 |
| 4900 - LR = 0.0022 | 9 | 4 | 1 | 7 | 21 |
| 3700 - LR = 0.0039 | 4 | 5 | 8 | 6 | 23 |
| 6100 - LR = 0.0023 | 8 | 9 | 1 | 6 | 24 |

Table 5: Final ranking for each test case

#### *4.4.2.5.(a)* Analysis

This table shows the final ranking and scoring of the test cases. Interestingly, even with scoring the ranking remains aligned with the ranking of accuracy. The mixture of quantitative and qualitative scoring was given equal weighting initially. However higher values were assigned where the researcher could see clear errors such as convergence around 50% in the error during training or too much focus paid to whitespace in the feature maps.

### 4.4.3 Stage 3

For stage three, the top three test cases from the ranked table were taken and evaluated with a larger dataset (62000 samples) to test the resilience of the accuracy of the model. The training and test datasets were kept the same to ensure the model learned in the same way it had in previous stages. As a result, the important information was the evaluation accuracy, confusion matrix values and F-scores. Additional graphs were produced to show the distribution of 'confidence' in predicting sample classes. This was separated by successful and unsuccessful predictions for each class. Making uncertainty simple to identify.

The researcher acknowledges that the qualitative scoring in stage two may not be the best measure so if the resilience of the accuracy was not stable for the best models identified then other models could be explored.

#### *4.4.3.1 Test Case 1 Sample Size 8100 - LR = 0.0022*

| Test Case | TP | TN | FN | FP | Precision | Recall/Sensitivity | F-Score | Specificity | Accuracy |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 19803 | 22605 | 11153 | 8439 | 0.70 | 0.64 | 0.67 | 0.73 | 0.68 |

Table 6: Performance metrics

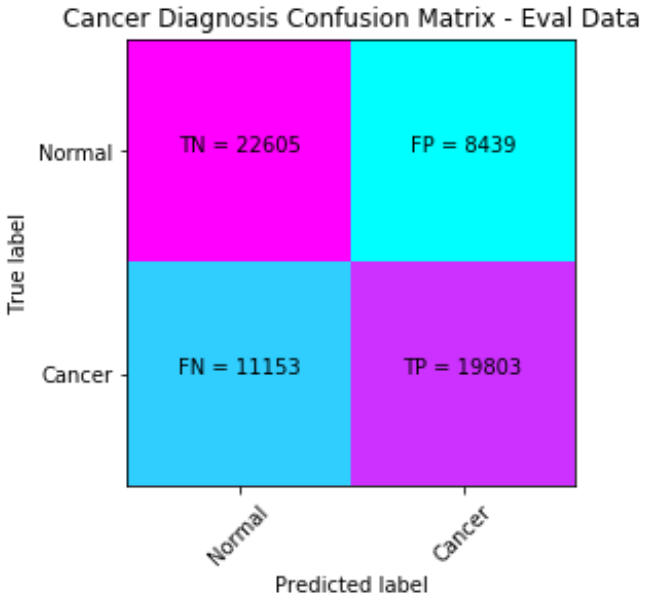



Figure 51: Confusion Matrix

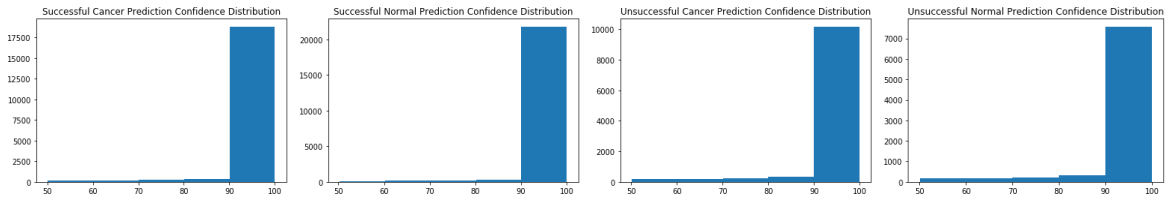



Figure 52: Prediction confidence distributions

#### *4.4.3.1.(a)* ROC Curve

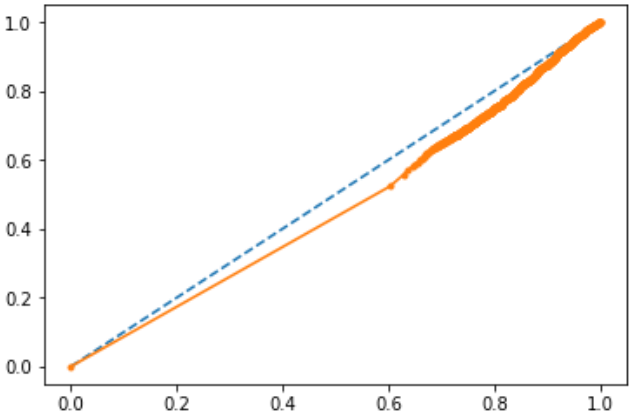



Figure 53: ROC curve

AUC: 0.46

### *4.4.3.2 Test Case 2 Sample Size 6000 - LR = 0.0027*

| Test Case | TP | TN | FN | FP | Precision | Recall/Sensitivity | F-Score | Specificity | Accuracy |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 20707 | 21282 | 10249 | 9762 | 0.68 | 0.67 | 0.67 | 0.69 | 0.68 |

Table 7: Performance Metrics

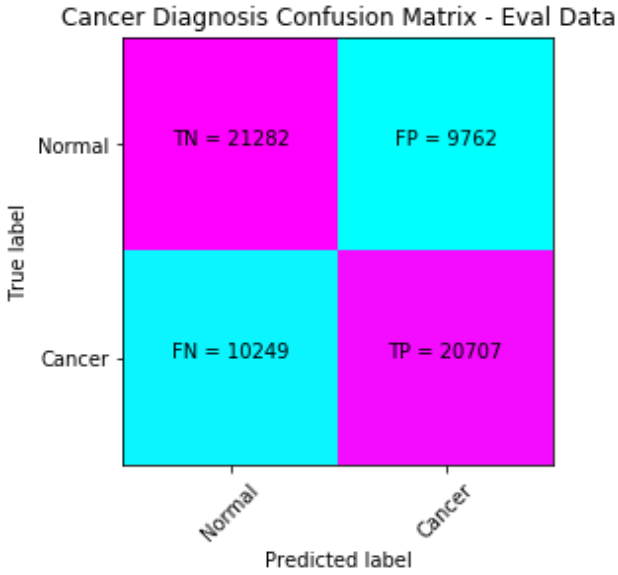



Figure 54: Confusion Matrix

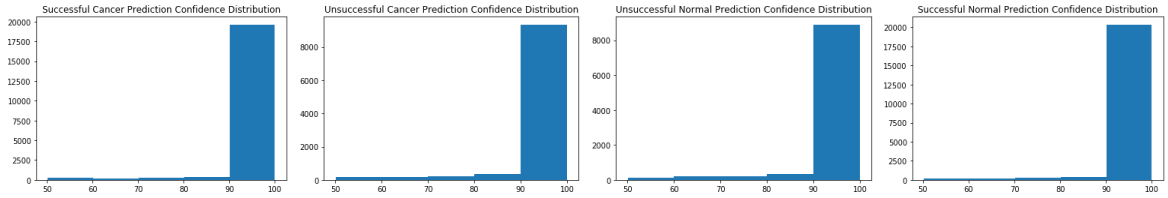



Figure 55: Prediction confidence distributions

#### *4.4.3.2.(a)* ROC Curve

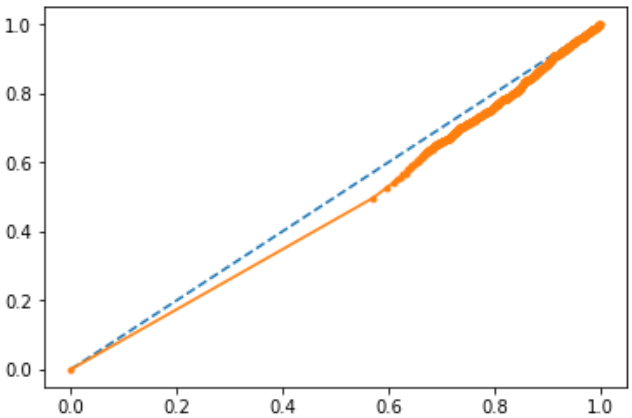



Figure 56: ROC curve

AUC: 0.46

### *4.4.3.3 Test Case 3 Sample Size 4000 - LR = 0.0053*

| Test Case | TP | TN | FN | FP | Precision | Recall/Sensitivity | F-Score | Specificity | Accuracy |
|---|---|---|---|---|---|---|---|---|---|
| 3 | 23611 | 17764 | 7345 | 13280 | 0.64 | 0.76 | 0.70 | 0.57 | 0.67 |

Table 8: Performance Metrics

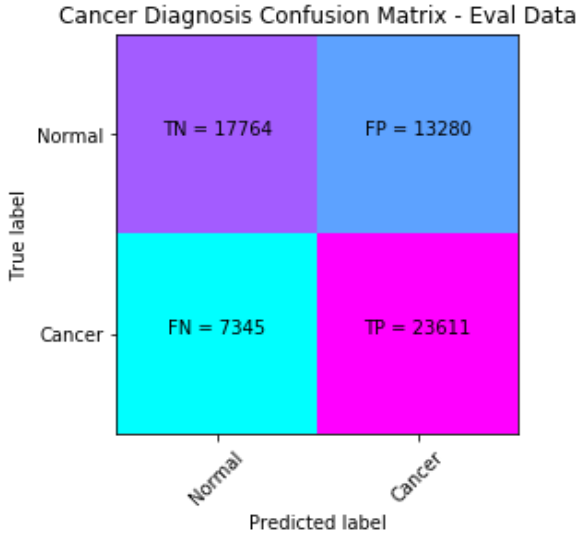



Figure 57: Confusion Matrix

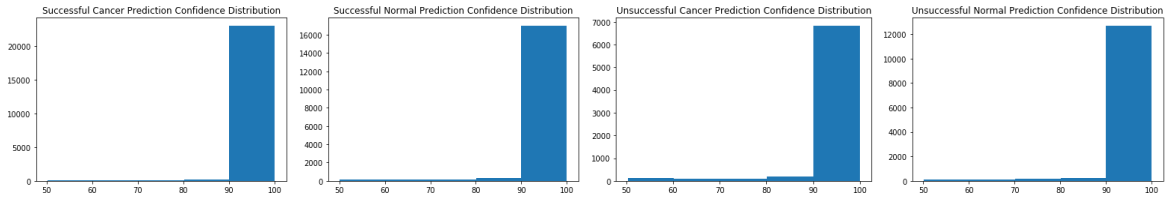



Figure 58: Prediction confidence distribution

### *4.4.3.3.(a)* ROC Curve

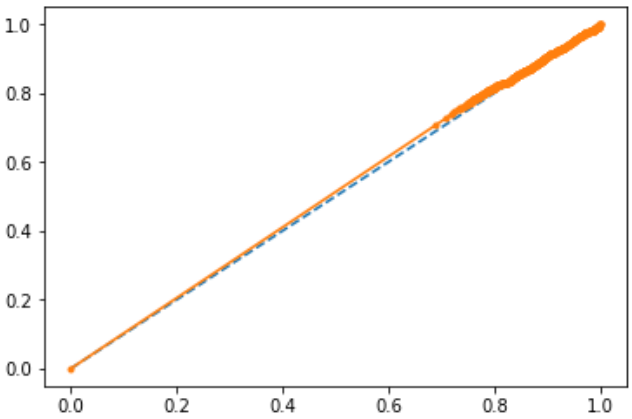



Figure 59: ROC curve

AUC: 0.509

#### *4.4.3.4 Analysis*

Original test:

| Test Case | Test Case (Sample Size and Learning Rate) | True Cancer (TP) | True Normal (TN) | False Cancer (FN) | False Normal (FP) | Recall/ Sensitivity (TPR) | Precision | F-Score | % Accuracy |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 8100 - LR = 0.0022 | 1606 | 1799 | 928 | 667 | 0.63 | 0.71 | 0.67 | 68.1 |
| 2 | 6000 - LR = 0.0027 | 1677 | 1691 | 857 | 775 | 0.66 | 0.68 | 0.67 | 67.36 |
| 3 | 4000 - LR = 0.0053 | 1927 | 1426 | 607 | 1040 | 0.76 | 0.65 | 0.70 | 67.06 |

Table 9: Performance Metrics from stage 2

Larger Dataset (62000 samples):

| Test Case | TP | TN | FN | FP | Precision | Recall/ Sensitivity (TPR) | F-Score | Specificity | AUC | % Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 19803 | 22605 | 11153 | 8439 | 0.70 | 0.64 | 0.67 | 0.73 | 0.46 | 0.68 |
| 2 | 20707 | 21282 | 10249 | 9762 | 0.68 | 0.67 | 0.67 | 0.69 | 0.46 | 0.68 |
| 3 | 23611 | 17764 | 7345 | 13280 | 0.64 | 0.76 | 0.70 | 0.57 | 0.51 | 0.67 |

Table 10: Performance Metrics when testing against a larger dataset

When tested against a larger dataset, the accuracy remains similar showing the model is stable when scaling across the larger datasets. This is also true at individual class prediction scoring such as recall and precision. If a researcher was attempting to capture as many cancer cases as possible, they may opt for test case 3 as the recall is greatest. Alternatively, if the aim is to minimise misdiagnoses of healthy people then the precision would be used, resulting in the selection of test case 1.

When reviewing the ROC curves, (indicated in the table by the AUC score) it is clear that the ability for each of the models to effectively discriminate between the two classes is incredibly limited. Therefore, there is a need for further testing of network parameters to see if there is a way to optimise this feature of statistical analysis. It should be noted that the ROC seeks to define an optimum threshold for classification and still cannot comment on the quality of the feature extraction being used to identify these classes. It may also be the case that AUC is not necessarily the best measure for the purposes of this research as a poor AUC may not always represent the "goodness-of-fit" of the model (Lobo, Jiménez-Valverde and Real, 2008). However, the author of this report acknowledges the necessity to present this metric. Further testing was carried out in stage 4 to evaluate the metric itself after sufficient progress was made in developing a sensible model.

Examples of the images and prediction outputs are available in Appendix VI (section 8.6)

### 4.4.4 Stage 4

The final stage of testing involves changing parameters in the neural network topology to observe the effects on the performance metrics of the model. For this stage the number of layers remained unchanged, but the number of feature maps and the dimensions of the convolutional kernels were varied, and the effects recorded. For this stage the quantitative and attention mapping metrics were utilised to provide a clear understanding at both an individual image and across the evaluation dataset. The author would suggest that varying these hyperparameters may result in a model with greater predictive performance. However eventually, testing will push the model beyond sensible constraints (such as convolution dimensions close to the image dimensions) and result in a degradation in performance.

#### *4.4.4.1 Test Case 3 4000 - LR = 0.0053 – 12 filters in convolution layer 1*

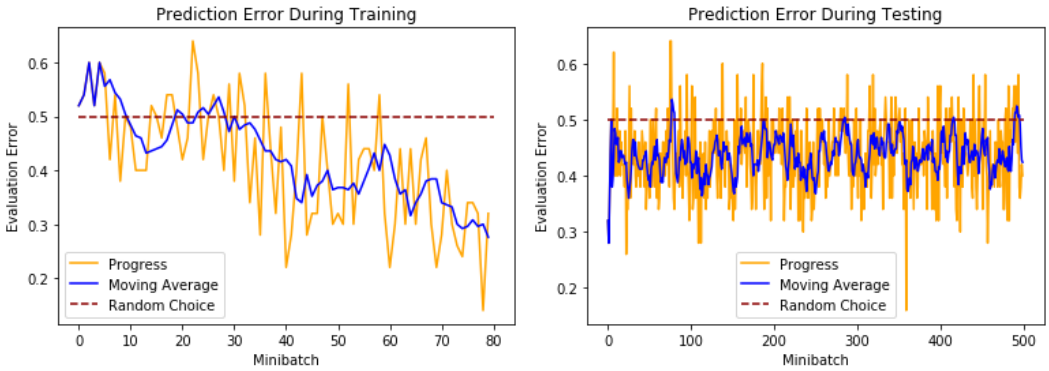



Figure 60: Train/Test minibatch error rates - Average test error: 43.67%

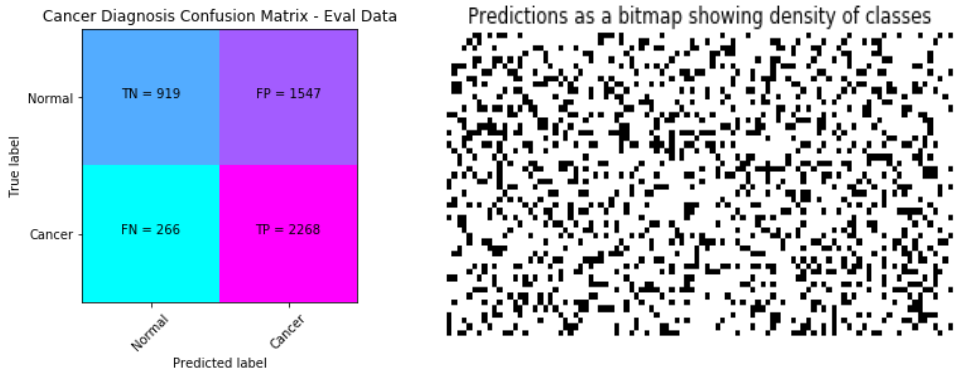



Figure 61: Confusion matrix and bitmap

| Kernels in Conv 1 Layer | Test Case | TP | TN | FN | FP | Precision | Recall | F-Score | Sensitivity | Specificity | Accuracy |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 3 | 2268 | 919 | 266 | 1547 | 0.59 | 0.90 | 0.71 | 0.90 | 0.37 | 0.64 |

Figure 62: Performance metrics

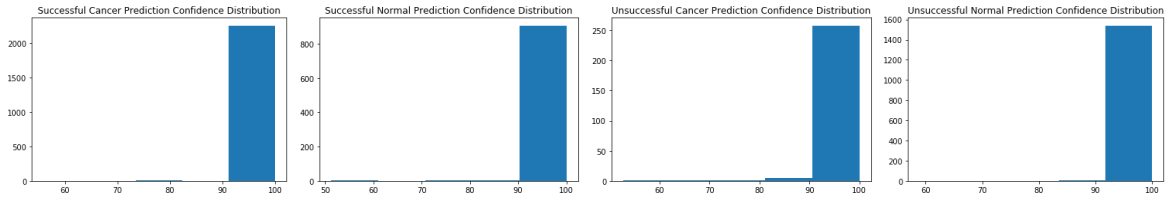



Figure 63: Prediction confidence distributions

#### *4.4.4.1.(a)* Analysis

It is evident that increasing the feature maps further would not benefit this test case as the test error has reduced and the model is overpredicting to the cancer class. This may be useful for researchers looking to maximise recall however for the purposes of this research it is not necessary to continue testing on this case.

#### *4.4.4.2 Test Case 2 6000 - LR = 0.0027 – 12 filters in convolution layer 1*

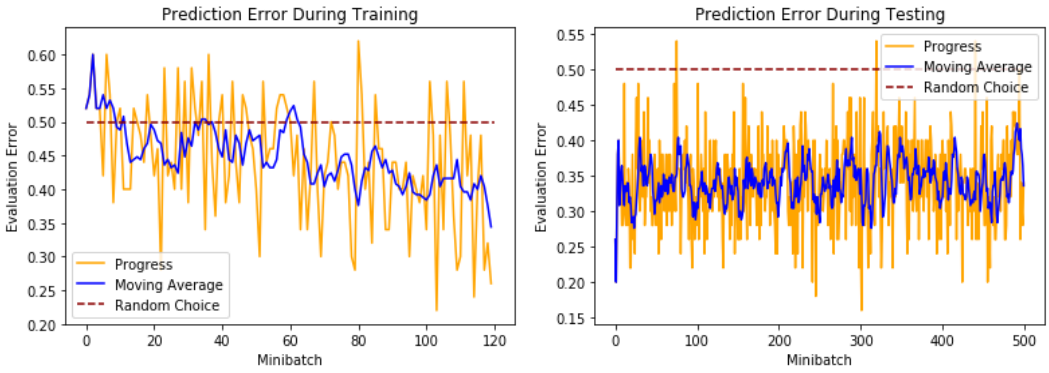



Figure 64: Train/Test minibatch error rates - Average test error: 34.07%

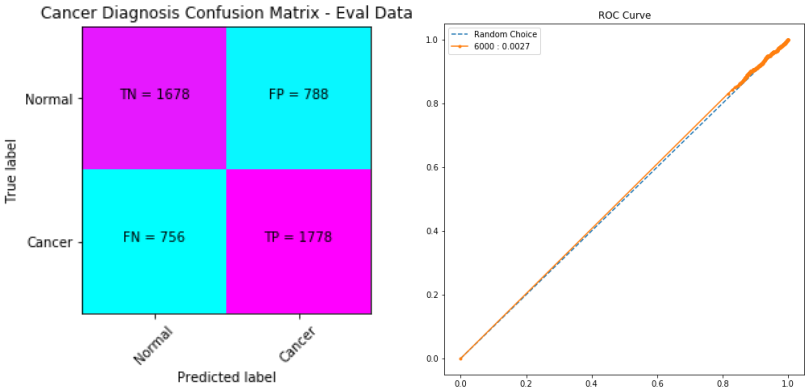



Figure 65: Confusion Matrix and ROC curve

| Kernels in Conv 1 Layer | Test Case | TP | TN | FN | FP | Precision | Recall | F-Score | Sensitivity | Specificity | Accuracy | AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 2 | 1778 | 1678 | 756 | 788 | 0.69 | 0.70 | 0.70 | 0.70 | 0.68 | 0.69 | 0.51 |

Figure 66: Performance metrics

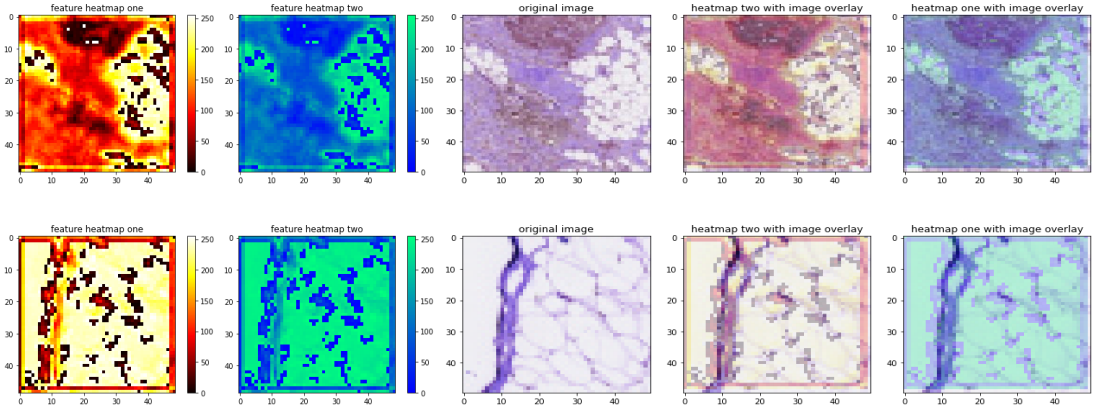



Figure 67: Attention heatmaps

#### *4.4.4.2.(a)* **Analysis**

This test case has been shown to improve the accuracy. However, as is shown by the attention maps, there is too much attention paid to the whitespace in the image and as a result any further testing on this test case would no longer be useful to a researcher. It is worth noting that the AUC has also

increased with the increase of feature maps. However, the attention map analysis supports the notion that all metrics must be considered before arriving at a conclusion about the model.

#### *4.4.4.3 Test Case 1 8100 - LR = 0.0022 – 12 filters in convolution layer 1*

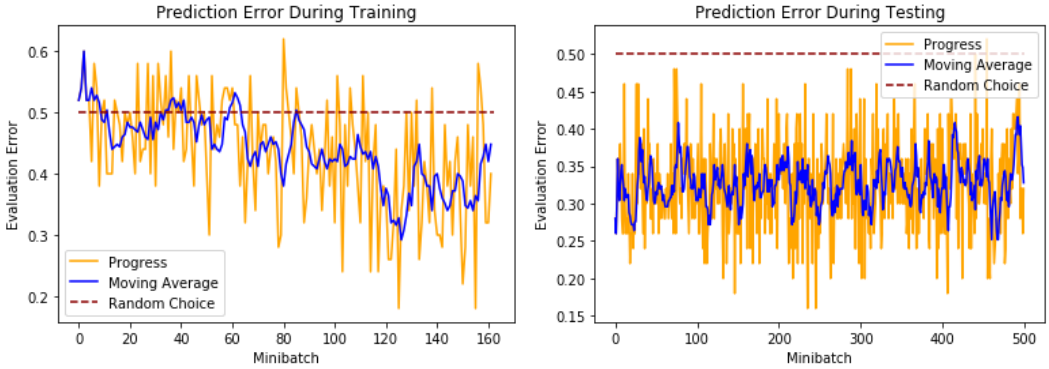



Figure 68: Train/Test minibatch error rates - Average test error: 32.68%

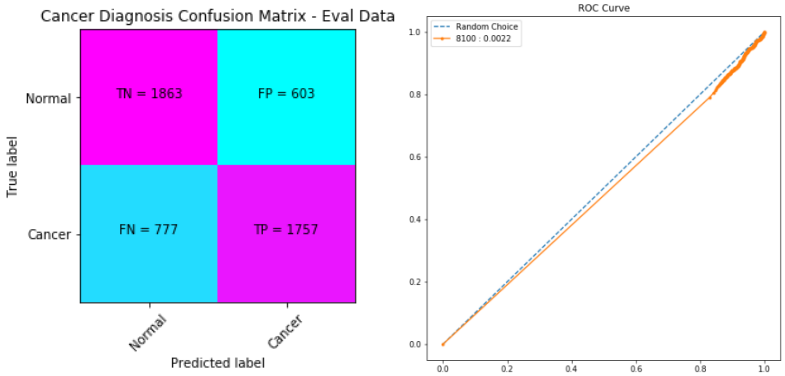



Figure 69: Confusion Matrix and ROC curve

| Kernels in Conv 1 Layer | Test Case | TP | TN | FN | FP | Precision | Recall | F-Score | Sensitivity | Specificity | Accuracy | AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 1 | 1757 | 1863 | 777 | 603 | 0.74 | 0.69 | 0.72 | 0.69 | 0.76 | 0.72 | 0.48 |

Figure 70: Performance metrics

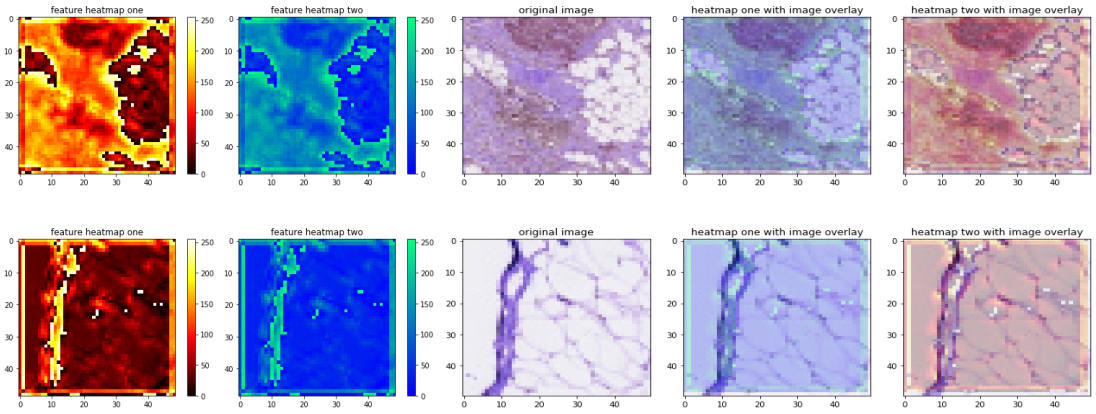



Figure 71: Attention heatmaps

#### *4.4.4.3.(a)* **Analysis**

Increasing the number of filters in the 1st convolutional layer has increased the accuracy of the model in both testing and evaluation with a skew towards classifying the normal cases. This has also

increased the AUC although it is still below 0.5 showing poor discrimination between the classes. It must be pointed out that there is a relationship between this skew towards normal cases (reducing the FPR) and the AUC value. In cases where there is a weight attributed to the positive case this would be poor a classifier, but for the purposes of this research, seeking to maximise the True predictions for both classes it is not particularly relevant. The attention mapping is showing focus on tissue which is a positive result for feature extraction. Complete attention maps are available in Appendix VII (section 8.7)

#### *4.4.4.3.(b)* Iterative cycle of testing

Having eliminated two test cases, the author decided to iterate through various network parameters within test case 1 record any notable effects which these parameters changes may have caused.

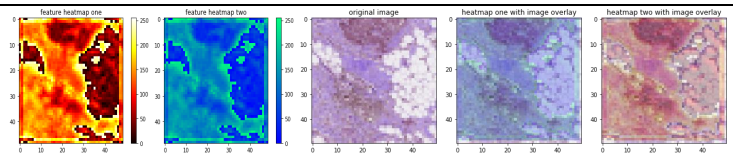
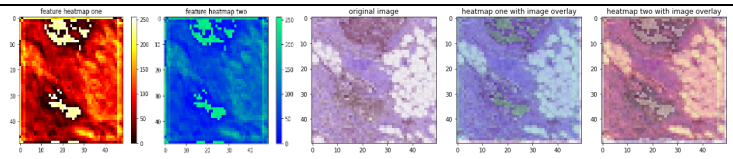
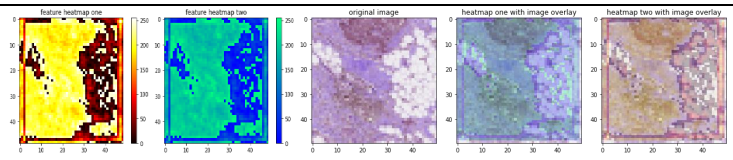
The table below shows the results for models for various kernel sizes (both convolutional layers) and numbers of feature maps in the first convolutional layer for test case 1.

| Test Case | conv1 – kernel number | Kernel 1 - dim | Kernel 2 - dim | Sample size - Learning Rate | TP | TN | FN | FP | Accuracy | TPR/Sen/Recall | Precision | F-Score | FPR | Spec | AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.1 | 12 | 3x3 | 3x3 | 8100 - LR = 0.0022 | 1757 | 1863 | 777 | 603 | 72.4 | 0.693 | 0.744 | 0.718 | 0.245 | 0.755 | 0.480 |
| 1.2 | 12 | 3x3 | 5x5 | 8100 - LR = 0.0022 | 2153 | 1423 | 381 | 1043 | 71.52 | 0.850 | 0.674 | 0.751 | 0.423 | 0.577 | 0.559 |
| 1.3 | 12 | 5x5 | 3x3 | 8100 - LR = 0.0022 | 1701 | 1563 | 833 | 903 | 65.28 | 0.671 | 0.653 | 0.662 | 0.366 | 0.634 | 0.509 |
| 1.4 | 12 | 2x2 | 3x3 | 8100 - LR = 0.0022 | 1779 | 1580 | 755 | 886 | 67.18 | 0.702 | 0.668 | 0.684 | 0.359 | 0.641 | 0.522 |
| 1.5 | 12 | 3x3 | 8x8 | 8100 - LR = 0.0022 | 1898 | 1333 | 636 | 1133 | 64.62 | 0.749 | 0.626 | 0.682 | 0.459 | 0.541 | 0.502 |
| 1.6 | 12 | 8x8 | 3x3 | 8100 - LR = 0.0022 | 1507 | 1775 | 1027 | 691 | 65.64 | 0.595 | 0.686 | 0.637 | 0.280 | 0.720 | 0.491 |
| 1.7 | 16 | 3x3 | 3x3 | 8100 - LR = 0.0022 | 1291 | 1822 | 1243 | 644 | 62.26 | 0.509 | 0.667 | 0.578 | 0.261 | 0.739 | 0.471 |
| 1.8 | 16 | 3x3 | 5x5 | 8100 - LR = 0.0022 | 1434 | 1936 | 1100 | 530 | 67.4 | 0.566 | 0.730 | 0.638 | 0.215 | 0.785 | 0.450 |

Table 11: Performance metrics for network topology iterative testing cycle

#### *4.4.4.3.(c)* Attention maps

For each new test case an analysis of each of one of the attention maps is demonstrated below. This could be useful for a pathologist in evaluating the ability of the model to discriminate between cancerous and normal tissue based on expert opinion. The table below shows the effects a change to the model's parameters has on the quality of feature extraction and output. The same image has been displayed for consistency of testing. All attention maps are provided in Appendix VIII (section 8.8)

| Test Case | Attention Mapping | Analysis |
|---|---|---|
| 1.1 |  | Good discrimination between whitespace and tissue. |
| 1.2 |  | Whitespace included in focus. Certain regions of tissue are identified as having greater importance in the class prediction. |
| 1.3 |  | Whitespace almost ignored. All tissue included in focus |

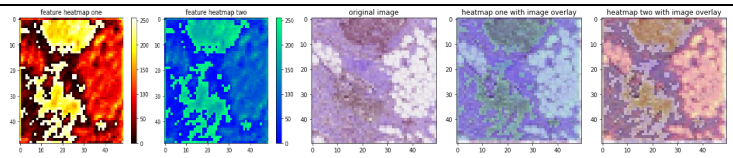
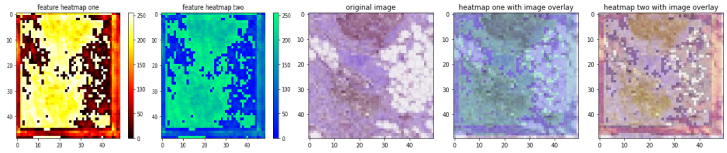
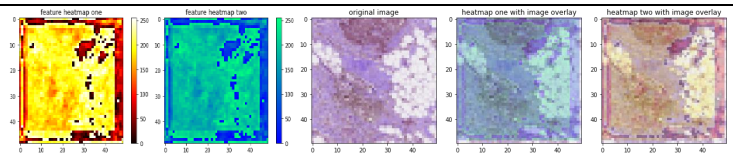
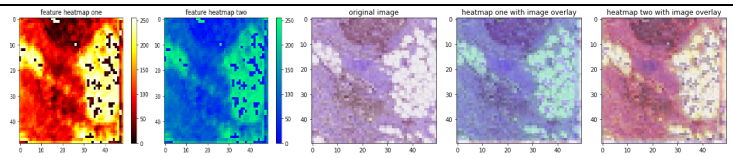
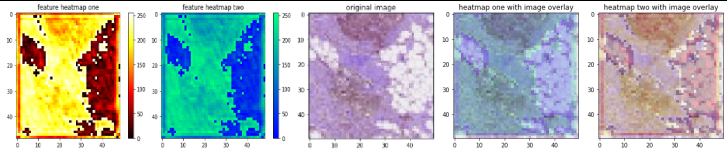
| | | |
|---|---|---|
| 1.4 |  | Whitespace ignored and discrimination in different tissue regions |
| 1.5 |  | Whitespace almost ignored. All tissue included in focus |
| 1.6 |  | Poor discrimination between whitespace and tissue. Almost all regions considered |
| 1.7 |  | Focus appears to be greater for whitespace than tissue |
| 1.8 |  | Whitespace almost ignored. All tissue included in focus |

Table 12: Attention heatmap analysis – network topology parameter Iterative testing cycle

These tests suggest the following:

- With an increase in kernel dimension sizes, there is an increase in the contrast between vastly different features. However, a degradation in the quality of feature extraction in regions of similar pixel ranges. This may hamper differentiation between cancerous and normal tissue. With a larger enough kernel size, the degradation is sufficiently large to lead to an inability to differentiate between tissue and whitespace.
- Increasing the number of kernels used in the first convolutional layer is a balancing act between increased feature recognition, and too much focus being paid to incorrect regions in the images (whitespace).

The top test cases (see table 11) from this iterative cycle were then tested with a larger dataset to validate the stability of the result. The results are shown below:

| Test Case | conv1 – kernel number | kernel1 dim | kernel2 dim | Sample size - Learning Rate | TP | TN | FN | FP | Accuracy | TPR/Sensitivity /Recall | Precision | F-Score | FPR | Specificity | AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.1 | 12 | 3x3 | 3x3 | 8100 - LR = 0.0022 | 21248 | 23532 | 9708 | 7512 | 72.23 | 0.686 | 0.739 | 0.712 | 0.242 | 0.758 | 0.486 |
| 1.2 | 12 | 3x3 | 5x5 | 8100 - LR = 0.0022 | 26195 | 17995 | 4761 | 13049 | 71.27 | 0.846 | 0.667 | 0.746 | 0.420 | 0.580 | 0.555 |

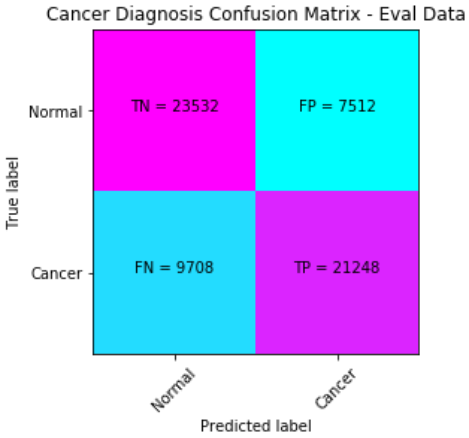
Table 13: Performance metrics testing on the larger evaluation dataset

The cases tested against the larger dataset show stability in the model when generalising against new data. The AUC is still relatively close to random selection again indicating poor discrimination. Examination of the data for test case 1.2 also re-affirms the bias of the AUC metric for maximisation of the positive class alone (at the cost of FP) rather than the distribution of all true class predictions.

#### *4.4.4.4 Result*

Based on the testing conducted above the author of this report would suggest that the optimal model for maximising the accuracy of prediction for both classes is Test Case 1.1. When optimising for Cancer diagnoses, test case 1.2 provides a stable model which reduces the number of false negatives. Further testing with the sole purpose of optimising the AUC is presented in section 4.4.5. performance statistics and visualisations for the larger evaluation dataset for Test cases 1.1 and 1.2 are presented below.

### *4.4.4.4.(a)* Test case 1.1 - Larger dataset




| conv1 – kernel number | kernel1 dim | kernel2 dim | Sample size - Learning Rate | True Positive | True Negative | False Negative | False Positive | TPR/Sensitivity/Recall | Precision | F-Score | FPR | Specificity | Accuracy | AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 3x3 | 3x3 | 8100 - LR = 0.0022 | 21248 | 23532 | 9708 | 7512 | 0.686 | 0.739 | 0.712 | 0.242 | 0.758 | 72.23 | 0.486 |

*Table 14: Performance metrics*

### *4.4.4.4.(b)* Test case 1.2 – Larger Dataset

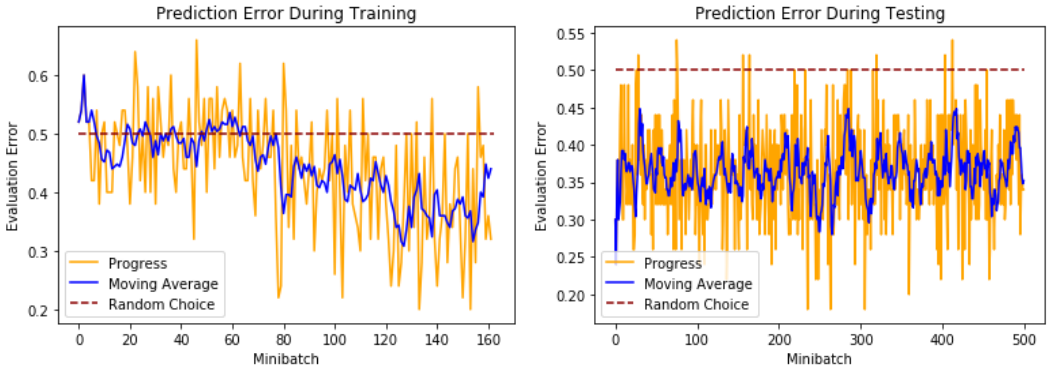



*Table 15: Train/Test minibatch error rates - Average test error: 36.44%*

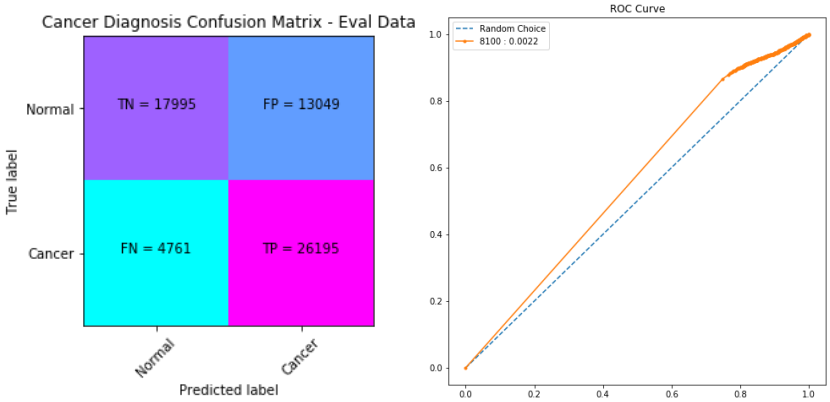



Figure 72: Confusion matrix and ROC curve

| conv1 – kernel number | kernel1 dim | kernel2 dim | Sample size - Learning Rate | True Positive | True Negative | False Negative | False Positive | TPR/Sensitivity/Recall | Precision | F-Score | FPR | Specificity | Accuracy | AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 3x3 | 5x5 | 8100 - LR = 0.0022 | 26195 | 17995 | 4761 | 13049 | 0.846 | 0.667 | 0.746 | 0.420 | 0.580 | 71.27 | 0.56 |

Table 16: Performance metrics

#### *4.4.4.5 Iterative Cycle - Optimising the AUC*

To explore the relevance of the AUC with regard to this research, a series of iterations were carried out specifically to observe the relationship between different performance metrics of the model and the AUC. The data presented shows that where the AUC can be optimised is not necessarily where the model can be trusted to predict sensibly for both classes. The trade-off between the avoidance of FN at the cost of FP is evident and so, for the purposes of this research, should not be considered a suitable metric in terms of "goodness-of-fit" of the model. (Lobo, Jiménez-Valverde and Real, 2008). It is evident that regions of high accuracy may be where a reasonable balance has been struck in the model's ability to predict both classes minimising false positives and negatives. An example of the ROC comparisons is presented in Figure 73.

#### *4.4.4.5.(a)* Iterative test cycle results

| Sample Size - LR | TP | TN | FN | FP | Accuracy | Recall | Precision | F-Score | TPR/ Sensitivity | FPR | Specificity | AUC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5100 - LR = 0.0025 | 2301 | 707 | 233 | 1759 | 60.16 | 0.91 | 0.57 | 0.70 | 0.91 | 0.71 | 0.29 | 0.62 |
| 5100 - LR = 0.002 | 2372 | 504 | 162 | 1962 | 57.52 | 0.94 | 0.55 | 0.69 | 0.94 | 0.80 | 0.20 | 0.61 |
| 4000 - LR = 0.0045 | 2392 | 609 | 142 | 1857 | 60.02 | 0.94 | 0.56 | 0.71 | 0.94 | 0.75 | 0.25 | 0.60 |
| 6200 - LR = 0.0035 | 2381 | 607 | 153 | 1859 | 59.76 | 0.94 | 0.56 | 0.70 | 0.94 | 0.75 | 0.25 | 0.59 |
| 8000 - LR = 0.002 | 2260 | 702 | 274 | 1764 | 59.24 | 0.89 | 0.56 | 0.69 | 0.89 | 0.72 | 0.28 | 0.59 |
| 8000 - LR = 0.001 | 2213 | 715 | 321 | 1751 | 58.56 | 0.87 | 0.56 | 0.68 | 0.87 | 0.71 | 0.29 | 0.59 |
| 8000 - LR = 0.0015 | 2336 | 490 | 198 | 1976 | 56.52 | 0.92 | 0.54 | 0.68 | 0.92 | 0.80 | 0.20 | 0.59 |
| 5100 - LR = 0.0015 | 2201 | 798 | 333 | 1668 | 59.98 | 0.87 | 0.57 | 0.69 | 0.87 | 0.68 | 0.32 | 0.59 |
| 4000 - LR = 0.004 | 2494 | 191 | 40 | 2275 | 53.7 | 0.98 | 0.52 | 0.68 | 0.98 | 0.92 | 0.08 | 0.57 |
| 8000 - LR = 0.0025 | 2002 | 1296 | 532 | 1170 | 65.96 | 0.79 | 0.63 | 0.70 | 0.79 | 0.47 | 0.53 | 0.57 |
| 6200 - LR = 0.004 | 2128 | 1246 | 406 | 1220 | 67.48 | 0.84 | 0.64 | 0.72 | 0.84 | 0.49 | 0.51 | 0.56 |
| 4000 - LR = 0.005 | 2159 | 1106 | 375 | 1360 | 65.3 | 0.85 | 0.61 | 0.71 | 0.85 | 0.55 | 0.45 | 0.55 |
| 6200 - LR = 0.003 | 2468 | 222 | 66 | 2244 | 53.8 | 0.97 | 0.52 | 0.68 | 0.97 | 0.91 | 0.09 | 0.55 |
| 6000 - LR = 0.002 | 1997 | 1239 | 537 | 1227 | 64.72 | 0.79 | 0.62 | 0.69 | 0.79 | 0.50 | 0.50 | 0.55 |
| 5100 - LR = 0.003 | 2015 | 1292 | 519 | 1174 | 66.14 | 0.80 | 0.63 | 0.70 | 0.80 | 0.48 | 0.52 | 0.55 |
| 6200 - LR = 0.0025 | 2503 | 100 | 31 | 2366 | 52.06 | 0.99 | 0.51 | 0.68 | 0.99 | 0.96 | 0.04 | 0.52 |
| 4000 - LR = 0.0035 | 2520 | 59 | 14 | 2407 | 51.58 | 0.99 | 0.51 | 0.68 | 0.99 | 0.98 | 0.02 | 0.52 |
| 4000 - LR = 0.001 | 2521 | 38 | 13 | 2428 | 51.18 | 0.99 | 0.51 | 0.67 | 0.99 | 0.98 | 0.02 | 0.51 |
| 6000 - LR = 0.0015 | 1868 | 1303 | 666 | 1163 | 63.42 | 0.74 | 0.62 | 0.67 | 0.74 | 0.47 | 0.53 | 0.51 |
| 6200 - LR = 0.002 | 2519 | 47 | 15 | 2419 | 51.32 | 0.99 | 0.51 | 0.67 | 0.99 | 0.98 | 0.02 | 0.51 |
| 6200 - LR = 0.001 | 2529 | 26 | 5 | 2440 | 51.1 | 1.00 | 0.51 | 0.67 | 1.00 | 0.99 | 0.01 | 0.51 |
| 6200 - LR = 0.0015 | 2529 | 31 | 5 | 2435 | 51.2 | 1.00 | 0.51 | 0.67 | 1.00 | 0.99 | 0.01 | 0.51 |
| 4000 - LR = 0.003 | 2529 | 26 | 5 | 2440 | 51.1 | 1.00 | 0.51 | 0.67 | 1.00 | 0.99 | 0.01 | 0.51 |
| 4000 - LR = 0.0015 | 2532 | 12 | 2 | 2454 | 50.88 | 1.00 | 0.51 | 0.67 | 1.00 | 1.00 | 0.00 | 0.51 |
| 4000 - LR = 0.0025 | 2532 | 11 | 2 | 2455 | 50.86 | 1.00 | 0.51 | 0.67 | 1.00 | 1.00 | 0.00 | 0.50 |
| 4000 - LR = 0.002 | 2533 | 6 | 1 | 2460 | 50.78 | 1.00 | 0.51 | 0.67 | 1.00 | 1.00 | 0.00 | 0.50 |
| 4200 - LR = 0.0075 | 56 | 2407 | 2478 | 59 | 49.26 | 0.02 | 0.49 | 0.04 | 0.02 | 0.02 | 0.98 | 0.50 |
| 8100 - LR = 0.0015 | 1767 | 1535 | 767 | 931 | 66.04 | 0.70 | 0.65 | 0.68 | 0.70 | 0.38 | 0.62 | 0.50 |
| 5000 - LR = 0.0085 | 1 | 2464 | 2533 | 2 | 49.3 | 0.00 | 0.33 | 0.00 | 0.00 | 0.00 | 1.00 | 0.50 |
| 8100 - LR = 0.0085 | 1 | 2465 | 2533 | 1 | 49.32 | 0.00 | 0.50 | 0.00 | 0.00 | 0.00 | 1.00 | 0.50 |
| 6000 - LR = 0.0025 | 1810 | 1542 | 724 | 924 | 67.04 | 0.71 | 0.66 | 0.69 | 0.71 | 0.37 | 0.63 | 0.50 |
| 4400 - LR = 0.0075 | 9 | 2453 | 2525 | 13 | 49.24 | 0.00 | 0.41 | 0.01 | 0.00 | 0.01 | 0.99 | 0.50 |
| 6200 - LR = 0.0085 | 6 | 2463 | 2528 | 3 | 49.38 | 0.00 | 0.67 | 0.00 | 0.00 | 0.00 | 1.00 | 0.50 |
| 4200 - LR = 0.008 | 55 | 2413 | 2479 | 53 | 49.36 | 0.02 | 0.51 | 0.04 | 0.02 | 0.02 | 0.98 | 0.50 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 8000 - LR = 0.0045 | 10 | 2465 | 2524 | 1 | 49.5 | 0.00 | 0.91 | 0.01 | 0.00 | 0.00 | 1.00 | 0.50 |
| 4200 - LR = 0.0085 | 35 | 2430 | 2499 | 36 | 49.3 | 0.01 | 0.49 | 0.03 | 0.01 | 0.01 | 0.99 | 0.49 |
| 6200 - LR = 0.008 | 7 | 2464 | 2527 | 2 | 49.42 | 0.00 | 0.78 | 0.01 | 0.00 | 0.00 | 1.00 | 0.49 |
| 6000 - LR = 0.005 | 8 | 2465 | 2526 | 1 | 49.46 | 0.00 | 0.89 | 0.01 | 0.00 | 0.00 | 1.00 | 0.49 |
| 5100 - LR = 0.0055 | 8 | 2465 | 2526 | 1 | 49.46 | 0.00 | 0.89 | 0.01 | 0.00 | 0.00 | 1.00 | 0.49 |
| 6200 - LR = 0.0065 | 22 | 2460 | 2512 | 6 | 49.64 | 0.01 | 0.79 | 0.02 | 0.01 | 0.00 | 1.00 | 0.49 |
| 8100 - LR = 0.002 | 1706 | 1674 | 828 | 792 | 67.6 | 0.67 | 0.68 | 0.68 | 0.67 | 0.32 | 0.68 | 0.49 |
| 6200 - LR = 0.0075 | 20 | 2461 | 2514 | 5 | 49.62 | 0.01 | 0.80 | 0.02 | 0.01 | 0.00 | 1.00 | 0.48 |
| 4000 - LR = 0.0085 | 1353 | 1612 | 1181 | 854 | 59.3 | 0.53 | 0.61 | 0.57 | 0.53 | 0.35 | 0.65 | 0.48 |
| 5100 - LR = 0.0075 | 17 | 2453 | 2517 | 13 | 49.4 | 0.01 | 0.57 | 0.01 | 0.01 | 0.01 | 0.99 | 0.48 |
| 8100 - LR = 0.004 | 31 | 2457 | 2503 | 9 | 49.76 | 0.01 | 0.78 | 0.02 | 0.01 | 0.00 | 1.00 | 0.47 |
| 4000 - LR = 0.008 | 1099 | 1774 | 1435 | 692 | 57.46 | 0.43 | 0.61 | 0.51 | 0.43 | 0.28 | 0.72 | 0.47 |
| 6200 - LR = 0.006 | 89 | 2443 | 2445 | 23 | 50.64 | 0.04 | 0.79 | 0.07 | 0.04 | 0.01 | 0.99 | 0.47 |
| 5100 - LR = 0.008 | 41 | 2453 | 2493 | 13 | 49.88 | 0.02 | 0.76 | 0.03 | 0.02 | 0.01 | 0.99 | 0.46 |
| 5100 - LR = 0.0085 | 74 | 2449 | 2460 | 17 | 50.46 | 0.03 | 0.81 | 0.06 | 0.03 | 0.01 | 0.99 | 0.46 |
| 4000 - LR = 0.0055 | 1707 | 1621 | 827 | 845 | 66.56 | 0.67 | 0.67 | 0.67 | 0.67 | 0.34 | 0.66 | 0.46 |
| 5000 - LR = 0.0045 | 18 | 2458 | 2516 | 8 | 49.52 | 0.01 | 0.69 | 0.01 | 0.01 | 0.00 | 1.00 | 0.46 |
| 4000 - LR = 0.0075 | 717 | 1978 | 1817 | 488 | 53.9 | 0.28 | 0.60 | 0.38 | 0.28 | 0.20 | 0.80 | 0.46 |
| 6200 - LR = 0.0045 | 1575 | 1876 | 959 | 590 | 69.02 | 0.62 | 0.73 | 0.67 | 0.62 | 0.24 | 0.76 | 0.46 |
| 5100 - LR = 0.005 | 59 | 2448 | 2475 | 18 | 50.14 | 0.02 | 0.77 | 0.05 | 0.02 | 0.01 | 0.99 | 0.45 |
| 8000 - LR = 0.003 | 1465 | 1956 | 1069 | 510 | 68.42 | 0.58 | 0.74 | 0.65 | 0.58 | 0.21 | 0.79 | 0.45 |
| 6000 - LR = 0.0045 | 92 | 2441 | 2442 | 25 | 50.66 | 0.04 | 0.79 | 0.07 | 0.04 | 0.01 | 0.99 | 0.45 |
| 8100 - LR = 0.001 | 1521 | 1636 | 1013 | 830 | 63.14 | 0.60 | 0.65 | 0.62 | 0.60 | 0.34 | 0.66 | 0.44 |
| 5100 - LR = 0.0035 | 1527 | 1841 | 1007 | 625 | 67.36 | 0.60 | 0.71 | 0.65 | 0.60 | 0.25 | 0.75 | 0.43 |
| 6200 - LR = 0.0055 | 303 | 2372 | 2231 | 94 | 53.5 | 0.12 | 0.76 | 0.21 | 0.12 | 0.04 | 0.96 | 0.43 |
| 8000 - LR = 0.004 | 180 | 2424 | 2354 | 42 | 52.08 | 0.07 | 0.81 | 0.13 | 0.07 | 0.02 | 0.98 | 0.43 |
| 8100 - LR = 0.0025 | 1393 | 1977 | 1141 | 489 | 67.4 | 0.55 | 0.74 | 0.63 | 0.55 | 0.20 | 0.80 | 0.41 |
| 6000 - LR = 0.003 | 1409 | 1917 | 1125 | 549 | 66.52 | 0.56 | 0.72 | 0.63 | 0.56 | 0.22 | 0.78 | 0.41 |
| 5000 - LR = 0.004 | 85 | 2435 | 2449 | 31 | 50.4 | 0.03 | 0.73 | 0.06 | 0.03 | 0.01 | 0.99 | 0.41 |
| 4000 - LR = 0.006 | 1309 | 1922 | 1225 | 544 | 64.62 | 0.52 | 0.71 | 0.60 | 0.52 | 0.22 | 0.78 | 0.40 |
| 4000 - LR = 0.007 | 85 | 2440 | 2449 | 26 | 50.5 | 0.03 | 0.77 | 0.06 | 0.03 | 0.01 | 0.99 | 0.39 |
| 5000 - LR = 0.001 | 105 | 2383 | 2429 | 83 | 49.76 | 0.04 | 0.56 | 0.08 | 0.04 | 0.03 | 0.97 | 0.39 |
| 6200 - LR = 0.005 | 795 | 2236 | 1739 | 230 | 60.62 | 0.31 | 0.78 | 0.45 | 0.31 | 0.09 | 0.91 | 0.39 |
| 8100 - LR = 0.0035 | 372 | 2360 | 2162 | 106 | 54.64 | 0.15 | 0.78 | 0.25 | 0.15 | 0.04 | 0.96 | 0.39 |
| 8000 - LR = 0.0035 | 811 | 2284 | 1723 | 182 | 61.9 | 0.32 | 0.82 | 0.46 | 0.32 | 0.07 | 0.93 | 0.38 |
| 5100 - LR = 0.001 | 1119 | 1793 | 1415 | 673 | 58.24 | 0.44 | 0.62 | 0.52 | 0.44 | 0.27 | 0.73 | 0.38 |
| 4000 - LR = 0.0065 | 885 | 2168 | 1649 | 298 | 61.06 | 0.35 | 0.75 | 0.48 | 0.35 | 0.12 | 0.88 | 0.38 |
| 6000 - LR = 0.004 | 427 | 2345 | 2107 | 121 | 55.44 | 0.17 | 0.78 | 0.28 | 0.17 | 0.05 | 0.95 | 0.37 |
| 5100 - LR = 0.0045 | 392 | 2353 | 2142 | 113 | 54.9 | 0.15 | 0.78 | 0.26 | 0.15 | 0.05 | 0.95 | 0.37 |
| 8100 - LR = 0.003 | 943 | 2222 | 1591 | 244 | 63.3 | 0.37 | 0.79 | 0.51 | 0.37 | 0.10 | 0.90 | 0.36 |
| 6000 - LR = 0.001 | 892 | 1967 | 1642 | 499 | 57.18 | 0.35 | 0.64 | 0.45 | 0.35 | 0.20 | 0.80 | 0.36 |
| 5000 - LR = 0.0035 | 306 | 2374 | 2228 | 92 | 53.6 | 0.12 | 0.77 | 0.21 | 0.12 | 0.04 | 0.96 | 0.35 |
| 6000 - LR = 0.0035 | 904 | 2195 | 1630 | 271 | 61.98 | 0.36 | 0.77 | 0.49 | 0.36 | 0.11 | 0.89 | 0.35 |
| 5100 - LR = 0.004 | 938 | 2180 | 1596 | 286 | 62.36 | 0.37 | 0.77 | 0.50 | 0.37 | 0.12 | 0.88 | 0.35 |
| 5000 - LR = 0.002 | 1002 | 2017 | 1532 | 449 | 60.38 | 0.40 | 0.69 | 0.50 | 0.40 | 0.18 | 0.82 | 0.34 |
| 5000 - LR = 0.0015 | 633 | 2147 | 1901 | 319 | 55.6 | 0.25 | 0.66 | 0.36 | 0.25 | 0.13 | 0.87 | 0.34 |

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 5000 - LR = 0.0025 | 902 | 2134 | 1632 | 332 | 60.72 | 0.36 | 0.73 | 0.48 | 0.36 | 0.13 | 0.87 | 0.33 |
| 5000 - LR = 0.003 | 593 | 2285 | 1941 | 181 | 57.56 | 0.23 | 0.77 | 0.36 | 0.23 | 0.07 | 0.93 | 0.32 |

**Table 17: Performance metrics iterative testing cycle: Optimising the AUC**

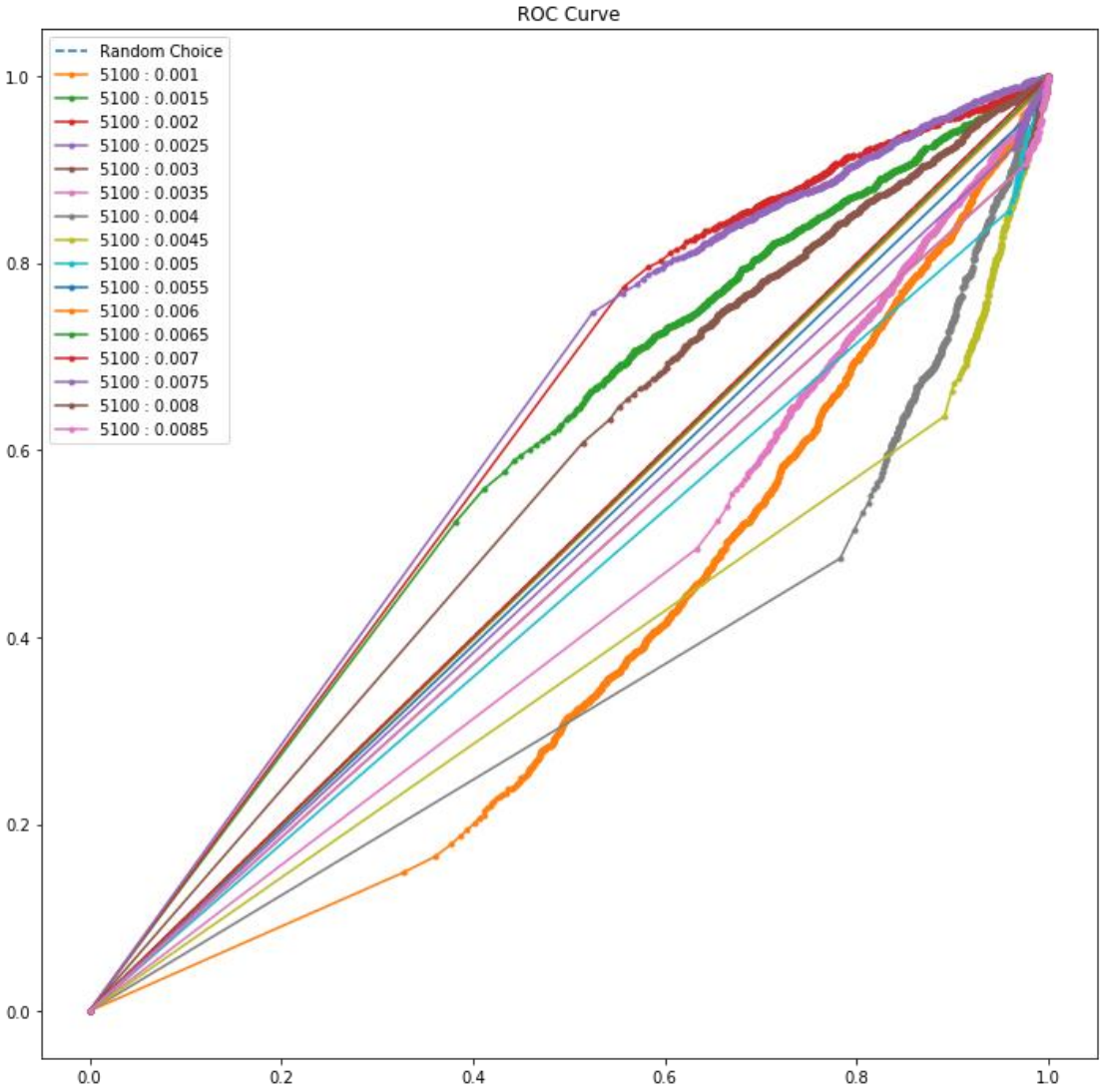



Figure 73: Example ROC curves obtained for a range of learning rates during the iterative test cycle - optimising the AUC

## 4.4.5 Experimental Results Storage

All model testing and results can be viewed as read-only Jupyter Notebooks in Microsoft Azure notebooks at the following address:

https://notebooks.azure.com/garyos/projects/tutorials/tree/Final%20Year%20Project

# CHAPTER 5: DISCUSSION AND CONCLUSIONS

## 5.1 Interpretation of Results

### 5.1.1 Stage 1

This stage saw the initial testing of the ability of the model to learn and attempt to predict against a subset of the samples. Finding suitable parameters to enable training of the model was essential to facilitating the latter stages of the experiment. The testing which took place established specific ranges within the sample size and learning rates. The regions of increased accuracy were an indication that the sample sizes were providing suitable features for the model to generalise. There were trade-offs to be made in the learning rate, as it was demonstrated that there were extreme values at both higher and lower orders of magnitude where the model was not converging correctly.

This could be more of a comment on the quality of the images provided as the images themselves are incredibly diverse. Some images have very little tissue available to train a model on. If this is the case, then the author would suggest a more manual process in selecting images which provide as much clarity on the features required during pre-processing.

Manual selection of images could raise questions over the suitability of the usage of these images in the first place, as slide preparation techniques may vary between laboratories. In addition, slides are developed for the receptive fields of the human eye looking through a microscope. Therefore, this may not be the optimum preparation technique for image processing.

### 5.1.2 Stage 2

The second stage focused on the generalisation of the model and attempted to provide greater granularity of performance evaluation for models which displayed an ability to learn in the previous stage. Several evaluation methods were employed to assure maximum coverage, demonstrating transparency of the model's predictive process. The usage of both quantitative and qualitative evaluation by the author in order to rank models may not be the most rigorous technique for model selection as ranking can be skewed to support the researcher's view.

Where qualitative ranking was employed, the author chose the simplest criteria for evaluation to test the validity of this approach and encourage further exploration. The author believed that it was necessary to review attention maps for representative data as part of model evaluation. Limiting bias was crucial, so a simple test was chosen to ensure testing was objective and could be agreed by other readers of this research. The ranking that was established, favoured the maximisation of accurate predictions rather than a certain class. This research is focused on analysing the effect of parameter changes rather than the utilisation of this model within the domain of medical diagnosis. As a result, class weights were treated as equal rather than opting to maximise the number of cancer predictions which would be preferable for a doctor using the model.

Looking at the relationship between parameters and performance has highlighted the multi-variate issues with evaluation. The number of metrics that can be chosen to test a model rely largely on the domain within which the model is to be used. Even the fact that the same metric has different names used by different researchers is a frustrating to a researcher attempting to justify their research within the community. The idea for producing a ranking was born out of the multi-variate analysis that can take place. The author believes that as the black box of neural networks are explored in greater detail, the need for a protocol for evaluation needs to be established.

### 5.1.3 Stage 3

The third stage was implemented solely to test the stability of the models by testing the model against a larger dataset. Exposing the model to as much new data as possible confirms the resilience of the model. The accuracy of the models tested were an obvious issue which may be further evidence of data quality issues, preventing accurate discrimination between the features of both classes. However, the author was confident that the model was demonstrating stability in predictive outputs. In order to report on the original research question with integrity, it was important to ensure that the model was as stable as possible so that variations in predictive outputs could be correctly attributed to the parameter being tested at the time and not random noise within the model. This stage also served to validate decisions taken in previous stages.

### 5.1.4 Stage 4

Following the testing of the training sample sizes and learning rate hyperparameter, the next logical stage in the research was to revisit the neural network topology to further understand the relationship between the design parameters and the output of the model when subjected to the same evaluation set. There was still a question over which metric was best to evaluate the model sufficiently. It was decided that it was pertinent to continue to look at maximising the true predictions for each class as well as evaluating the attention mapping on the images. This provided a wealth of information to the researcher as to the effects of varying certain parameters. It was established that there was a trade-off to be made between the quality of the feature extraction and the ability to generalise as constantly increasing the number of feature maps lead to a decline in the accuracy of the model as the attention of the model shifted away from the tissue and toward the whitespace in the image.

This stage also allowed the researcher to explore further the idea that certain metrics such as AUC in the context of the ROC curves was not a suitable metric when equal weight was attributed in the ability of the model to distinguish between both classes. This research did not need to strike any sort of trade-off in favour of a particular class. It simply needed a stable model, capable of generalising across a larger sample set in order to test the effects of hyperparameter tuning. The author believes that whilst the accuracy of the model still leaves a lot to be desired compared to state of the art (section 2.10.3) and with the original research, (Angel Cruz-Roa, 2014) the model was stable enough to allow this analysis to take place.

# 5.2 Research Conclusions

To conclude, the research has provided a range of comparisons in the performance of numerous variations in convolutional neural network architecture for the classification of microscope slide images.

## 5.2.1 Reviewing the research question:

The research question being explored was:

*"*What is the relationship between neural network architecture parameters and the quality of detection of cancer in microscope slide images?*"*

#### *5.2.1.1 Conclusions*

#### *5.2.1.1.(a)* Sample size

Stage 1 of the experiment highlighted that the sample size was less import than the quality of the images being passed through the network. There was no relationship between the sample size and accuracy it was more that certain sample sizes happened to include more data which trained on certain features than others (e.g. whitespace vs tissue).

#### *5.2.1.1.(b)* Learning Rate

The learning rate proved to be vital for the ability of the network to obtain a higher degree of accuracy than random choice. The orders of magnitude selected suggested that a lower order (10^-3) value was more successful than a higher order counterpart. There was also evidence that if the learning rate was lowered too far then the model performance declined. This is presumably because the model is failing to converge around the appropriate weights for each node during training whilst he higher order learning rates suggested that the optimal values are being missed.

#### *5.2.1.1.(c)* Convolution Kernel Dimensions

The relationship between the kernel dimensions and the accuracy of a model to predict was that with increasing kernel dimensions, a loss in resolution in the feature maps was observed. This was evident in the attention mapping as a degradation in the differentiation between tissue regions and whitespace occurred when the kernel dimensions were increased too far.

#### *5.2.1.1.(d)* Number of Kernels in a layer

The number of kernels available on a particular convolutional layer corresponded with an increase in the resolution of heatmap images. This was not always a positive as it was found that certain feature maps were over training on the whitespace of an image as the number of kernels increased.

Overall, the results presented show that there is a balance to be struck in what is a multi-variate optimisation problem, creating a significant challenge to those looking to design a network for a new use case. This underpins the need for an iterative approach to experimentation as outlined in the methodology section as well as suitable performance evaluation metrics to be identified.

## 5.2.2 Review of Aims and Objectives

The research has resulted in the development of a product which has allowed for satisfactory testing and evaluation of both a single neural network model and models within various hyperparameter ranges to be tested and compared. This has achieved the main aim of the project. The objectives that were established at the start of the project were:

- To pre-process a sufficiently large (approx. 100k) dataset of images to be used as training and test data for consumption by a neural network before December.
    - This was achieved as demonstrated by the models tested
- To demonstrate an image classification system with an accuracy of at least greater than 50% (better than a random choice) before the end of the project.

    - Several model architectures achieved this goal with the highest achieving approx. 72% accuracy when tested across 62000 samples. However, questions were raised and explored as to what form 'accuracy' should take in this context and how a model can be trusted to be predicting based on the correct features in an image.
    - The accuracy was based on random choice and whilst the objective was achieved, when compared with the original research (Angel Cruz-Roa, 2014) the model would not be considered successful as the original accuracy is around 13% higher than the model produced. The multi-class models described (see section 2.10) also achieve far higher accuracy.
- To utilise the architecture of at least one deep learning design pattern (i.e Convolutional Neural Network) before the early prototype demo.
    - This was achieved with the successful implementation of a convolutional neural network
- To provide a comparison in accuracy between at least two deep learning parameters before the end of the project.
    - Sample size, Learning Rate, Convolution/kernel dimensions and the number of kernels used in a model were tested and results analysed, exceeding this objective's expectation.

### 5.2.3 Use Cases and Operational Considerations

The context within which the neural network architecture was tested obviously points to the application of cancer diagnostics. However, the research question and artefacts created seek to provide ideas and approaches which are applicable to the wider field of neural network design and evaluation.

It is evident from the best model identified that the architecture created is not sufficient for operational usage in cancer diagnostics. There is simply too much risk to patients for a model with an accuracy of 72% to be used and trusted in isolation. The very best that could be hoped is that the approaches to development and identifying models are adopted as a practice rather than the model proposed being utilised. It is also hoped that the discussions around performance evaluation feed into the wider discussion over any ethical concerns ascribed to the usage of neural networks in clinical decisions.

### 5.2.4 Research Strengths

Some of the strengths of this research are:

- The research provided offers a strong positive representation of the capability of the tools and frameworks in the rapid development of artificial intelligence models with several artefacts created to support the visualisation and evaluation of the neural network architecture.
- This research also offers a critical discussion on the need to understand the context and domain within which the model may be applied when selecting appropriate performance metrics by which the model can be evaluated.
- Another strength of the research is the wealth of data created during iterative cycles allowing a researcher to quickly identify trends in the data and highlight points for discussion in feedback sessions.

### 5.2.5 Research Limitations

Some potential limitations of the research are:

- The experimentation phases are focused solely on one neural network design pattern and as such it cannot be established that this particular design pattern is most suitable for this particular context.
- The models are only constructed and evaluated with one deep learning framework. There may be tools present in other frameworks which may have improved the network performance.
- One further limitation is that it was necessary to fix certain parameters due to the timescales for development and experimentation. As a result, the view presented may not represent a complete evaluation of the network parameters. This was a necessary trade-off for the sake of ensuring a working product could be developed in the time constraints of the project.

### 5.2.6 Suggestions for future work

The author wishes to present the following ideas of future work within this area:

- Expanding the number of network design patterns used to compare the performance of each.
- Using Generative Adversarial Network (Goodfellow *et al.,* 2014) architecture to generate slide images. This could be a potential method for understanding what it is that a neural network identifies as cancerous tissue vs normal tissue. This could be used as a comparison with the attention heatmaps as a way of ranking model performance in order to validate this method of evaluation.
- Expansion of the neural network architecture to include a greater number of layers to identify relationships in the types of layers and the accuracy of the model.

# CHAPTER 6: CRITICAL REVIEW

The following chapter offers a critical review of the project. The author reflects on the challenges faced, offers improvements and discusses the project management approach used.

## 6.1 Challenges

One of the main challenges of a project of this nature was the significant learning curve required to enter the world of artificial intelligence, from both a conceptual and technical perspective. The author had no previous experience with the neural networks and the frameworks used to create them. A significant amount of time had to be invested in the early stages to gain the fundamental understanding required to even attempt a project of this nature. The usage of the CNTK framework selected was also a particular challenge, as the majority of the information was from the documentation itself rather than community driven FAQs, making it harder to narrow down exactly what aspects of the library were useful for this particular application. Fortunately, some tutorials were available in the installation of CNTK with some boiler plate code which acted as a helpful starting point.

Another aspect of this type of project was the level of background research required. It was helpful in addressing the learning curve challenge, but offered its own challenge as it is difficult to identify relevant papers from the masses of research papers available. The author of this report had not conducted background research at this level before and found that a lot of time was spent simply finding and reading research papers, distracting from the development of the application itself.

Performance evaluation also presented a significant issue for the author. It has been outlined that several metrics are available depending on the context within which the research is taking place. Far more time was devoted to performance metric development than was originally planned for which reduced the amount of time which could be spent experimenting with different parameters. The analysis of the data created by these metrics also required more time than was originally allocated as the sheer volume of data created was underestimated by the author. Multi-variate analysis is not a simple problem to approach and finding a clear method which would provide meaningful insights was not easy.

## 6.2 Improvements

The author would suggest the following improvements to a project of this nature.

1. Choose a simpler application – a more well-established dataset would help a researcher by eliminating any concerns that the dataset is not of a high enough quality for training and testing. The images used in this project were from a single study and the range of images were far too diverse, even within the same class, to ensure that a well-trained model could be produced. This would also offer simpler validation with a larger range of state-of-the-art comparisons as the research will likely have benchmarked with same dataset.
2. Data augmentation – No data augmentation (see section 3.2.5.2) took place when training the images for this project so inclusion of this technique may provide some improvement in training
3. Feedback from the research community – It is easy to get trapped in a deluge of research papers with no clear path forward. The author believes that more frequent and greater inclusion with other researchers in the field at the end of each sprint would help to direct the researcher and eliminate unnecessary waste created through reading papers which may not be relevant for guidance. This is standard within modern software development. However, the constraints around the author's and other researcher's timetables conflicted with this approach.

## 6.3 Project management approach

The author believes that while the accuracy of the model may not be seen as successful in the eyes of the research community, the project management approach, creating a hybridisation of software development and experimental methodologies, provides some useful features. These are outlined below.

### 6.3.1 The Agile Mindset

For a project of this kind, having such many unknowns, the need to adapt to changes during development and testing were essential. This requires a mindset which embraces the ability to fail-fast and move onto the next approach. Throughout the development cycle, the author needed to be able to throw away useless code and failed model configurations whilst keeping up motivation. Embracing the failures rather than seeing them as a setback kept the momentum up throughout the different stages of the project.

### 6.3.2 Principles and Values

It is the author's opinion that agile development principles, values and practices (Beck, 2001) were essential to the successful creation of a neural network model. The "Research -> Write Report -> Code -> Test -> Review" sprint cycle proposed in this paper (section 3.3) allowed the author to maintain focus on code delivery as well as research and documentation, wherever possible. This is in-line with the Agile value of producing "working software over comprehensive documentation". Both aspects are required for this research, but the focus should always remain on the construction of a quality product. As mentioned above the need for "Responding to change" was also essential in successful delivery.

### 6.3.3 Agile science – iterative experimentation

The iterative cycles at different stages in the experimentation proved to be a useful artefact in narrowing down a model architecture which provided an example of a model trained on features in the image. It is hoped that this approach provides a practical demonstration of "Agile Science" (Hekler *et al.*, 2016). These rapid cycles with simple testing and evaluation allowed the author to gain a quick insight into the nature of the development process for neural networks, which could be refined in later stages.

## 6.4 Time management

The time management of this research was not ideal. Unfortunately, the research project was not the sole focus throughout the entire duration, as other project deadlines were required to take precedence throughout different stages. Whilst this was the case, the trade-off was necessary to facilitate successful completion of the other projects to a high standard. With the benefit of hindsight, the author would have attempted to carry out a greater amount of background research prior at the start of the project, to limit the impact of the challenges faced in terms of the learning curve and identifying relevant papers. This would have created far greater contingency when re-allocating time commitments to other projects.

# CHAPTER 7: Bibliography

Angel Cruz-Roa, A. B. (2014). "Automatic detection of invasive ductal carcinoma in whole slide images with convolutional neural networks," . *Medical Imaging 2014: Digital Pathology, 904103.* Proc. SPIE 9041, .

Beck, K. B. (2001). *Manifesto for Agile Software Development.* Retrieved from agilemanifesto Accesed 10 Feb 2019: http://agilemanifesto.org/

Breastcancer.org. (2019, March 9). *Invasive Ductal Carcinoma (IDC)* . Retrieved from Breastcancer.org accesed: (24th April 2019): https://www.breastcancer.org/symptoms/types/idc

Butler, S. (1863, June 13). *DARWIN AMONG THE MACHINES — [TO THE EDITOR OF THE PRESS, CHRISTCHURCH, NEW ZEALAND, 13 JUNE, 1863.]*. Retrieved from Victoria University of Wellington Library: http://nzetc.victoria.ac.nz/tm/scholarly/tei-ButFir-t1-g1-t1-g1-t4-body.html

Cancer Research UK. (2016). *Breast cancer statistics*. Retrieved from Cancer Research UK accessed: April 24th 2019: https://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/breast-cancer#heading-Zero

CNTK. (n.d.). *Python API for CNTK*. Retrieved from CNTK: https://cntk.ai/pythondocs/ (accessed 20 October 2018)

Forbes. (2016, 08 05). *Forbes*. Retrieved from This ist the cutting edge of deep learning research: https://www.forbes.com/sites/quora/2016/08/05/this-is-the-cutting-edge-of-deep-learning-research/#5aeb601951c8

Fukushima, K. (1975). Cognitron: a self-organizing multilayered neural network. . *Biol. Cybernetics 20* , 121-136 .

Gartner. (2016, August 19). *3 Trends Appear in the Gartner Hype Cycle for Emerging Technologies, 2016*. Retrieved from Gartner: https://www.gartner.com/smarterwithgartner/3-trends-appear-in-the-gartner-hype-cycle-for-emerging-technologies-2016/ (accessed 19 January 2019)

Gartner. (2017, August 15). *Gartner Identifies Three Megatrends That Will Drive Digital Business Into the Next Decade*. Retrieved from Gartner: https://www.gartner.com/en/newsroom/press-releases/2017-08-15-gartner-identifies-three-megatrends-that-will-drive-digital-business-into-the-next-decade (accessed 19 January 2019)

Gartner. (2018, August 16). *5 Trends Emerge in the Gartner Hype Cycle for Emerging Technologies, 2018*. Retrieved from Gartner: https://www.gartner.com/smarterwithgartner/5-trends-emerge-in-gartner-hype-cycle-for-emerging-technologies-2018/ (accessed 19 January 2019)

Kohavi, R. a. (1998). *Glossary of terms. Editorial for the Special Issue on Applications of Machine.* Boston: Kluwer Academic Publishers.

McCorduck, P. (2004). *Machines Who Think (2nd ed.).* Natick, MA: A. K. Peters, Ltd., ISBN 1-56881-205-1.

Microsoft. (2018). *Microsoft Azure Notebooks*. Retrieved from Microsoft Azure Notebooks (Accessed: 19 April 2019): https://notebooks.azure.com/

Mooney, P. T. (2018). *Predicting IDC in Breast Cancer Histology Images*. Retrieved from Kaggle: https://www.kaggle.com/paultimothymooney/predict-idc-in-breast-cancer-histology-images

Moore, G. (1965). “Cramming More Components onto Integrated Circuits,”. *Electronics, vol. 38, no. 8,* 114–117.

New York Times. (1958, July 13). *Electronic 'Brain' Teaches Itself*. Retrieved from New York Times Accessed: 27 April 2019: https://www.nytimes.com/1958/07/13/archives/electronic-brain-teaches-itself.html

Rumelhart, D. H. (1985). *Learning internal representations by error propagation (No. ICS-8506).* California University San Diego LA Jolla Inst. for Cognitive Science.

Schwaber, K. (1995). “Scrum development process”. *OOPSLA’95 Workshop on Business Object Design and Implementation.* Austin, TX: Springer-Verlag, pp. 117-134. .

Seide, F. &. ( 2016). *CNTK: Microsoft's Open-Source Deep-Learning Toolkit.* . Microsoft 2135-2135. 10.1145/2939672.2945397.

Tesla. (2016, October 19). *All Tesla cars being produced now have full self-driving hardware*. Retrieved from Tesla: https://www.tesla.com/blog/all-tesla-cars-being-produced-now-have-full-self-driving-hardware

Abadi, M. *et al.* (2016) 'TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems', .

Adya, M. and Collopy, F. (1998) 'How effective are neural networks at forecasting and prediction? A review and evaluation', *Journal of Forecasting,* 17(5), pp. 481-495.

Alshamrani, A. and Bahattab, A. (2015) 'A Comparison Between Three SDLC Models Waterfall Model, Spiral Model, and Incremental/Iterative Model', *International Journal of Computer Science Issues (IJCSI),* 12(1), pp. 106-111.

Aresta, G. *et al.* (2018) 'BACH: Grand Challenge on Breast Cancer Histology Images', .

Butler, S. (1999) *Erewhon.*

Cantor, S.B. *et al.* (1999) 'A Comparison of C/B Ratios from Studies Using Receiver Operating Characteristic Curve Analysis', *Journal of Clinical Epidemiology,* 52(9), pp. 885-892.

Esmaeilzadeh, H. *et al.* (2012) 'Looking back and looking forward: power, performance, and upheaval', *Communications of the ACM,* 55(7), pp. 105-114.

Feng-Hsiung Hsu (1999) 'IBM's Deep Blue Chess grandmaster chips', *Micro, IEEE,* 19(2), pp. 70-81.

Fitch, F.B. (1944) *McCulloch Warren S. and Pitts Walter. A logical calculus of the ideas immanent in nervous activity. Bulletin of mathematical biophysics, vol. 5 (1943), pp. 115–133.*

Fraz, M.M. *et al.* (2012) 'Blood vessel segmentation methodologies in retinal images – A survey', *Computer Methods and Programs in Biomedicine,* 108(1), pp. 407-433.

Fukushima, K., Miyake, S. and Ito, T. (1983) 'Neocognitron: A neural network model for a mechanism of visual pattern recognition', *Systems, Man and Cybernetics, IEEE Transactions On,* SMC-13(5), pp. 826-834.

Golatkar, A., Anand, D. and Sethi, A. (2018) *Classification of Breast Cancer Histology Using Deep Learning.*

Goodfellow, I.J. *et al.* (2014) 'Generative Adversarial Networks', .

Hekler, E. *et al.* (2016) 'Agile science: creating useful products for behavior change in the real world', *Translational Behavioral Medicine,* 6(2), pp. 317-328.

Hernández-Orallo, J. (2017) 'Evaluation in artificial intelligence: from task-oriented to ability-oriented measurement', *Artificial Intelligence Review,* 48(3), pp. 397-447.

Hubel, D.H. and Wiesel, T.N. (1959) 'Receptive fields of single neurones in the cat's striate cortex', *The Journal of Physiology,* 148(3), pp. 574-591.

Hunt, E., Minsky, M. and Papert, S. (1971) *Perceptrons.*

Hyman, P. (2014) 'Speech-to-speech translations stutter, but researchers see mellifluous future', *Communications of the ACM,* 57(4), pp. 16-19.

Iesmantas, T. and Alzbutas, R. (2018) *Convolutional Capsule Network for Classification of Breast Cancer Histology Images.*

Japkowicz, N. (2001) 'Supervised Versus Unsupervised Binary-Learning by Feedforward Neural Networks', *Machine Learning,* 42(1), pp. 97-122.

Jetley, S. *et al.* (2018) 'Learn To Pay Attention', .

Kell, D.B. and Oliver, S.G. (2004) 'Here is the evidence, now what is the hypothesis? The complementary roles of inductive and hypothesis-driven science in the post-genomic era', *BioEssays,* 26(1), pp. 99-105.

Kotu, V. (2015) *Predictive analytics and data mining : concepts and practice with rapidminer.* Waltham, Massachusetts : Elsevier Inc.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2017) 'ImageNet classification with deep convolutional neural networks', *Communications of the ACM,* 60(6), pp. 84-90.

Krizhevsky, A., Sutskever, I. and Hinton, G. (2017) 'ImageNet classification with deep convolutional neural networks', *Communications of the ACM,* 60(6), pp. 84-90.

Krohs, U. (2012) 'Convenience experimentation', *Studies in History and Philosophy of Biol & Biomed Sci,* 43(1), pp. 52-57.

Kwok, S. (2018) *Multiclass Classification of Breast Cancer in Whole-Slide Images.*

Lecun, Y. *et al.* (1989) 'HANDWRITTEN DIGIT RECOGNITION - APPLICATIONS OF NEURAL NETWORK CHIPS AND AUTOMATIC LEARNING', *Ieee Communications Magazine,* 27(11), pp. 41-46.

Leonelli, S. (2012a) *Introduction: Making sense of data-driven research in the biological and biomedical sciences.*

Leonelli, S. (2012b) 'Introduction: Making sense of data-driven research in the biological and biomedical sciences', *Studies in History and Philosophy of Biol & Biomed Sci,* 43(1), pp. 1-3.

Liu, C. *et al.* (2016) 'Attention Correctness in Neural Image Captioning', .

Lobo, J.M., Jiménez-Valverde, A. and Real, R. (2008) 'AUC: a misleading measure of the performance of predictive distribution models', *Global Ecology and Biogeography,* 17(2), pp. 145-151.

Murphy, M.P. *et al.* (2011) 'The LittleDog robot', *The International Journal of Robotics Research,* 30(2), pp. 145-149.

Nawaz, W.*et al.* (2018) *Classification Of Breast Cancer Histology Images Using ALEXNET.*

Ohno, T. (1988) *Toyota production system: beyond large-scale production.* crc Press.

Olazaran, M. (1996) 'A Sociological Study of the Official History of the Perceptrons Controversy', *Social Studies of Science,* 26(3), pp. 611-659.

Perry, T.S. (2018) 'Move over, Moore's law. Make way for Huang's law Spectral Lines]', *Spectrum, IEEE,* 55(5), pp. 7.

Robbins, H. and Monro, S. (1951) 'A Stochastic Approximation Method', *The Annals of Mathematical Statistics,* 22(3), pp. 400-407.

Rosenblatt, F. (1958) 'The perceptron: A probabilistic model for information storage and organization in the brain', *Psychological Review,* 65(6), pp. 386-408.

Saito, T. and Rehmsmeier, M. (2015) 'The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets', *PLoS ONE,* 10(3), pp. e0118432.

Schwaber, K. (2000) *Agile Project Management with Scrum.* United States: Pearson Education.

Singh, J. (2017) *Facebook, Microsoft build open standard for different AI frameworks.* New Dehli:.

Sun, T.Q. and Medaglia, R. (2018) 'Mapping the challenges of Artificial Intelligence in the public sector: Evidence from public healthcare', *Government Information Quarterly,* .

Swets, J.A. (1988) 'Measuring the accuracy of diagnostic systems', *Science (New York, N.Y.),* 240(4857), pp. 1285-1293.

Turing, A.M. (1950) 'Computing Machinery and Intelligence', *Mind,* 59(236), pp. 433-460.

Von Neumann, J. (1993) 'First draft of a report on the EDVAC', *Annals of the History of Computing, IEEE,* 15(4), pp. 27-75.

Welikala, R.A.*et al.* (2017) *Automated arteriole and venule classification using deep learning for retinal images from the UK Biobank cohort.*

Zhang, Z. *et al.* (2018a) 'Deep learning in omics: a survey and guideline', *Briefings in Functional Genomics,* .

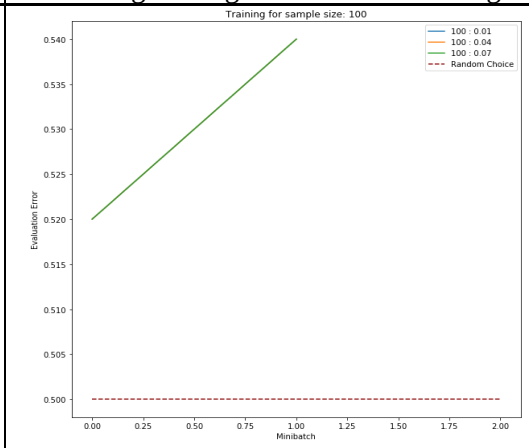
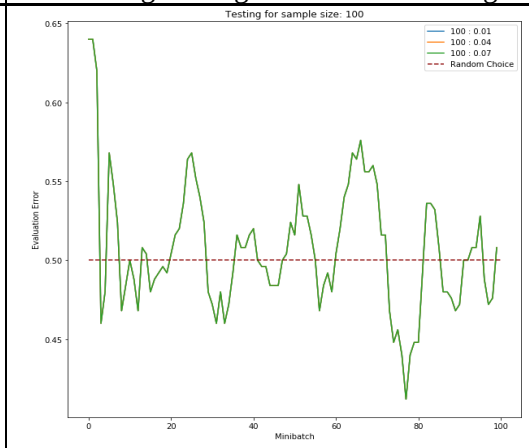
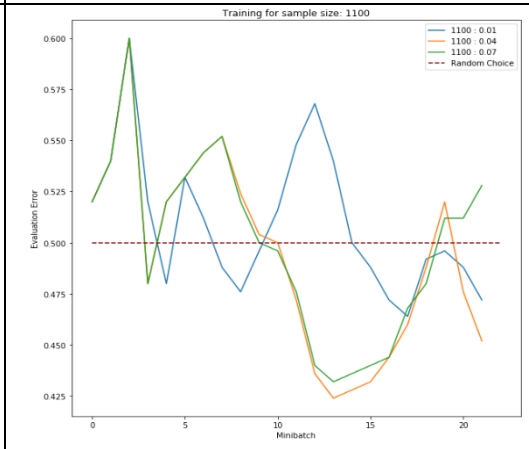
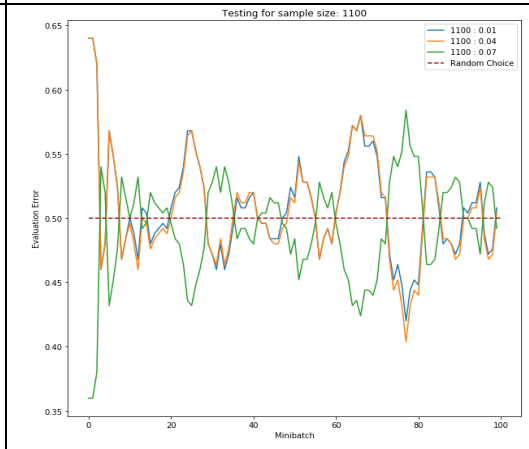
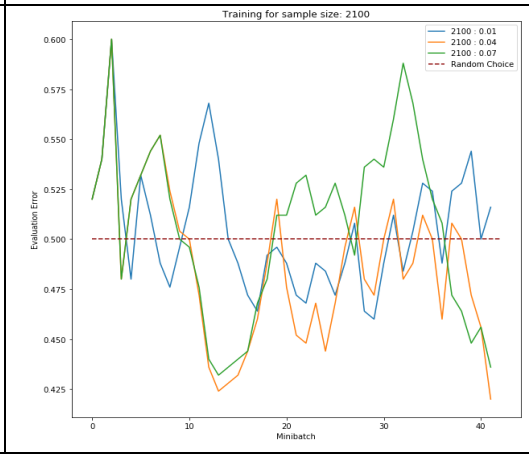
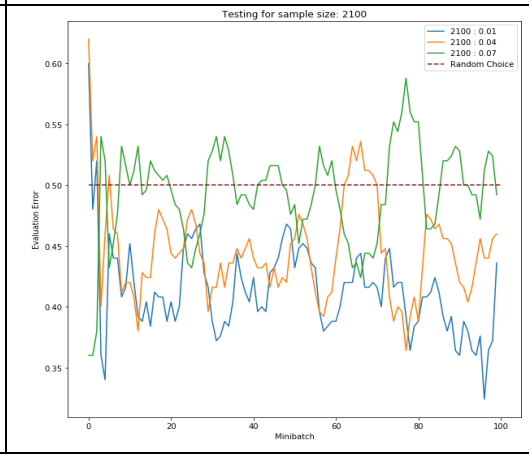
Zhang, Z. *et al.* (2018b) 'Deep learning in omics: a survey and guideline', *Briefings in Functional Genomics,* .

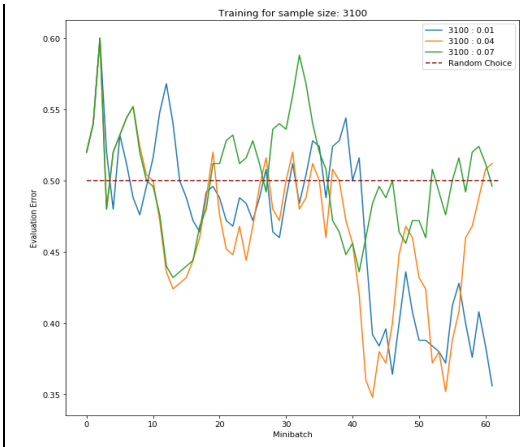
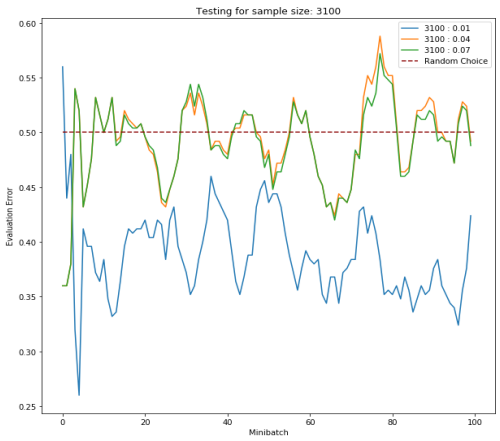
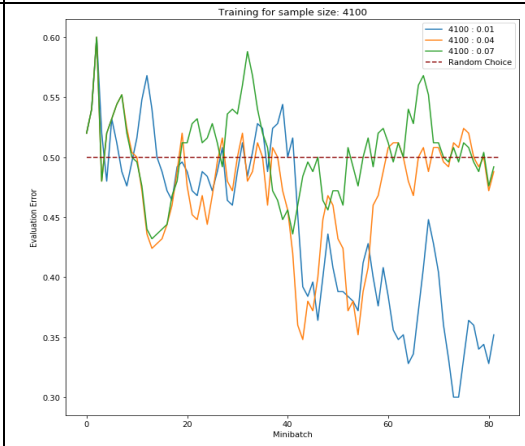
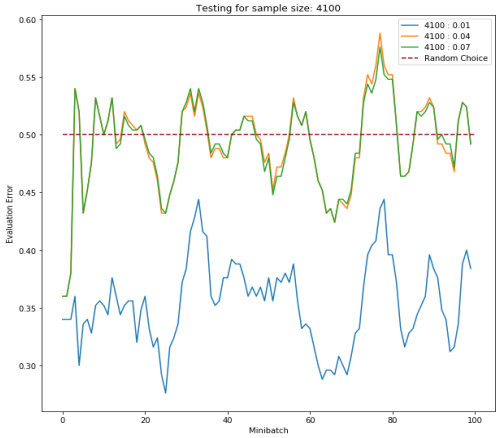
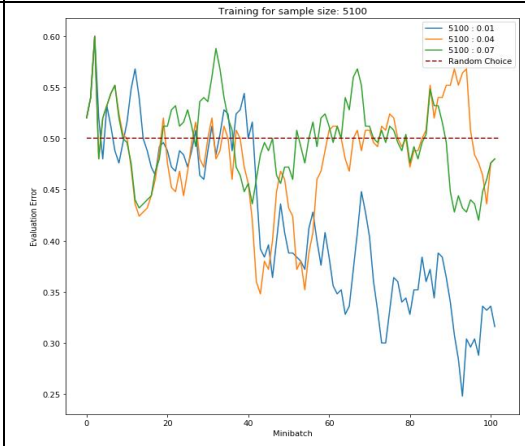
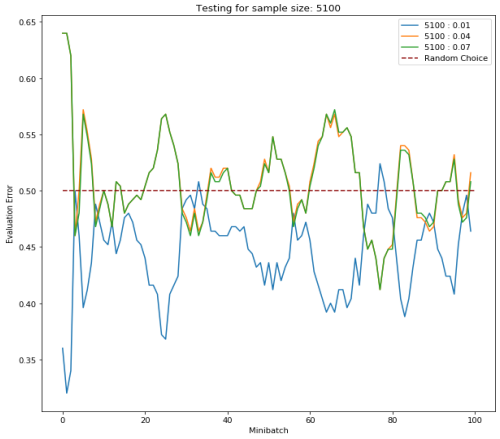
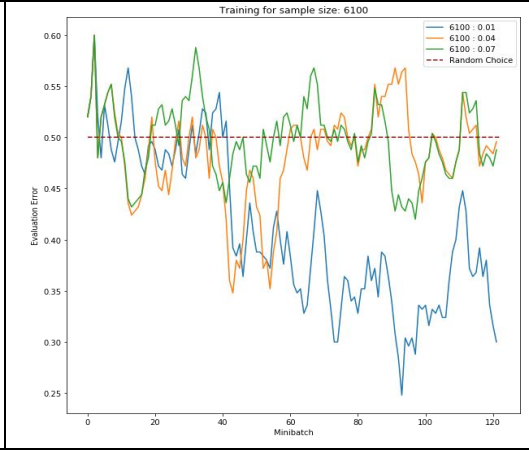
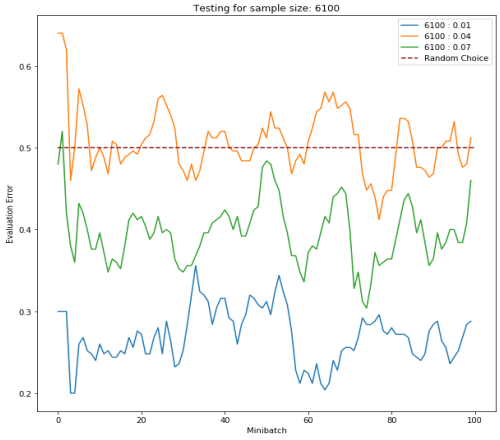
Zhou, J. *et al.* (2018) 'Graph Neural Networks: A Review of Methods and Applications', .

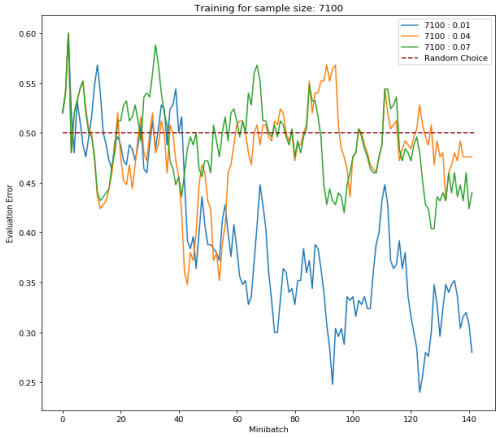
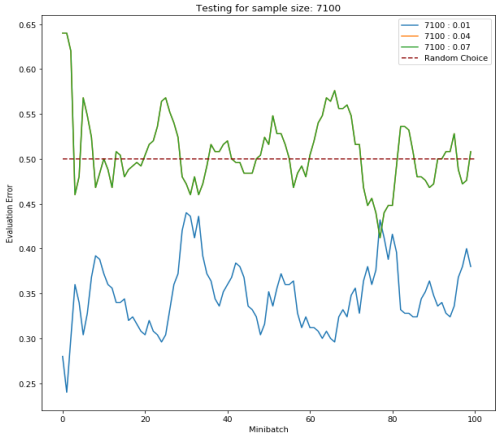
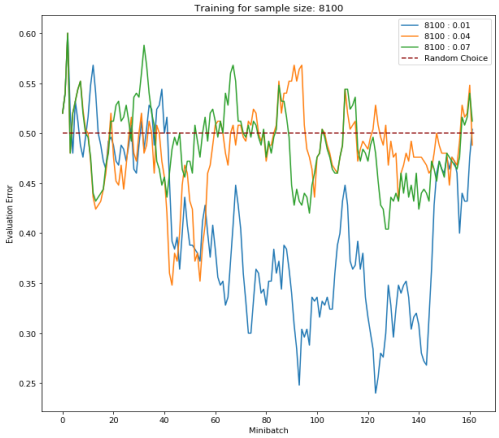
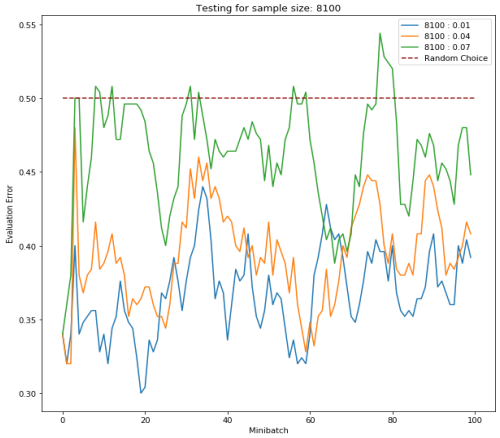
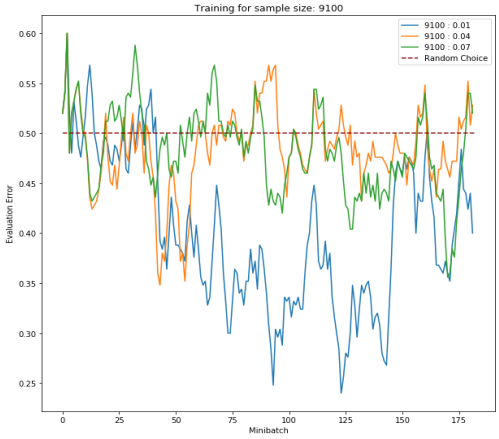
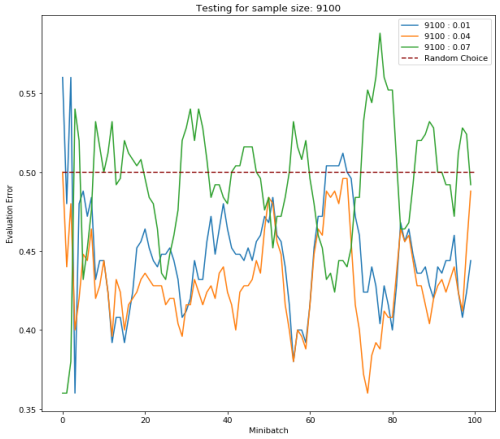
# CHAPTER 8: APPENDIX

## 8.1 Appendix I – Iterative cycles sample size and learning rate

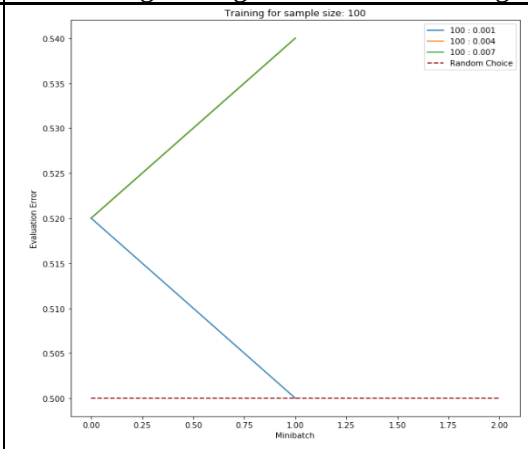
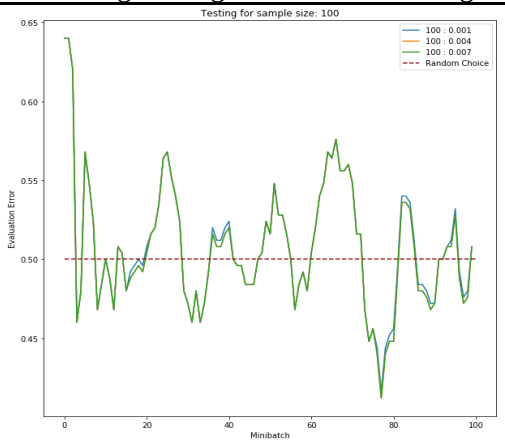
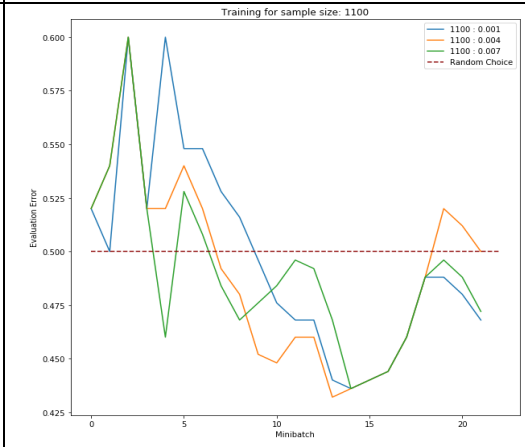
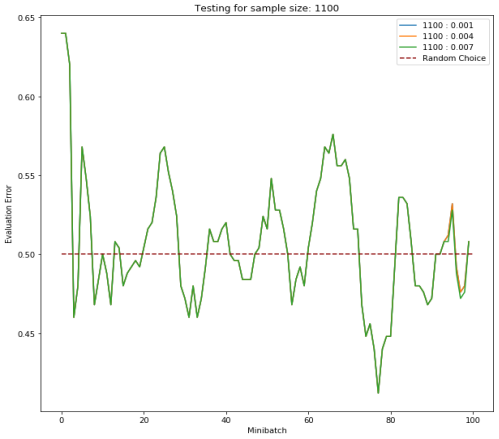
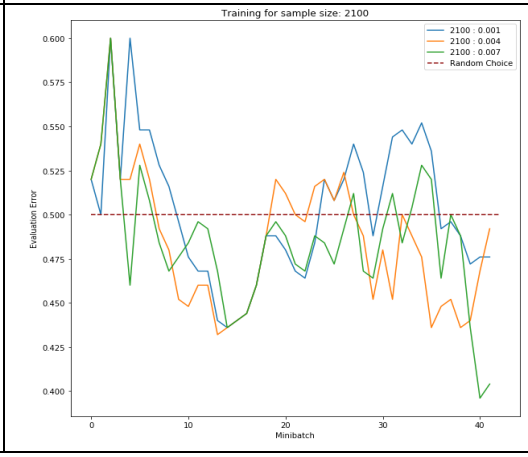
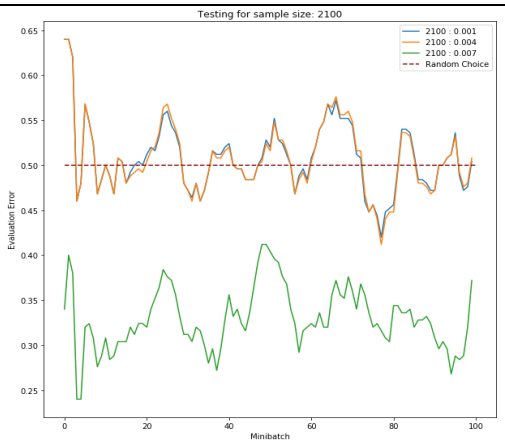
### 8.1.1 Iterative Cycle – Leaning rate 10^-2 range

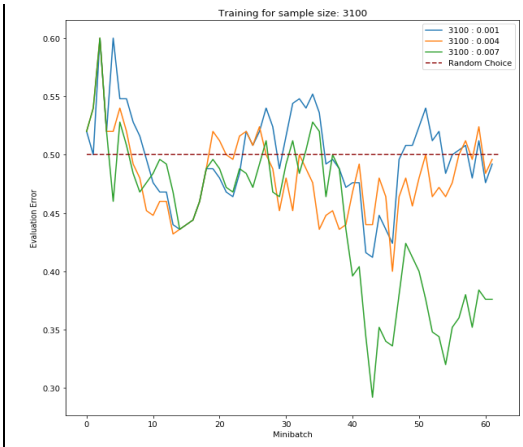
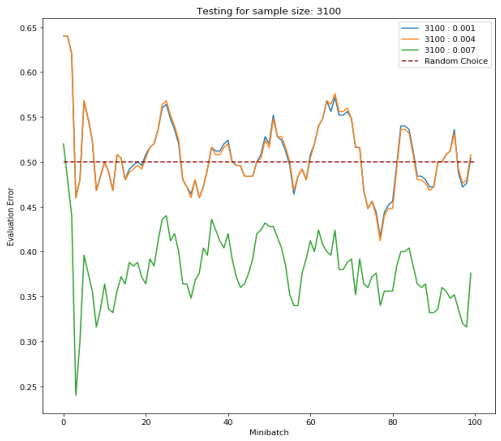
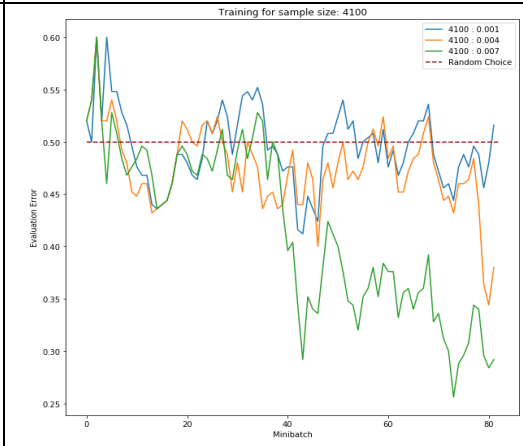
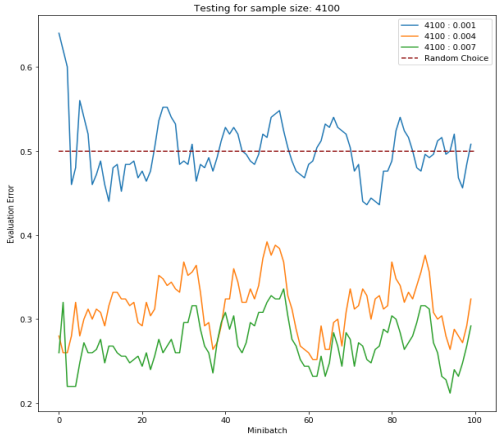
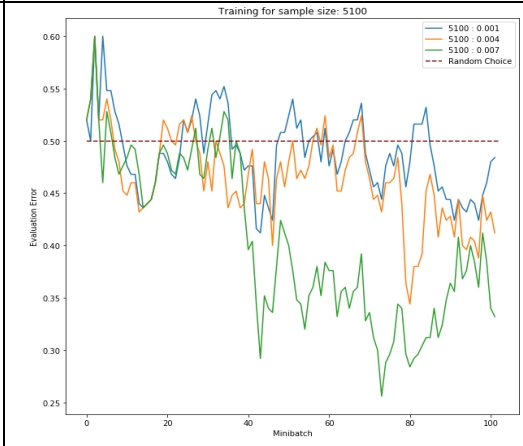
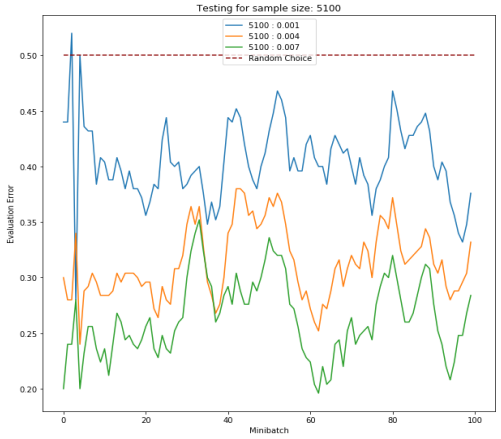
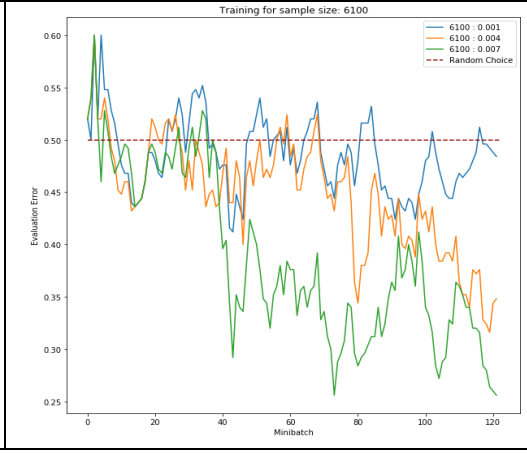
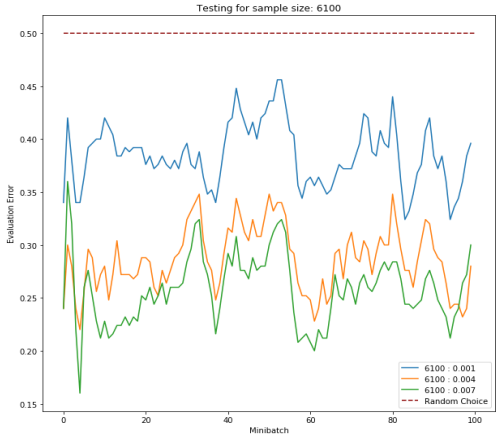
| Sample Size | Moving Average Error Rate -Training | Moving Average Error Rate -Testing |
|---|---|---|
| 100 |  |  |
| 100 |  |  |
| 2100 |  |  |

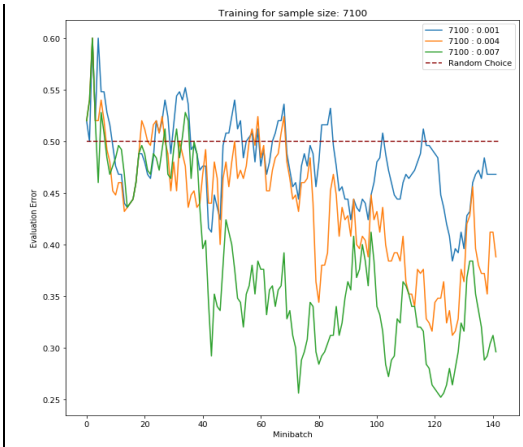
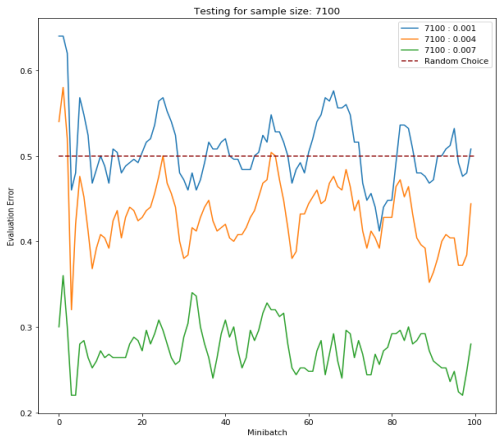
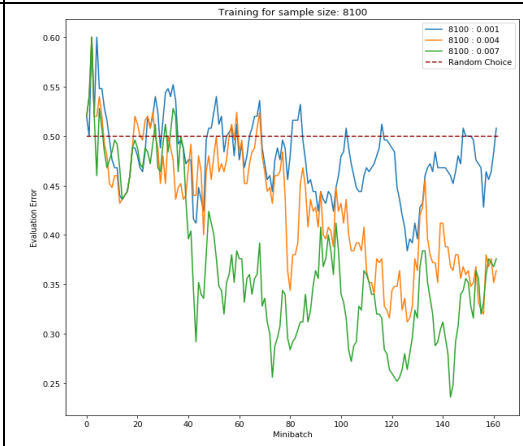
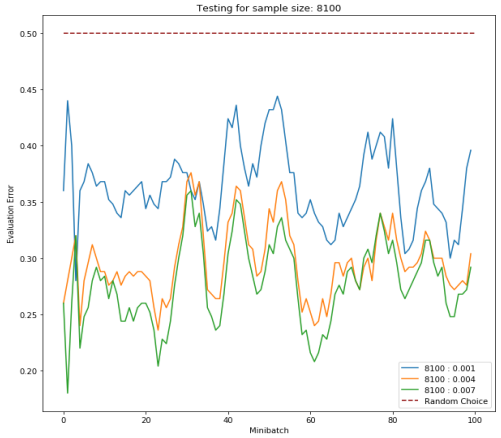
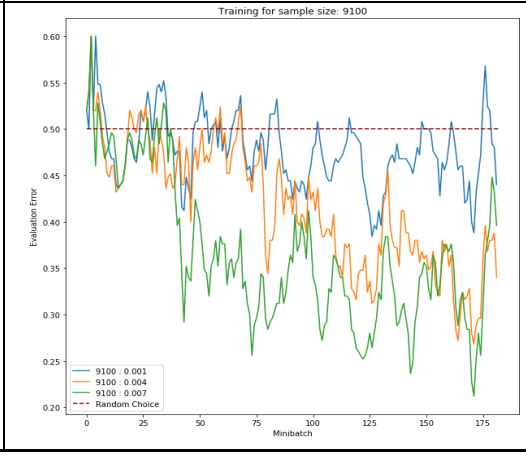
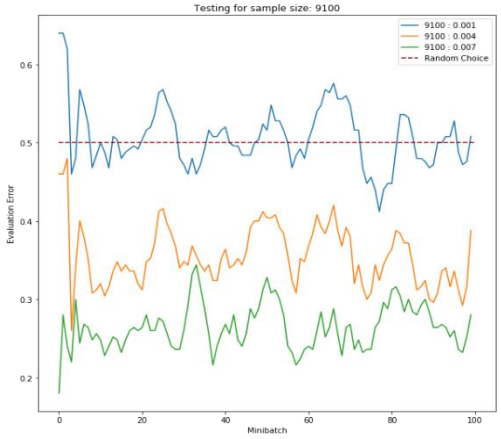
| | | |
|---|---|---|
| 3100 |  |  |
| 4100 |  |  |
| 5100 |  |  |
| 6100 |  |  |

| | | |
|---|---|---|
| 7100 |  |  |
| 8100 |  |  |
| 9100 |  |  |

## 8.1.2 Iterative Cycle – Leaning rate 10^-3 range

| Sample Size | Moving Average Error Rate -Training | Moving Average Error Rate -Testing |
|---|---|---|
| 100 |  |  |
| 1100 |  |  |
| 2100 |  |  |

| | | |
|---|---|---|
| 3100 |  |  |
| 4100 |  |  |
| 5100 |  |  |
| 6100 |  |  |

| | | |
|---|---|---|
| 7100 |  |  |
| 8100 |  |  |
| 9100 |  |  |

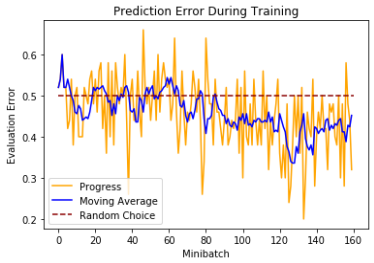
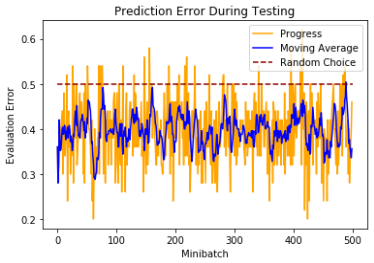
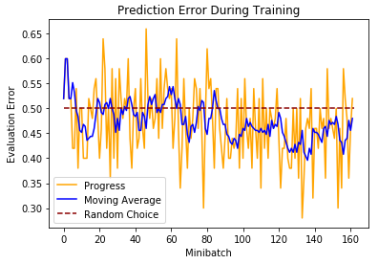
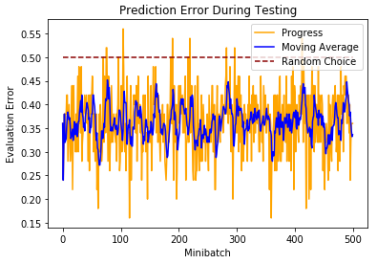
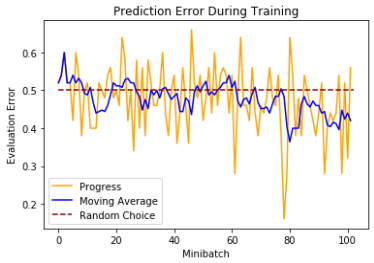
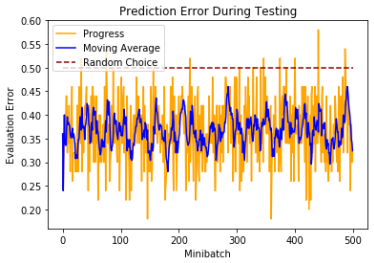
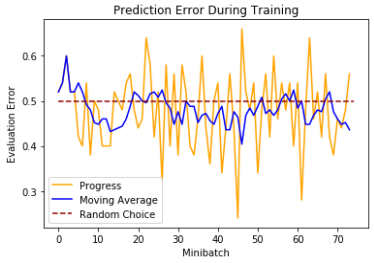
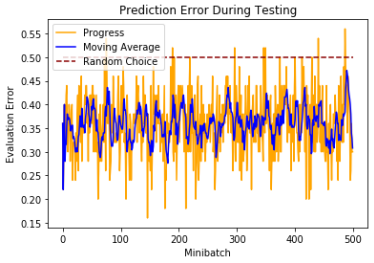
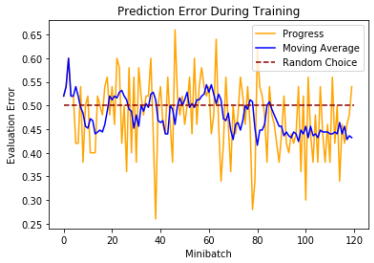
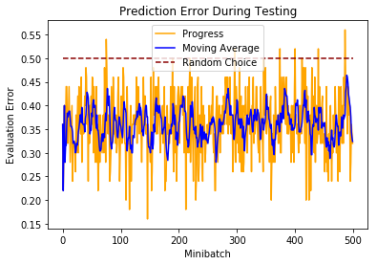
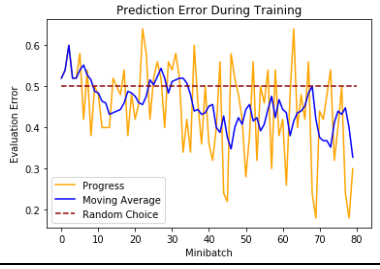
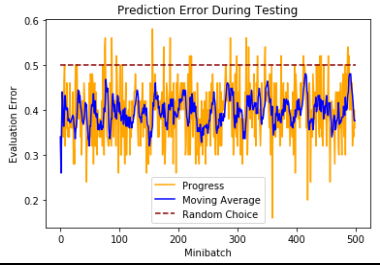
# 8.2 Appendix II – Macro Analysis Raw Data

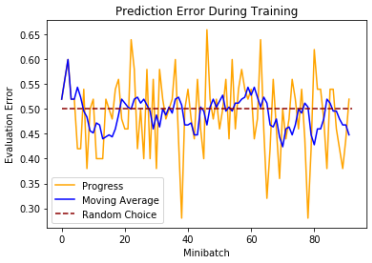
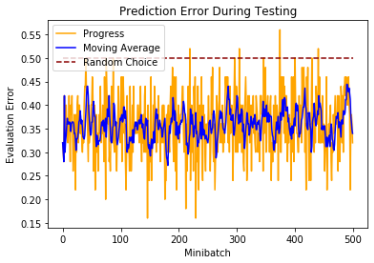
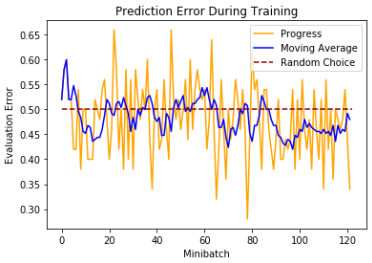
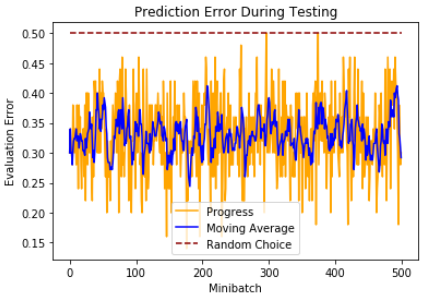
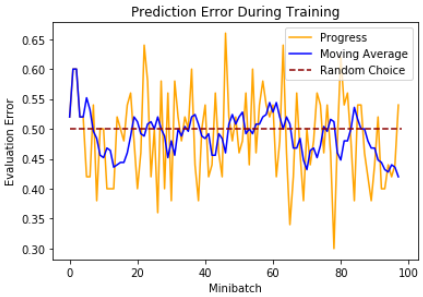
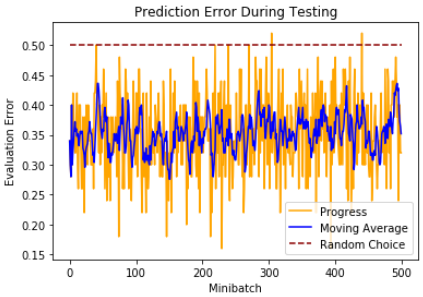
| numeric index | Index | True Cancer | True Normal | False Cancer | False Normal | % Correct Answers | numeric index | Index | True Cancer | True Normal | False Cancer | False Normal | % Correct Answers |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 100 - LR = 0.001 | 0 | 2460 | 2534 | 6 | 49.2 | 113 | 5600 - LR = 0.0025 | 0 | 2466 | 2534 | 0 | 49.32 |
| 2 | 100 - LR = 0.004 | 0 | 2466 | 2534 | 0 | 49.32 | 114 | 5600 - LR = 0.004 | 3 | 2466 | 2531 | 0 | 49.38 |
| 3 | 100 - LR = 0.007 | 0 | 2466 | 2534 | 0 | 49.32 | 115 | 5600 - LR = 0.0055 | 1 | 2466 | 2533 | 0 | 49.34 |
| 4 | 1100 - LR = 0.001 | 0 | 2465 | 2534 | 1 | 49.3 | 116 | 5700 - LR = 0.001 | 1 | 2461 | 2533 | 5 | 49.24 |
| 5 | 1100 - LR = 0.004 | 0 | 2465 | 2534 | 1 | 49.3 | 117 | 5700 - LR = 0.0025 | 63 | 2440 | 2471 | 26 | 50.06 |
| 6 | 1100 - LR = 0.007 | 0 | 2466 | 2534 | 0 | 49.32 | 118 | 5700 - LR = 0.004 | 3 | 2466 | 2531 | 0 | 49.38 |
| 7 | 2100 - LR = 0.001 | 8 | 2455 | 2526 | 11 | 49.26 | 119 | 5700 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 |
| 8 | 2100 - LR = 0.004 | 0 | 2466 | 2534 | 0 | 49.32 | 120 | 5900 - LR = 0.001 | 0 | 2465 | 2534 | 1 | 49.3 |
| 9 | 2100 - LR = 0.007 | 76 | 2448 | 2458 | 18 | 50.48 | 121 | 5900 - LR = 0.0025 | 0 | 2466 | 2534 | 0 | 49.32 |
| 10 | 3100 - LR = 0.001 | 5 | 2458 | 2529 | 8 | 49.26 | 122 | 5900 - LR = 0.004 | 0 | 2466 | 2534 | 0 | 49.32 |
| 11 | 3100 - LR = 0.0025 | 0 | 2465 | 2534 | 1 | 49.3 | 123 | 5900 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 |
| 12 | 3100 - LR = 0.004 | 0 | 2466 | 2534 | 0 | 49.32 | 124 | 6000 - LR = 0.001 | 892 | 1967 | 1642 | 499 | 57.18 |
| 13 | 3100 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 | 125 | 6000 - LR = 0.0015 | 1868 | 1303 | 666 | 1163 | 63.42 |
| 14 | 3100 - LR = 0.007 | 0 | 2466 | 2534 | 0 | 49.32 | 126 | 6000 - LR = 0.0017 | 1984 | 1208 | 550 | 1258 | 63.84 |
| 15 | 3700 - LR = 0.001 | 987 | 1810 | 1547 | 656 | 55.94 | 127 | 6000 - LR = 0.0019 | 2009 | 1208 | 525 | 1258 | 64.34 |
| 16 | 3700 - LR = 0.0025 | 2437 | 262 | 97 | 2204 | 53.98 | 128 | 6000 - LR = 0.0021 | 1978 | 1277 | 556 | 1189 | 65.1 |
| 18 | 3700 - LR = 0.0027 | 2422 | 340 | 112 | 2126 | 55.24 | 129 | 6000 - LR = 0.0023 | 1916 | 1409 | 618 | 1057 | 66.5 |
| 19 | 3700 - LR = 0.0029 | 2385 | 449 | 149 | 2017 | 56.68 | 130 | 6000 - LR = 0.0025 | 1810 | 1542 | 724 | 924 | 67.04 |
| 20 | 3700 - LR = 0.0031 | 2329 | 618 | 205 | 1848 | 58.94 | 132 | 6000 - LR = 0.0027 | 1677 | 1691 | 857 | 775 | 67.36 |
| 21 | 3700 - LR = 0.0033 | 2238 | 820 | 296 | 1646 | 61.16 | 133 | 6000 - LR = 0.0029 | 1496 | 1851 | 1038 | 615 | 66.94 |
| 22 | 3700 - LR = 0.0035 | 2133 | 1075 | 401 | 1391 | 64.16 | 134 | 6000 - LR = 0.0031 | 1302 | 1983 | 1232 | 483 | 65.7 |
| 23 | 3700 - LR = 0.0037 | 1974 | 1337 | 560 | 1129 | 66.22 | 135 | 6000 - LR = 0.0033 | 1120 | 2101 | 1414 | 365 | 64.42 |
| 24 | 3700 - LR = 0.0039 | 1791 | 1584 | 743 | 882 | 67.5 | 136 | 6000 - LR = 0.0035 | 904 | 2195 | 1630 | 271 | 61.98 |
| 25 | 3700 - LR = 0.004 | 1672 | 1700 | 862 | 766 | 67.44 | 137 | 6000 - LR = 0.0037 | 714 | 2258 | 1820 | 208 | 59.44 |
| 26 | 3700 - LR = 0.0041 | 1551 | 1809 | 983 | 657 | 67.2 | 138 | 6000 - LR = 0.0039 | 503 | 2317 | 2031 | 149 | 56.4 |
| 27 | 3700 - LR = 0.0043 | 1306 | 1980 | 1228 | 486 | 65.72 | 139 | 6000 - LR = 0.004 | 427 | 2345 | 2107 | 121 | 55.44 |
| 28 | 3700 - LR = 0.0045 | 1039 | 2127 | 1495 | 339 | 63.32 | 140 | 6000 - LR = 0.0055 | 4 | 2466 | 2530 | 0 | 49.4 |
| 29 | 3700 - LR = 0.0047 | 789 | 2224 | 1745 | 242 | 60.26 | 141 | 6100 - LR = 0.001 | 1620 | 1442 | 914 | 1024 | 61.24 |
| 30 | 3700 - LR = 0.0049 | 555 | 2302 | 1979 | 164 | 57.14 | 142 | 6100 - LR = 0.0015 | 1627 | 1587 | 907 | 879 | 64.28 |
| 31 | 3700 - LR = 0.0051 | 357 | 2360 | 2177 | 106 | 54.34 | 143 | 6100 - LR = 0.0017 | 1551 | 1679 | 983 | 787 | 64.6 |
| 32 | 3700 - LR = 0.0053 | 192 | 2400 | 2342 | 66 | 51.84 | 144 | 6100 - LR = 0.0019 | 1484 | 1765 | 1050 | 701 | 64.98 |
| 33 | 3700 - LR = 0.0055 | 87 | 2434 | 2447 | 32 | 50.42 | 145 | 6100 - LR = 0.0021 | 1400 | 1857 | 1134 | 609 | 65.14 |
| 34 | 4000 - LR = 0.001 | 2521 | 38 | 13 | 2428 | 51.18 | 146 | 6100 - LR = 0.0023 | 1328 | 1936 | 1206 | 530 | 65.28 |
| 35 | 4000 - LR = 0.0025 | 2532 | 11 | 2 | 2455 | 50.86 | 147 | 6100 - LR = 0.0025 | 1223 | 2004 | 1311 | 462 | 64.54 |
| 36 | 4000 - LR = 0.004 | 2494 | 191 | 40 | 2275 | 53.7 | 149 | 6100 - LR = 0.0027 | 1127 | 2099 | 1407 | 367 | 64.52 |
| 37 | 4000 - LR = 0.0041 | 2479 | 273 | 55 | 2193 | 55.04 | 150 | 6100 - LR = 0.0029 | 1012 | 2162 | 1522 | 304 | 63.48 |
| 38 | 4000 - LR = 0.0043 | 2449 | 426 | 85 | 2040 | 57.5 | 151 | 6100 - LR = 0.0031 | 885 | 2226 | 1649 | 240 | 62.22 |
| 39 | 4000 - LR = 0.0045 | 2392 | 609 | 142 | 1857 | 60.02 | 152 | 6100 - LR = 0.0033 | 775 | 2256 | 1759 | 210 | 60.62 |
| 40 | 4000 - LR = 0.0047 | 2306 | 790 | 228 | 1676 | 61.92 | 153 | 6100 - LR = 0.0035 | 661 | 2285 | 1873 | 181 | 58.92 |
| 41 | 4000 - LR = 0.0049 | 2221 | 1018 | 313 | 1448 | 64.78 | 154 | 6100 - LR = 0.0037 | 540 | 2317 | 1994 | 149 | 57.14 |
| 42 | 4000 - LR = 0.0051 | 2097 | 1204 | 437 | 1262 | 66.02 | 155 | 6100 - LR = 0.0039 | 422 | 2358 | 2112 | 108 | 55.6 |
| 43 | 4000 - LR = 0.0053 | 1927 | 1426 | 607 | 1040 | 67.06 | 156 | 6100 - LR = 0.004 | 347 | 2373 | 2187 | 93 | 54.4 |

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 44 | 4000 - LR = 0.0055 | 1707 | 1621 | 827 | 845 | 66.56 | 157 | 6100 - LR = 0.0055 | 2 | 2466 | 2532 | 0 | 49.36 |
| 46 | 4000 - LR = 0.0057 | 1524 | 1775 | 1010 | 691 | 65.98 | 158 | 6100 - LR = 0.007 | 0 | 2466 | 2534 | 0 | 49.32 |
| 47 | 4000 - LR = 0.0059 | 1400 | 1876 | 1134 | 590 | 65.52 | 159 | 6300 - LR = 0.001 | 38 | 2437 | 2496 | 29 | 49.5 |
| 48 | 4100 - LR = 0.001 | 114 | 2368 | 2420 | 98 | 49.64 | 160 | 6300 - LR = 0.0025 | 4 | 2465 | 2530 | 1 | 49.38 |
| 49 | 4100 - LR = 0.0025 | 2331 | 596 | 203 | 1870 | 58.54 | 161 | 6300 - LR = 0.004 | 4 | 2464 | 2530 | 2 | 49.36 |
| 50 | 4100 - LR = 0.004 | 571 | 2277 | 1963 | 189 | 56.96 | 162 | 6300 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 |
| 51 | 4100 - LR = 0.0055 | 40 | 2452 | 2494 | 14 | 49.84 | 163 | 7100 - LR = 0.001 | 0 | 2466 | 2534 | 0 | 49.32 |
| 52 | 4100 - LR = 0.007 | 0 | 2466 | 2534 | 0 | 49.32 | 164 | 7100 - LR = 0.0025 | 0 | 2466 | 2534 | 0 | 49.32 |
| 53 | 4300 - LR = 0.001 | 76 | 2401 | 2458 | 65 | 49.54 | 165 | 7100 - LR = 0.004 | 0 | 2466 | 2534 | 0 | 49.32 |
| 54 | 4300 - LR = 0.0025 | 2377 | 482 | 157 | 1984 | 57.18 | 166 | 7100 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 |
| 55 | 4300 - LR = 0.004 | 387 | 2358 | 2147 | 108 | 54.9 | 167 | 7100 - LR = 0.007 | 0 | 2466 | 2534 | 0 | 49.32 |
| 56 | 4300 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 | 168 | 7700 - LR = 0.001 | 0 | 2466 | 2534 | 0 | 49.32 |
| 57 | 4600 - LR = 0.001 | 6 | 2458 | 2528 | 8 | 49.28 | 169 | 7700 - LR = 0.0025 | 0 | 2466 | 2534 | 0 | 49.32 |
| 58 | 4600 - LR = 0.002 | 1699 | 1497 | 835 | 969 | 63.92 | 170 | 7700 - LR = 0.004 | 0 | 2466 | 2534 | 0 | 49.32 |
| 59 | 4600 - LR = 0.0022 | 1724 | 1501 | 810 | 965 | 64.5 | 171 | 7700 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 |
| 60 | 4600 - LR = 0.0024 | 1657 | 1609 | 877 | 857 | 65.32 | 172 | 8000 - LR = 0.001 | 2213 | 715 | 321 | 1751 | 58.56 |
| 61 | 4600 - LR = 0.0025 | 1603 | 1654 | 931 | 812 | 65.14 | 173 | 8000 - LR = 0.0015 | 2336 | 490 | 198 | 1976 | 56.52 |
| 62 | 4600 - LR = 0.0026 | 1532 | 1717 | 1002 | 749 | 64.98 | 174 | 8000 - LR = 0.0017 | 2322 | 542 | 212 | 1924 | 57.28 |
| 63 | 4600 - LR = 0.0028 | 1321 | 1890 | 1213 | 576 | 64.22 | 175 | 8000 - LR = 0.0019 | 2290 | 621 | 244 | 1845 | 58.22 |
| 64 | 4600 - LR = 0.003 | 1056 | 2062 | 1478 | 404 | 62.36 | 176 | 8000 - LR = 0.0021 | 2234 | 802 | 300 | 1664 | 60.72 |
| 65 | 4600 - LR = 0.0032 | 762 | 2205 | 1772 | 261 | 59.34 | 177 | 8000 - LR = 0.0023 | 2140 | 1016 | 394 | 1450 | 63.12 |
| 66 | 4600 - LR = 0.0034 | 502 | 2308 | 2032 | 158 | 56.2 | 178 | 8000 - LR = 0.0025 | 2002 | 1296 | 532 | 1170 | 65.96 |
| 67 | 4600 - LR = 0.0036 | 280 | 2375 | 2254 | 91 | 53.1 | 180 | 8000 - LR = 0.0027 | 1820 | 1588 | 714 | 878 | 68.16 |
| 68 | 4600 - LR = 0.0038 | 123 | 2430 | 2411 | 36 | 51.06 | 181 | 8000 - LR = 0.0029 | 1588 | 1858 | 946 | 608 | 68.92 |
| 69 | 4600 - LR = 0.004 | 44 | 2451 | 2490 | 15 | 49.9 | 182 | 8000 - LR = 0.0031 | 1325 | 2056 | 1209 | 410 | 67.62 |
| 70 | 4600 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 | 183 | 8000 - LR = 0.0033 | 1054 | 2193 | 1480 | 273 | 64.94 |
| 71 | 4700 - LR = 0.001 | 0 | 2466 | 2534 | 0 | 49.32 | 184 | 8000 - LR = 0.0035 | 811 | 2284 | 1723 | 182 | 61.9 |
| 72 | 4700 - LR = 0.0025 | 0 | 2466 | 2534 | 0 | 49.32 | 185 | 8000 - LR = 0.0037 | 535 | 2352 | 1999 | 114 | 57.74 |
| 73 | 4700 - LR = 0.004 | 0 | 2466 | 2534 | 0 | 49.32 | 186 | 8000 - LR = 0.0039 | 287 | 2401 | 2247 | 65 | 53.76 |
| 74 | 4700 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 | 187 | 8000 - LR = 0.004 | 180 | 2424 | 2354 | 42 | 52.08 |
| 75 | 4900 - LR = 0.001 | 14 | 2450 | 2520 | 16 | 49.28 | 188 | 8000 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 |
| 76 | 4900 - LR = 0.002 | 1556 | 1626 | 978 | 840 | 63.64 | 189 | 8100 - LR = 0.001 | 1521 | 1636 | 1013 | 830 | 63.14 |
| 77 | 4900 - LR = 0.0022 | 1571 | 1646 | 963 | 820 | 64.34 | 191 | 8100 - LR = 0.0012 | 1649 | 1579 | 885 | 887 | 64.56 |
| 78 | 4900 - LR = 0.0024 | 1510 | 1702 | 1024 | 764 | 64.24 | 192 | 8100 - LR = 0.0014 | 1733 | 1539 | 801 | 927 | 65.44 |
| 79 | 4900 - LR = 0.0025 | 1449 | 1761 | 1085 | 705 | 64.2 | 193 | 8100 - LR = 0.0016 | 1772 | 1543 | 762 | 923 | 66.3 |
| 80 | 4900 - LR = 0.0026 | 1371 | 1834 | 1163 | 632 | 64.1 | 194 | 8100 - LR = 0.0018 | 1757 | 1601 | 777 | 865 | 67.16 |
| 81 | 4900 - LR = 0.0028 | 1180 | 1977 | 1354 | 489 | 63.14 | 195 | 8100 - LR = 0.002 | 1706 | 1674 | 828 | 792 | 67.6 |
| 82 | 4900 - LR = 0.004 | 64 | 2446 | 2470 | 20 | 50.2 | 196 | 8100 - LR = 0.0022 | 1606 | 1799 | 928 | 667 | 68.1 |
| 83 | 4900 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 | 197 | 8100 - LR = 0.0024 | 1471 | 1931 | 1063 | 535 | 68.04 |
| 84 | 5000 - LR = 0.001 | 105 | 2383 | 2429 | 83 | 49.76 | 198 | 8100 - LR = 0.0025 | 1393 | 1977 | 1141 | 489 | 67.4 |
| 85 | 5000 - LR = 0.0025 | 902 | 2134 | 1632 | 332 | 60.72 | 199 | 8100 - LR = 0.0026 | 1303 | 2045 | 1231 | 421 | 66.96 |
| 86 | 5000 - LR = 0.004 | 85 | 2435 | 2449 | 31 | 50.4 | 200 | 8100 - LR = 0.0028 | 1138 | 2132 | 1396 | 334 | 65.4 |
| 87 | 5000 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 | 201 | 8100 - LR = 0.003 | 943 | 2222 | 1591 | 244 | 63.3 |
| 88 | 5100 - LR = 0.001 | 1119 | 1793 | 1415 | 673 | 58.24 | 202 | 8100 - LR = 0.0032 | 715 | 2279 | 1819 | 187 | 59.88 |
| 89 | 5100 - LR = 0.0025 | 2301 | 707 | 233 | 1759 | 60.16 | 203 | 8100 - LR = 0.0034 | 482 | 2340 | 2052 | 126 | 56.44 |
| 91 | 5100 - LR = 0.0027 | 2225 | 913 | 309 | 1553 | 62.76 | 204 | 8100 - LR = 0.0036 | 276 | 2388 | 2258 | 78 | 53.28 |

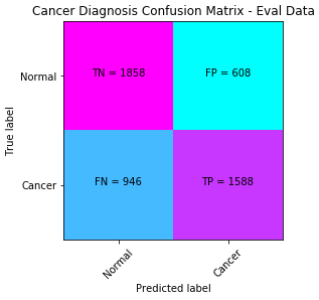
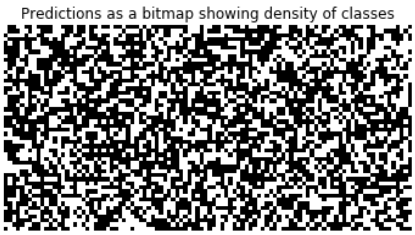
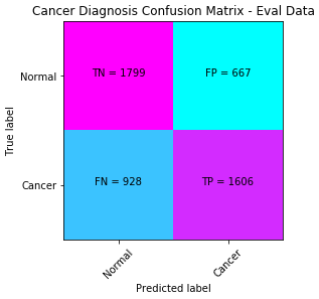
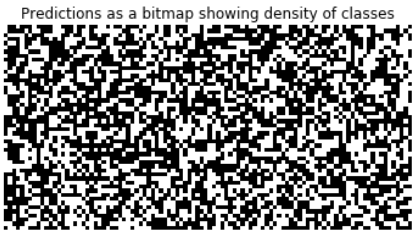
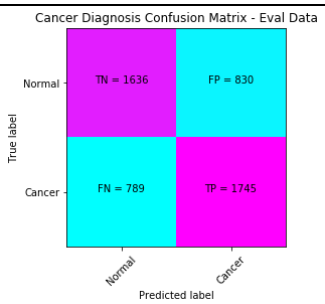
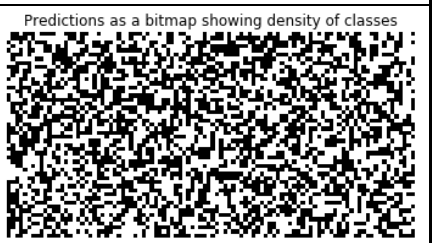
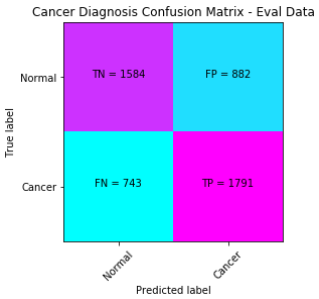
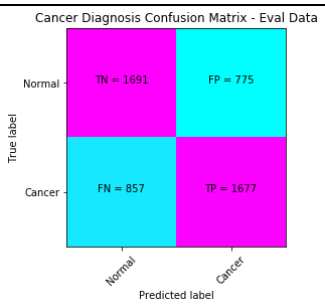
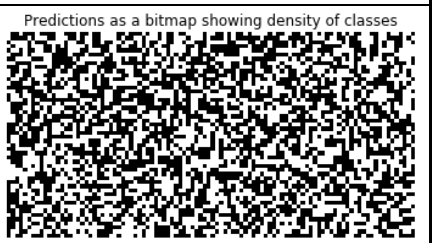
| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 92 | 5100 - LR = 0.0029 | 2096 | 1166 | 438 | 1300 | 65.24 | 205 | 8100 - LR = 0.0038 | 114 | 2431 | 2420 | 35 | 50.9 |
| 93 | 5100 - LR = 0.0031 | 1938 | 1410 | 596 | 1056 | 66.96 | 206 | 8100 - LR = 0.004 | 31 | 2457 | 2503 | 9 | 49.76 |
| 94 | 5100 - LR = 0.0033 | 1745 | 1636 | 789 | 830 | 67.62 | 207 | 8100 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 |
| 95 | 5100 - LR = 0.0035 | 1527 | 1841 | 1007 | 625 | 67.36 | 208 | 8100 - LR = 0.007 | 0 | 2466 | 2534 | 0 | 49.32 |
| 96 | 5100 - LR = 0.0037 | 1297 | 2000 | 1237 | 466 | 65.94 | 209 | 8300 - LR = 0.001 | 1032 | 1987 | 1502 | 479 | 60.38 |
| 97 | 5100 - LR = 0.0039 | 1085 | 2121 | 1449 | 345 | 64.12 | 210 | 8300 - LR = 0.0025 | 55 | 2448 | 2479 | 18 | 50.06 |
| 98 | 5100 - LR = 0.004 | 938 | 2180 | 1596 | 286 | 62.36 | 211 | 8300 - LR = 0.004 | 2 | 2466 | 2532 | 0 | 49.36 |
| 99 | 5100 - LR = 0.0041 | 820 | 2226 | 1714 | 240 | 60.92 | 212 | 8300 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 |
| 100 | 5100 - LR = 0.0043 | 597 | 2289 | 1937 | 177 | 57.72 | 213 | 9100 - LR = 0.001 | 0 | 2466 | 2534 | 0 | 49.32 |
| 101 | 5100 - LR = 0.0045 | 392 | 2353 | 2142 | 113 | 54.9 | 214 | 9100 - LR = 0.004 | 0 | 2466 | 2534 | 0 | 49.32 |
| 102 | 5100 - LR = 0.0047 | 213 | 2400 | 2321 | 66 | 52.26 | 215 | 9100 - LR = 0.007 | 0 | 2466 | 2534 | 0 | 49.32 |
| 103 | 5100 - LR = 0.0049 | 89 | 2440 | 2445 | 26 | 50.58 | | | | | | | |
| 104 | 5100 - LR = 0.0051 | 31 | 2457 | 2503 | 9 | 49.76 | | | | | | | |
| 105 | 5100 - LR = 0.0053 | 14 | 2463 | 2520 | 3 | 49.54 | | | | | | | |
| 106 | 5100 - LR = 0.0055 | 8 | 2465 | 2526 | 1 | 49.46 | | | | | | | |
| 107 | 5100 - LR = 0.007 | 0 | 2466 | 2534 | 0 | 49.32 | | | | | | | |
| 108 | 5300 - LR = 0.001 | 0 | 2466 | 2534 | 0 | 49.32 | | | | | | | |
| 109 | 5300 - LR = 0.0025 | 0 | 2466 | 2534 | 0 | 49.32 | | | | | | | |
| 110 | 5300 - LR = 0.004 | 0 | 2466 | 2534 | 0 | 49.32 | | | | | | | |
| 111 | 5300 - LR = 0.0055 | 0 | 2466 | 2534 | 0 | 49.32 | | | | | | | |
| 112 | 5600 - LR = 0.001 | 132 | 2371 | 2402 | 95 | 50.06 | | | | | | | |

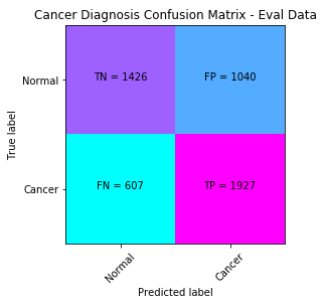
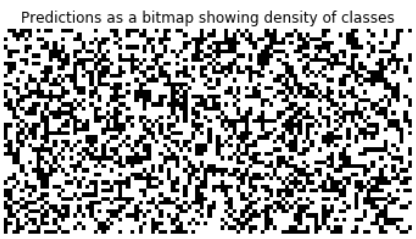
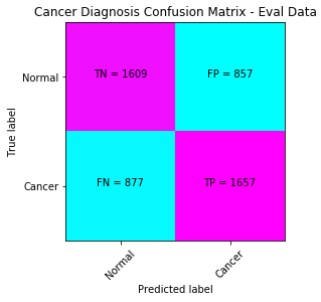
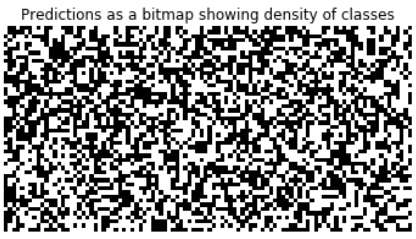
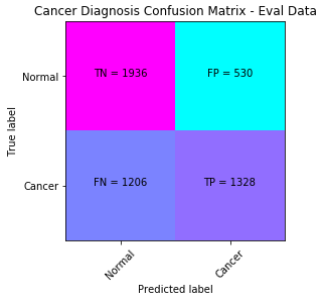
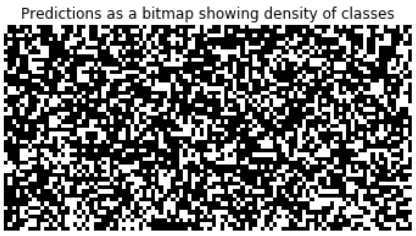
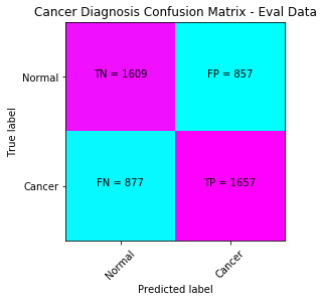
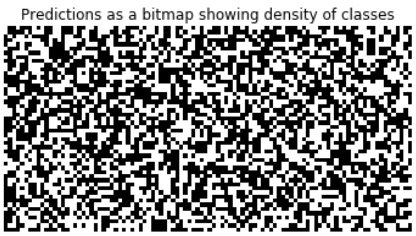
# 8.3 Appendix III – Training/testing Error rates for each Test case

| Training Sample Size and Learning Rate | Training | Testing (25000 Samples) |
|---|---|---|
| 8000 - LR = 0.0029 |  |  |
| 8100 - LR = 0.0022 |  |  |
| 5100 - LR = 0.0033 |  |  |
| 3700 - LR = 0.0039 |  |  |
| 6000 - LR = 0.0027 |  |  |
| 4000 - LR = 0.0053 |  |  |

| | | |
|---|---|---|
| 4600 - LR = 0.0024 |  |  |
| 6100 - LR = 0.0023 |  |  |
| 4900 - LR = 0.0022 |  |  |

# 8.4 Appendix IV – Confusion Matrices and Class Bitmaps for each test case

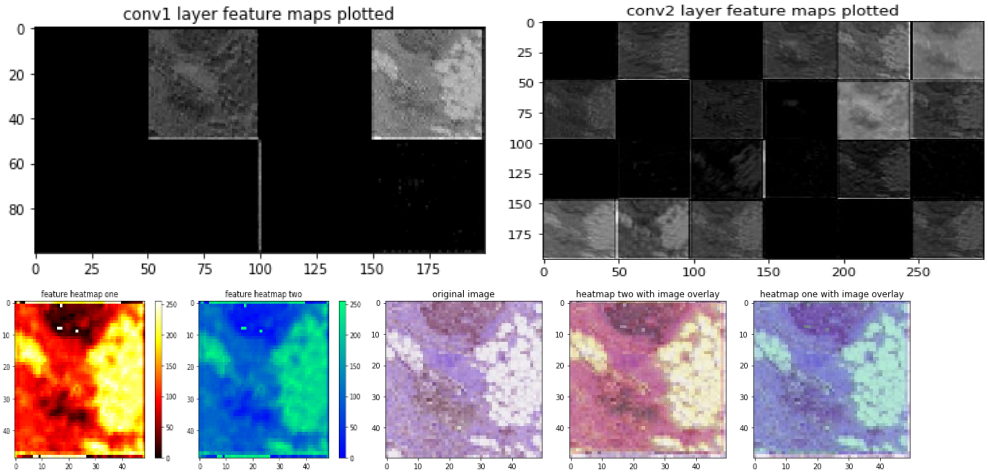
| Sample Size - Learning Rate | Percentage Accuracy | Confusion Matrix | Class Bitmap (1 = white, Black = 0) |
|---|---|---|---|
| 8000 - LR = 0.0029 | 68.92 |  |  |
| 8100 - LR = 0.0022 | 68.1 |  |  |
| 5100 - LR = 0.0033 | 67.62 |  |  |
| 3700 - LR = 0.0039 | 67.5 |  |  |
| 6000 - LR = 0.0027 | 67.36 |  |  |

| | | | |
|---|---|---|---|
| 4000 - LR = 0.0053 | 67.06 |  |  |
| 4600 - LR = 0.0024 | 65.32 |  |  |
| 6100 - LR = 0.0023 | 65.28 |  |  |
| 4900 - LR = 0.0022 | 64.34 |  |  |

# 8.5 Appendix V – Predictions and Attention maps for each test case

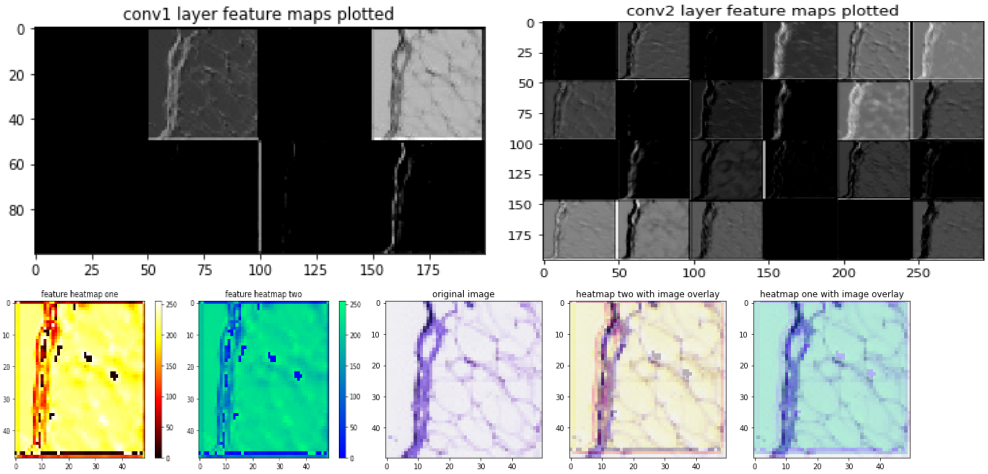
## 8.5.1 Image 1-5 – Training Sample Size 8000 - LR = 0.0029

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 2.498235162983571e-13 %



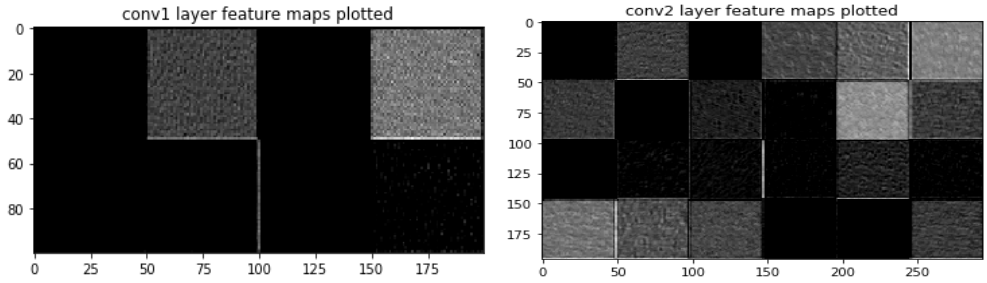

---

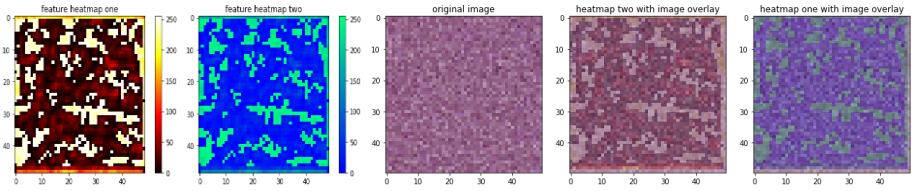
Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 4.724499396845279e-12 %




---

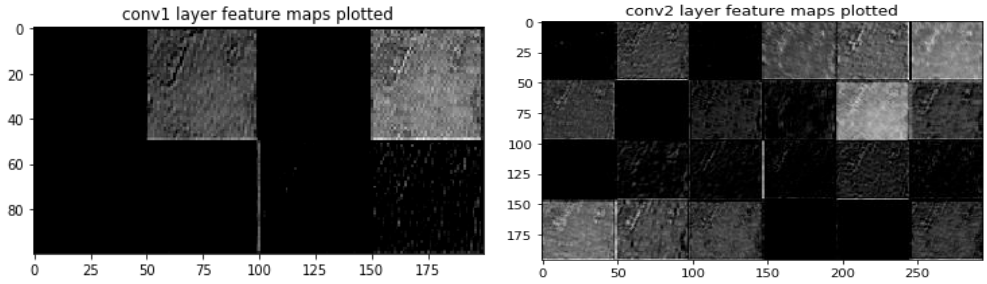
Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 6.4113665765308525e-09 %

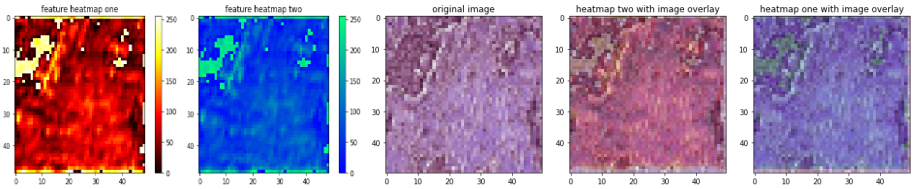



---

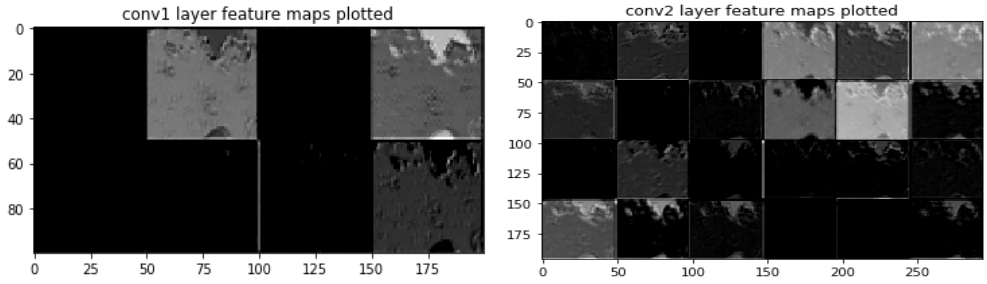
Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 2.7667191454527895e-09 %

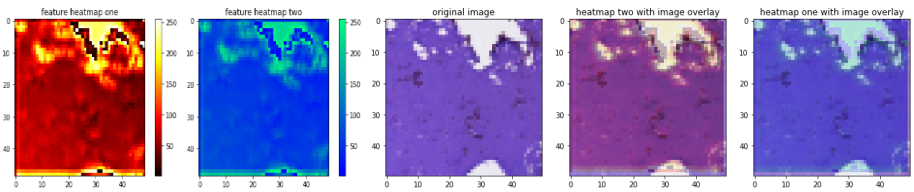



---

Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %

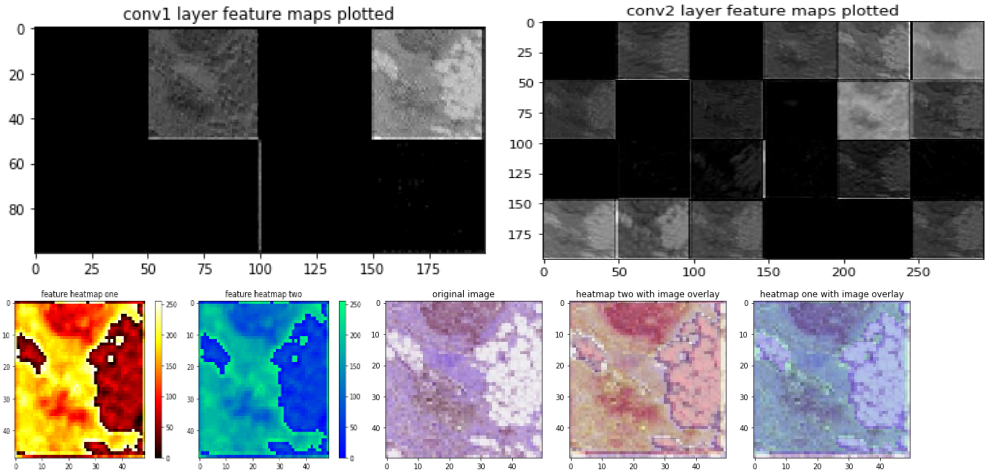


---

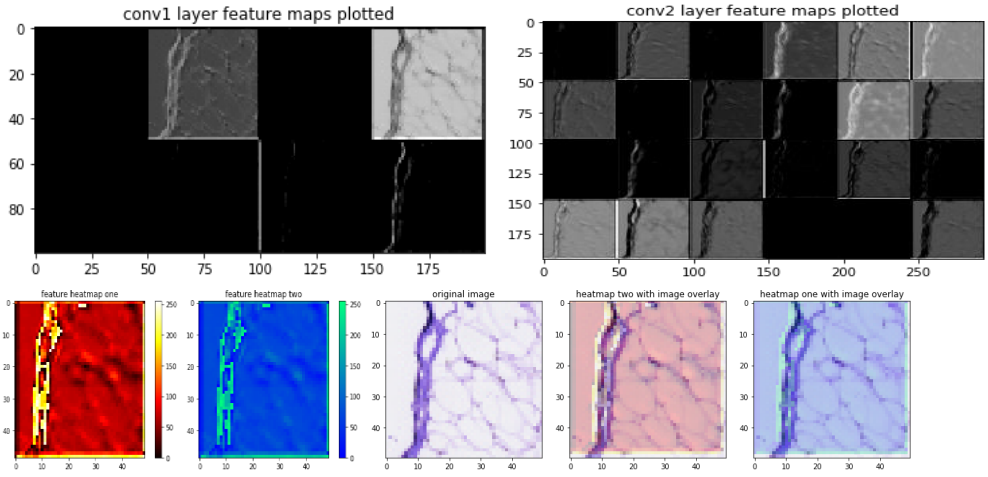
## 8.5.2 Image 1-5 – Training Sample Size 8100 - LR = 0.0022

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 1.8883785646295337e-07 %



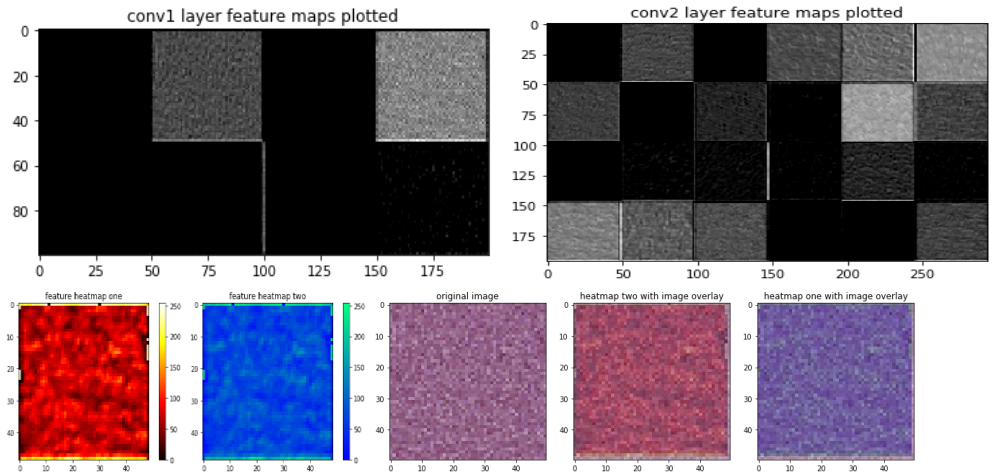

---

Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 1.201589609062248e-09 %



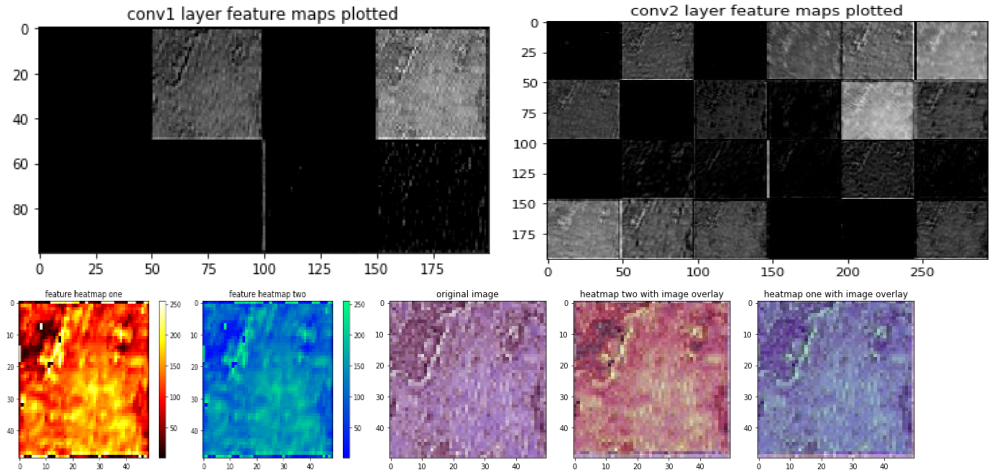

---

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 4.886999818154436e-05 %



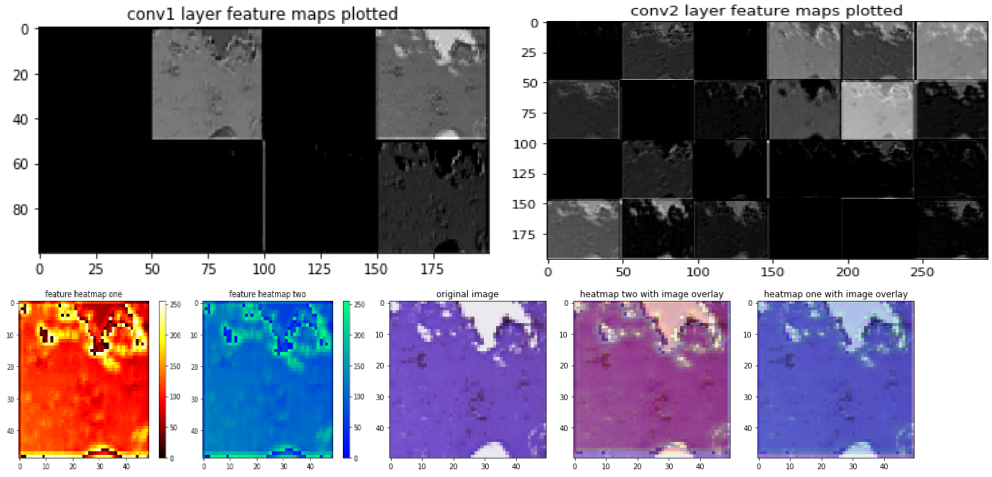

---

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 1.1387933651629822e-08 %




---

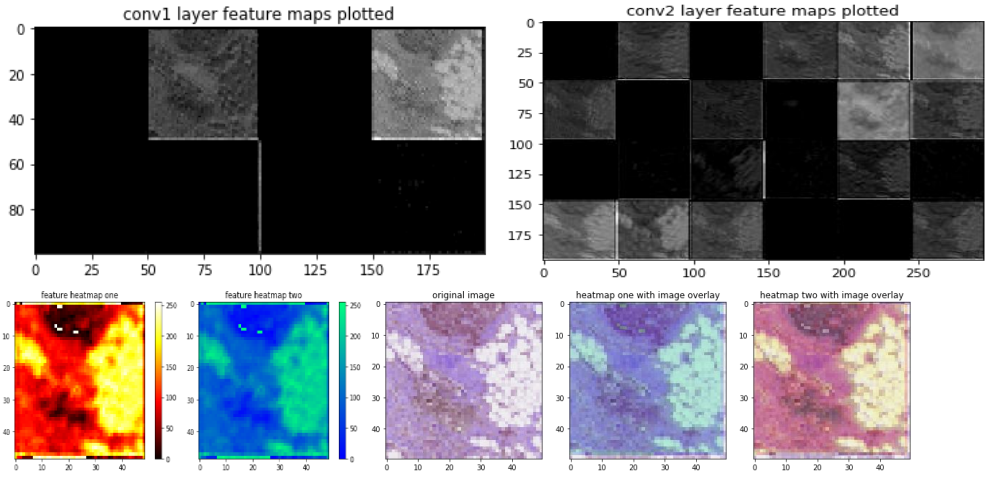
Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %




---

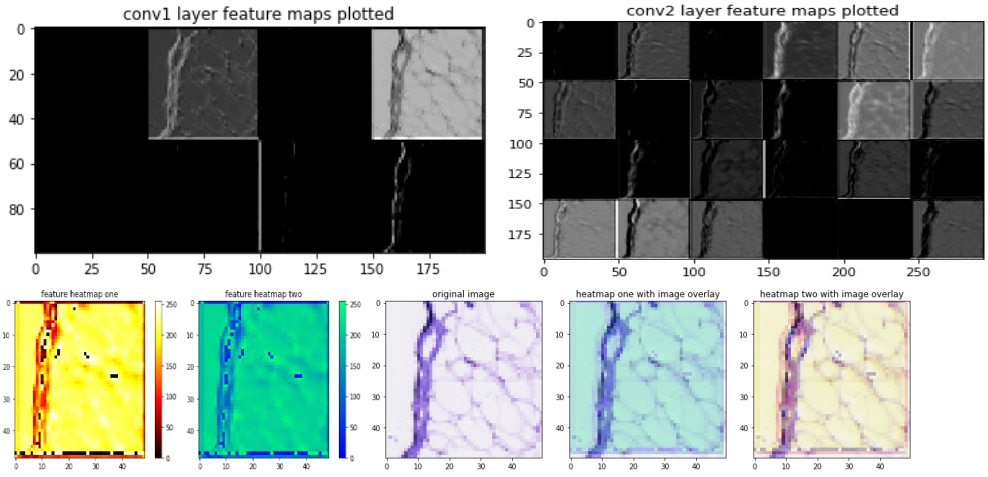
## 8.5.3 Image 1-5 – Training Sample Size 5100 - LR = 0.0033

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 7.46453399091962e-08 %



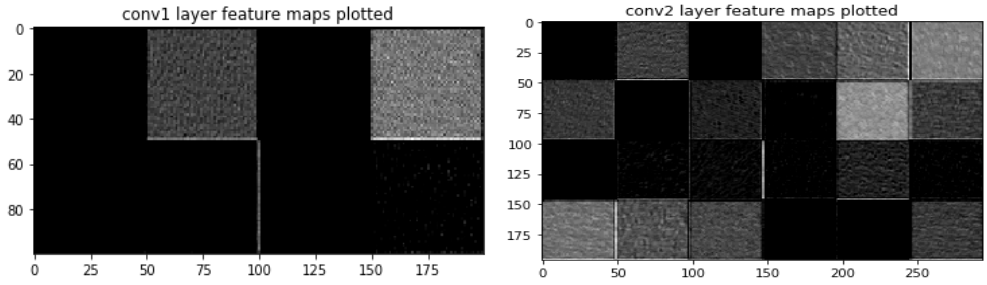

---

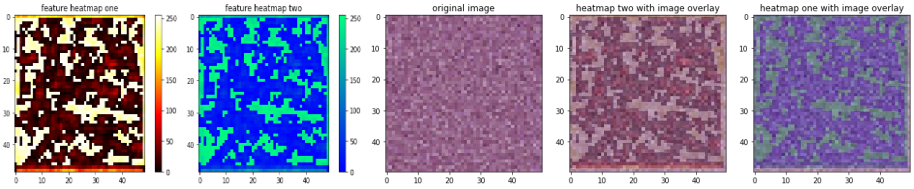
Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 2.864127392285948e-12 %




---

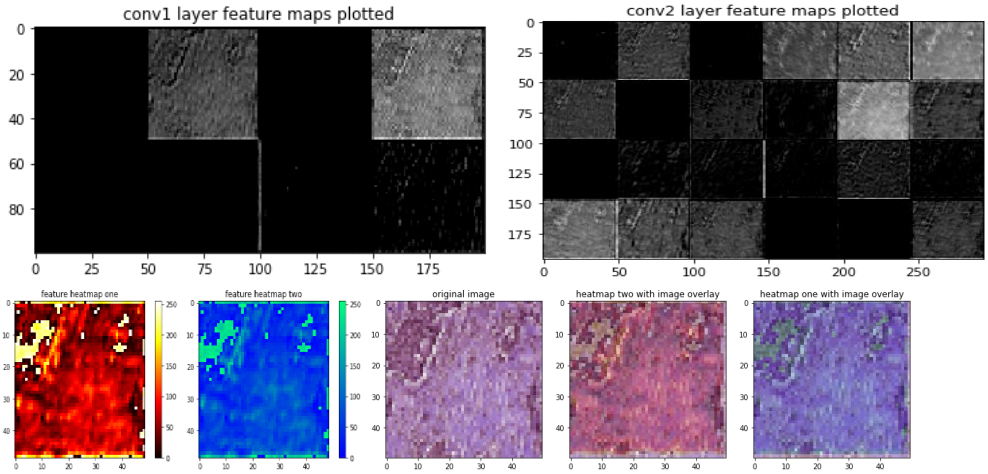
Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 7.021151304797968e-05 %





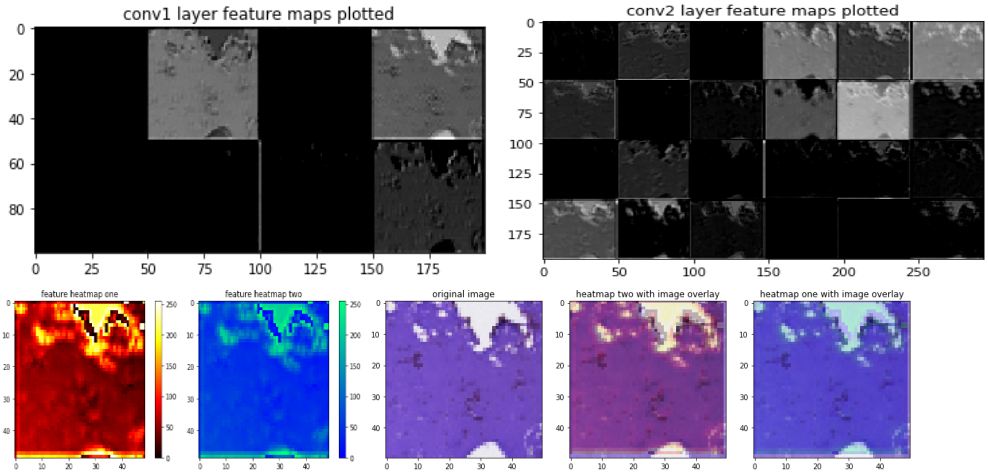

---

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 3.22826175513935e-09 %




---

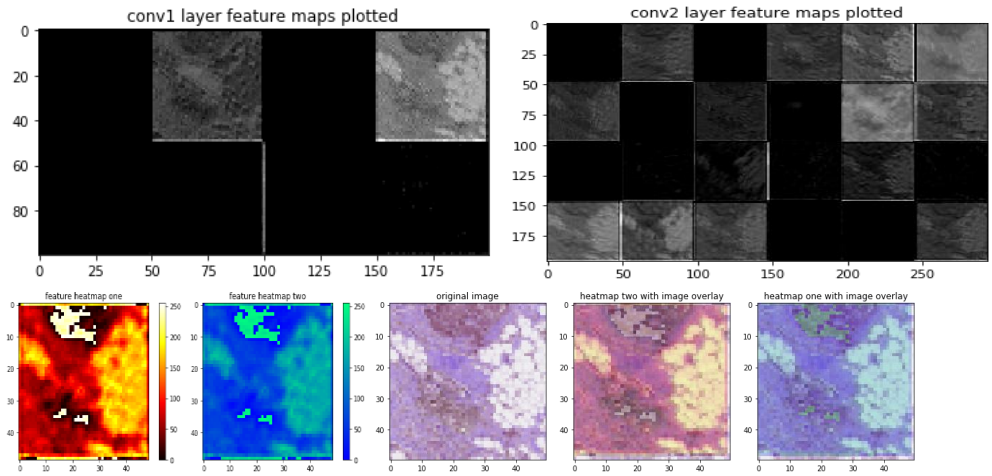
Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 1.4670636433331237e-35 %




---

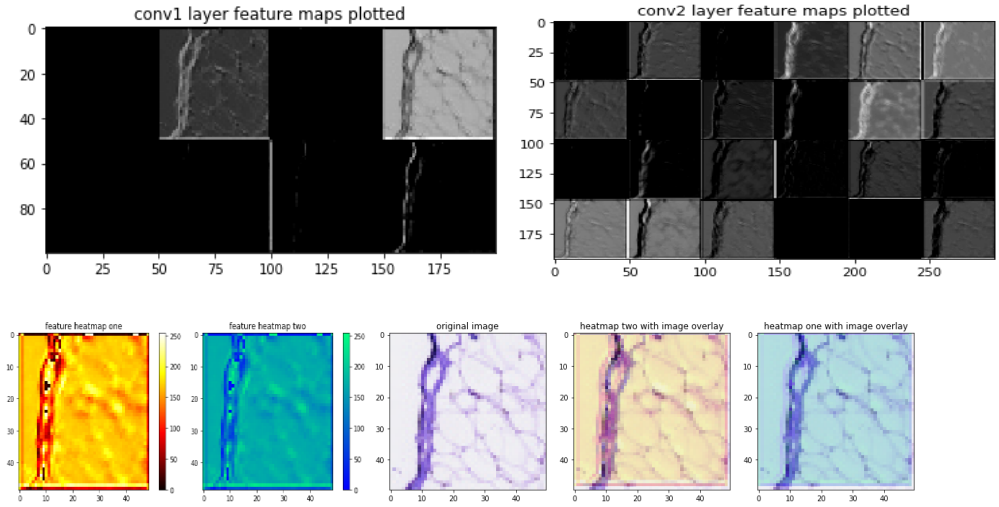
## 8.5.4 Image 1-5 – Training Sample Size 3700 - LR = 0.0039

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 6.502082205983584e-08 %




---

Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 7.989129358065838e-08 %




---

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0008109926966426428 %

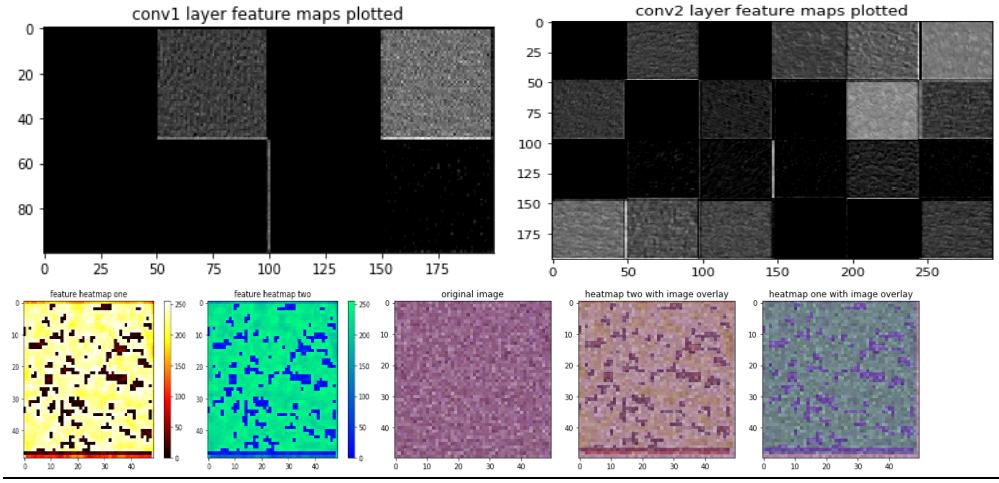
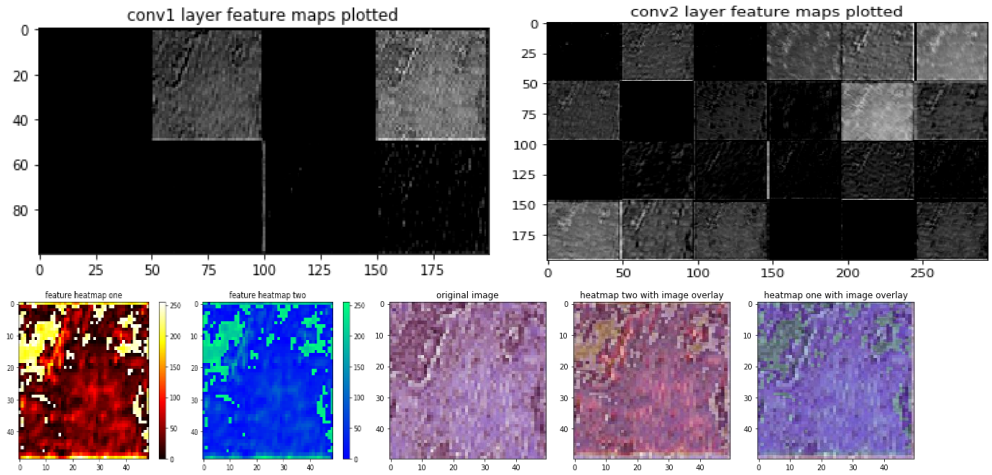
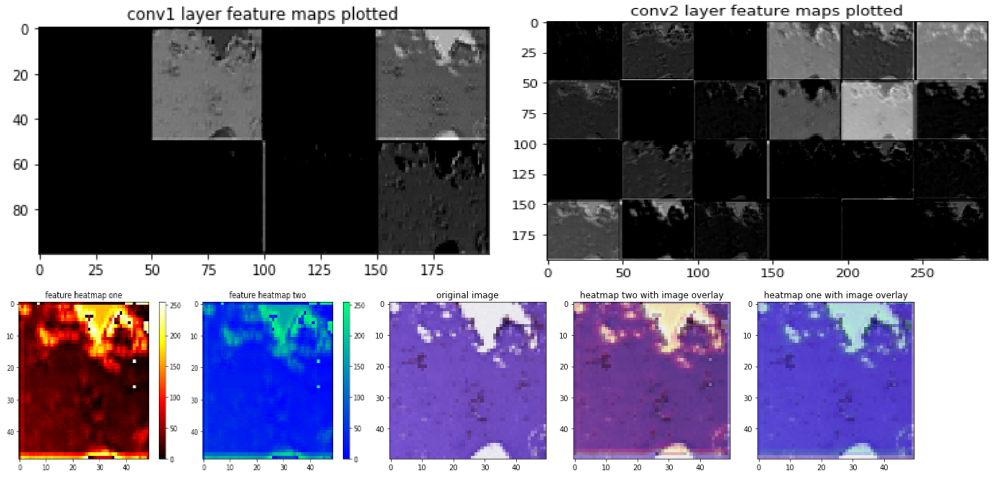


Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 1.0367314784431869e-07 %




---

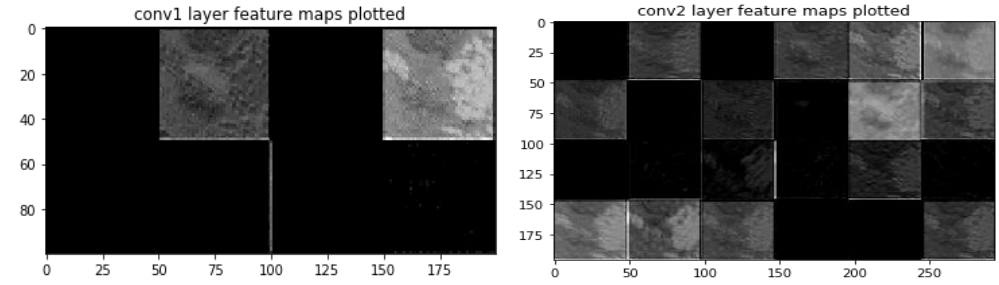
Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 3.944143427359344e-33 %

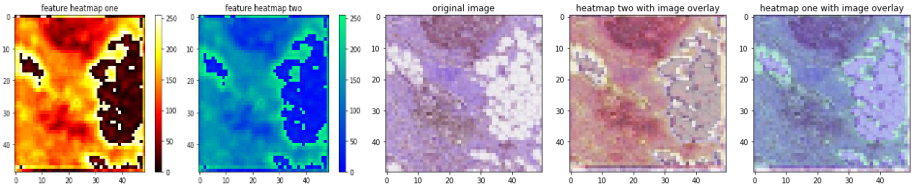



---

## 8.5.5 Image 1-5 – Training Sample Size 6000 - LR = 0.0027

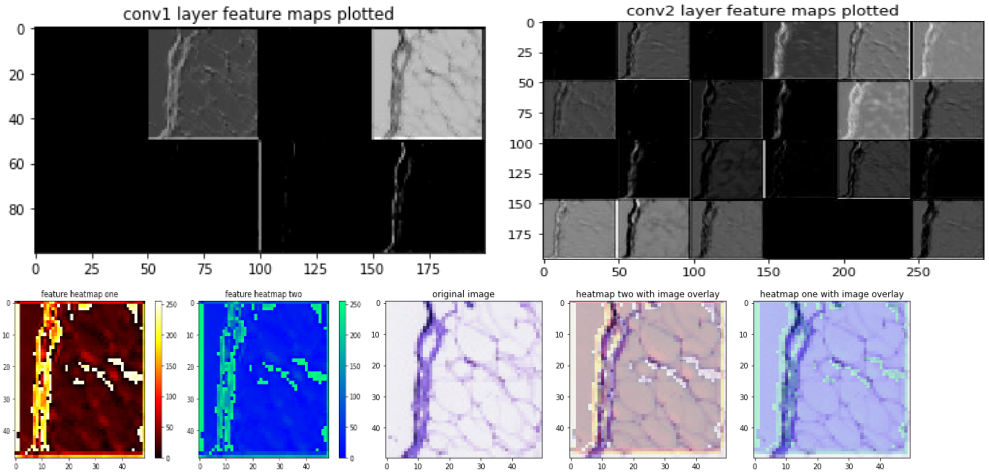
Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 3.933970038616508e-07 %





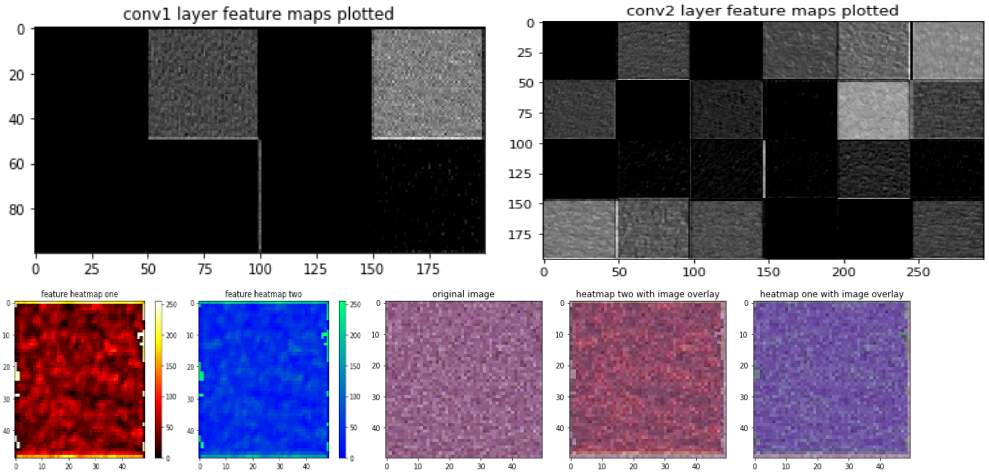

---

Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 1.8015398201010058e-10 %



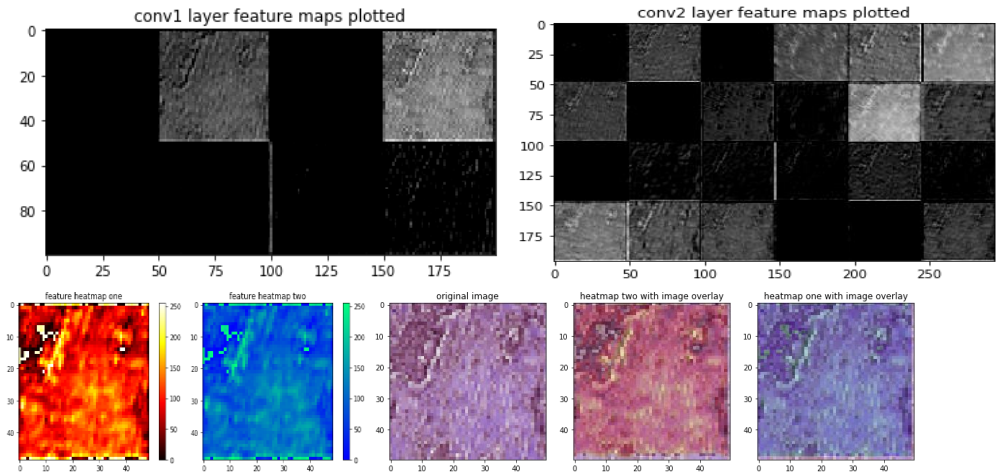

---

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.00035862190088664647 %



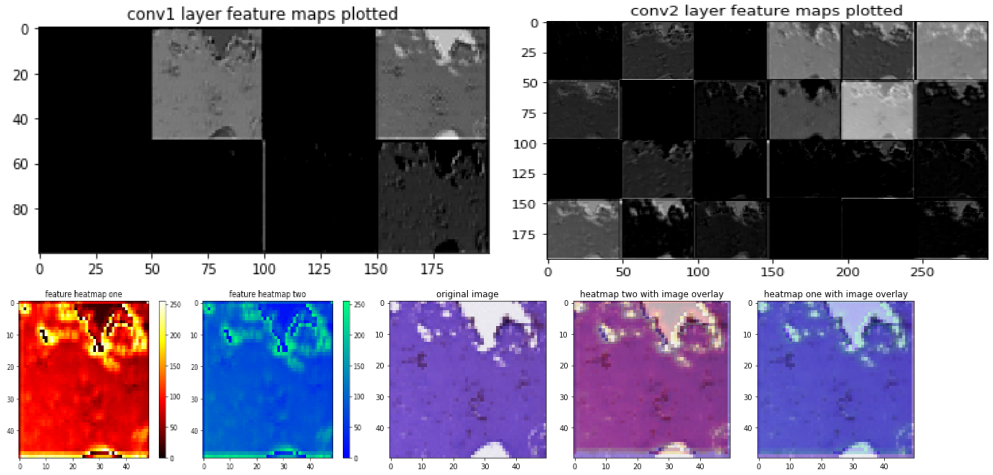

---

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 7.742173568914268e-09 %




---

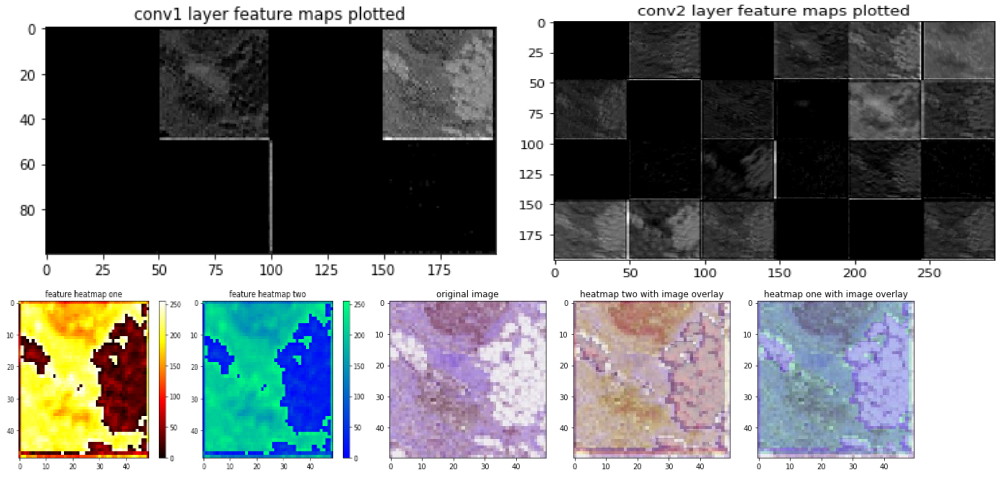
Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 9.120848516012907e-37 %




---

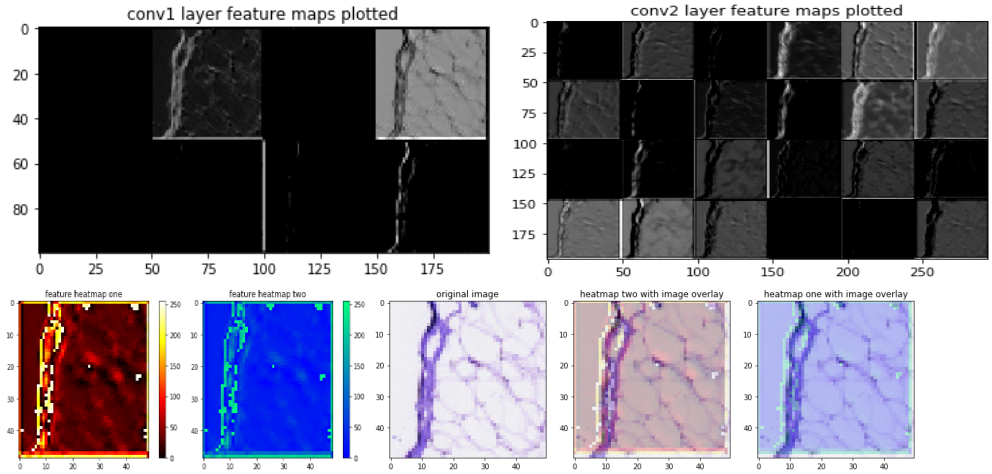
## 8.5.6 Image 1-5 – Training Sample Size 4000 - LR = 0.0053

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 3.5563427203122444e-08 %



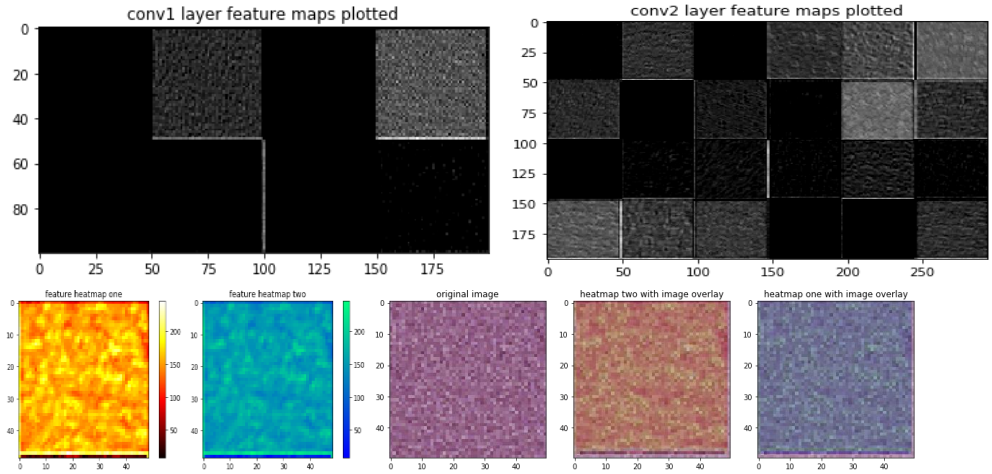

---

Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 8.021896962908523e-23 %



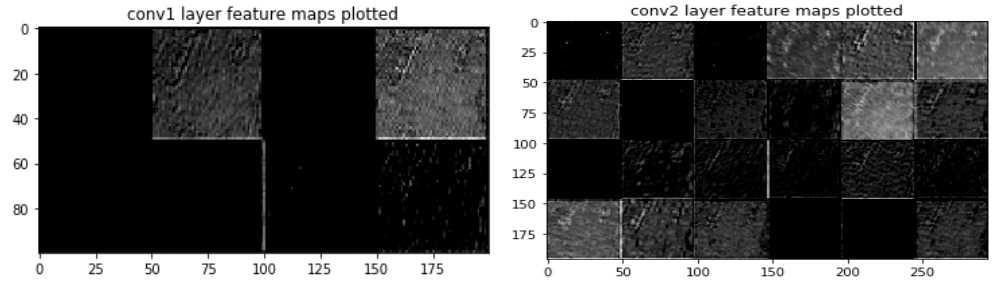

---

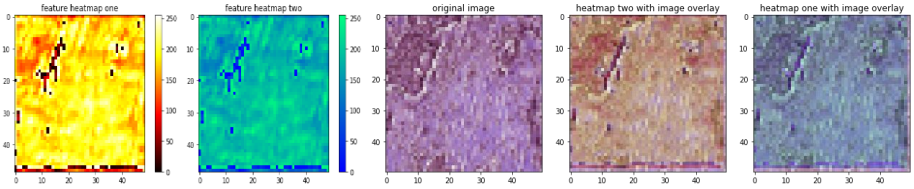
Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 1.2641825364312353e-07 %




---

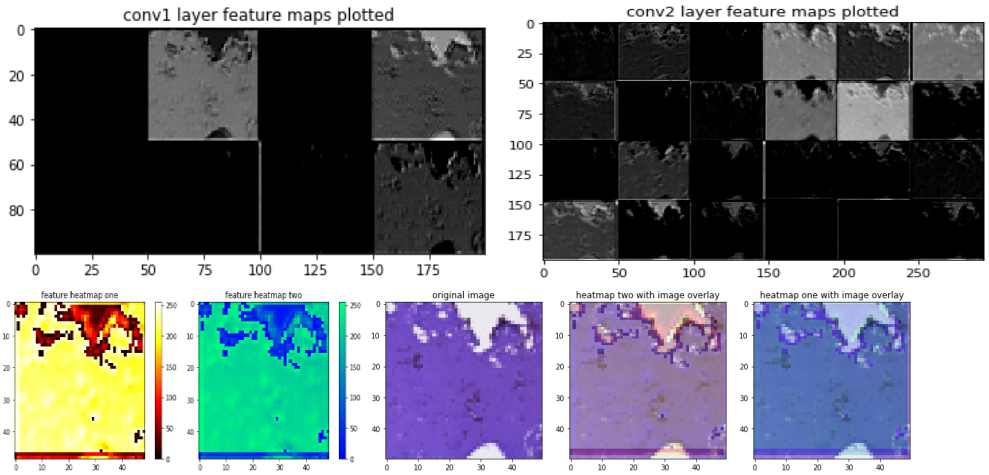
Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 5.019524360505741e-12 %






---

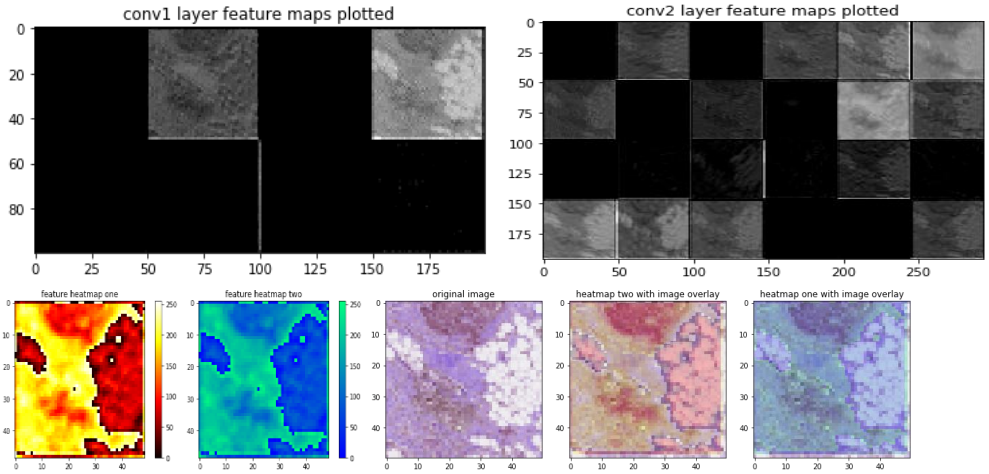
Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 9.720845562339486e-36 %




---

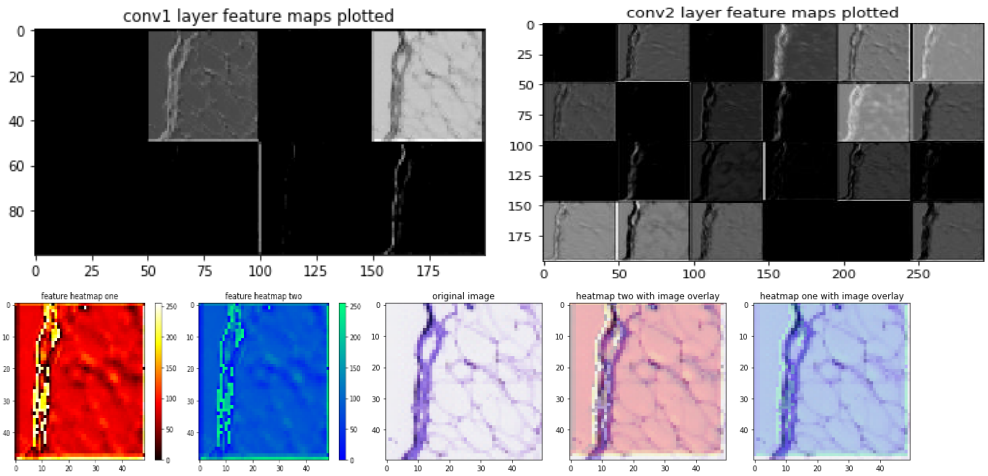
## 8.5.7 Image 1-5 – Training Sample Size 4600 - LR = 0.0024

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.00013363978723646142 %



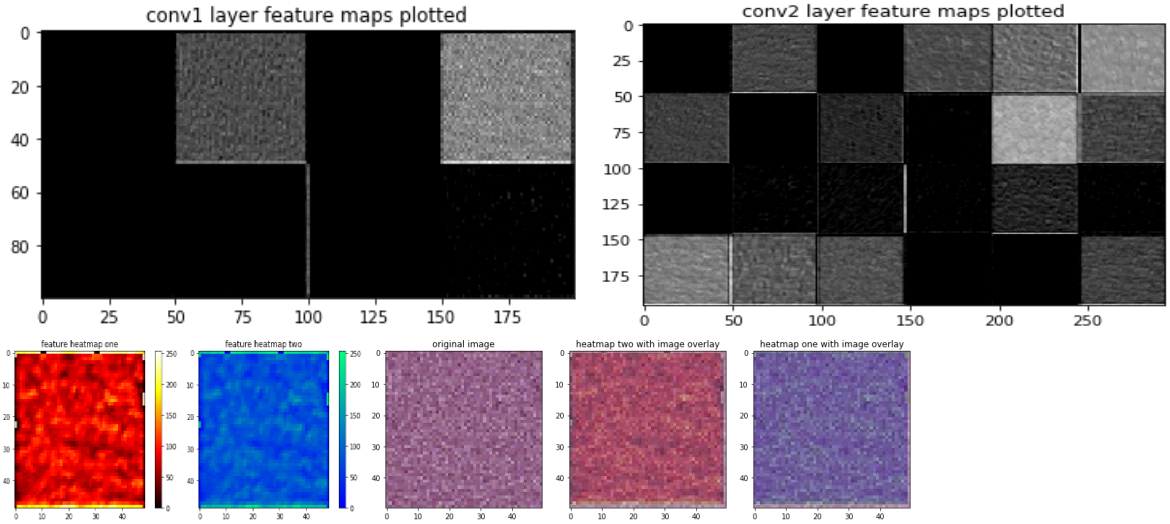

---

Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 4.208646942061023e-06 %



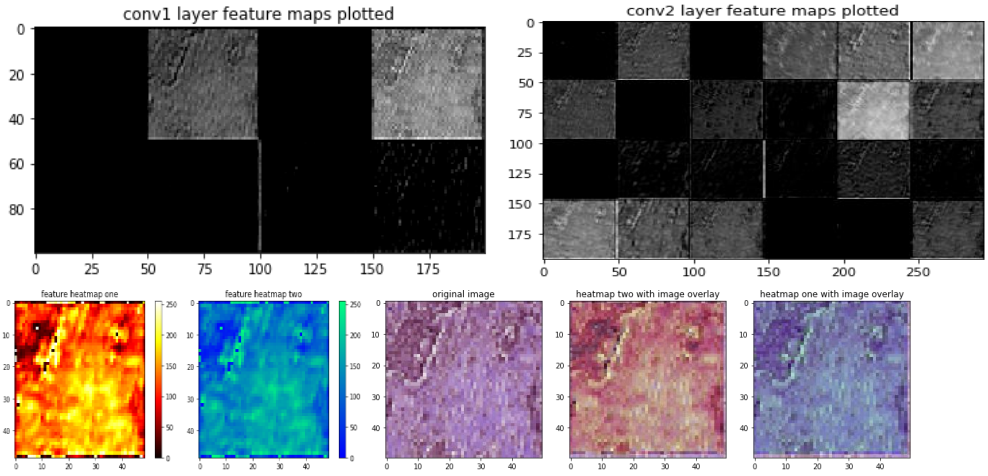

---

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 99.88 %
Confidence of Alternative: 0.11656025890260935 %

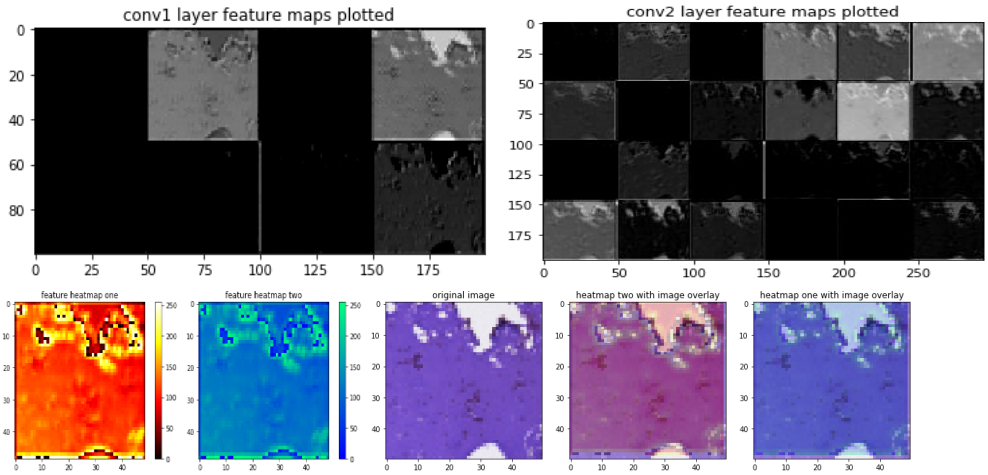



---

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 3.9941459029080306e-07 %



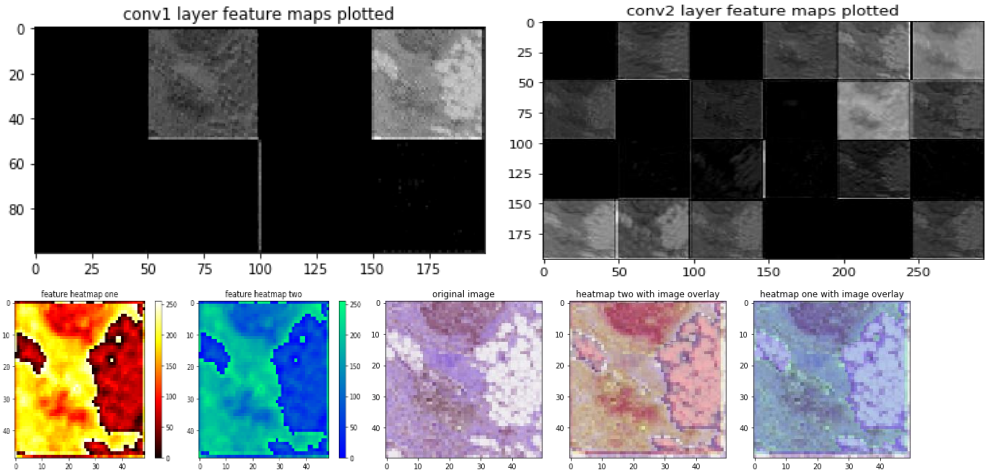
---
Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 5.717459846542184e-24 %



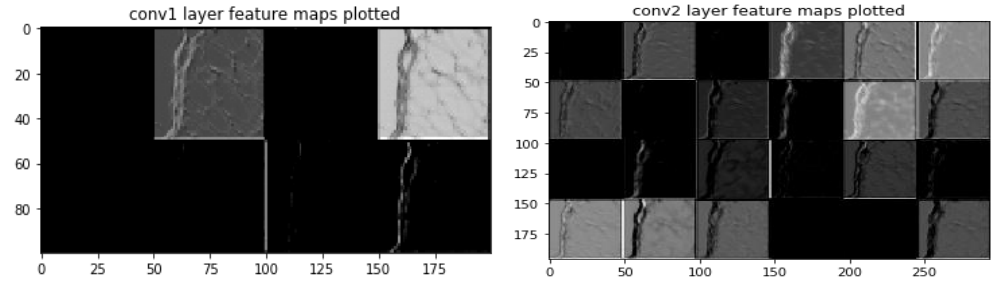

---

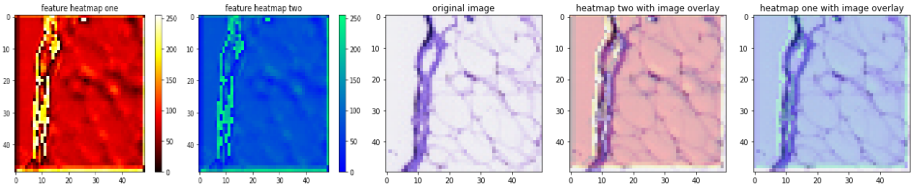
## 8.5.8 Image 1-5 – Training Sample Size 6100 - LR = 0.0023

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0002454016112096724 %



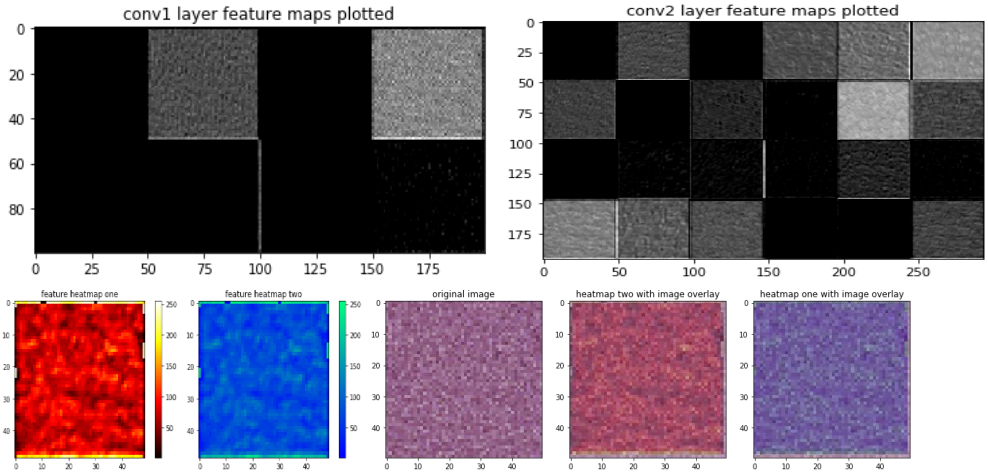

---
Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 5.707803587412519e-10 %





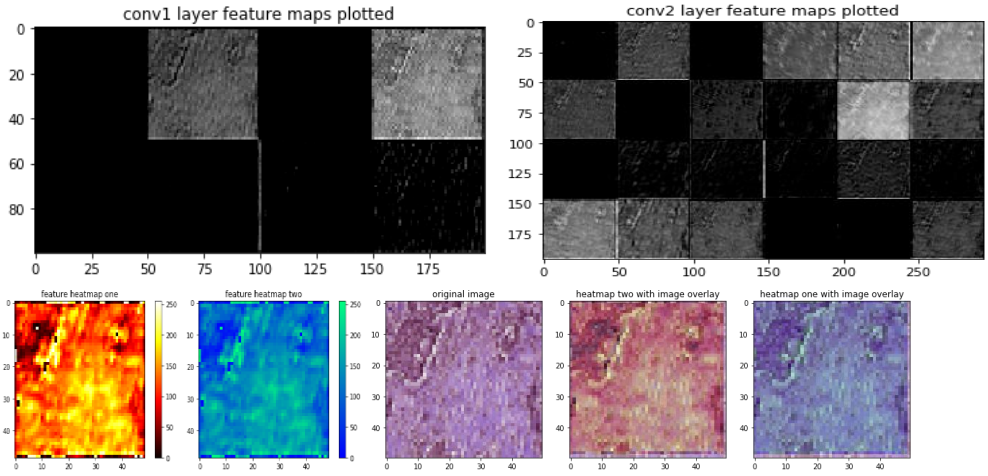

---

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 99.76 %
Confidence of Alternative: 0.2391295274719596 %



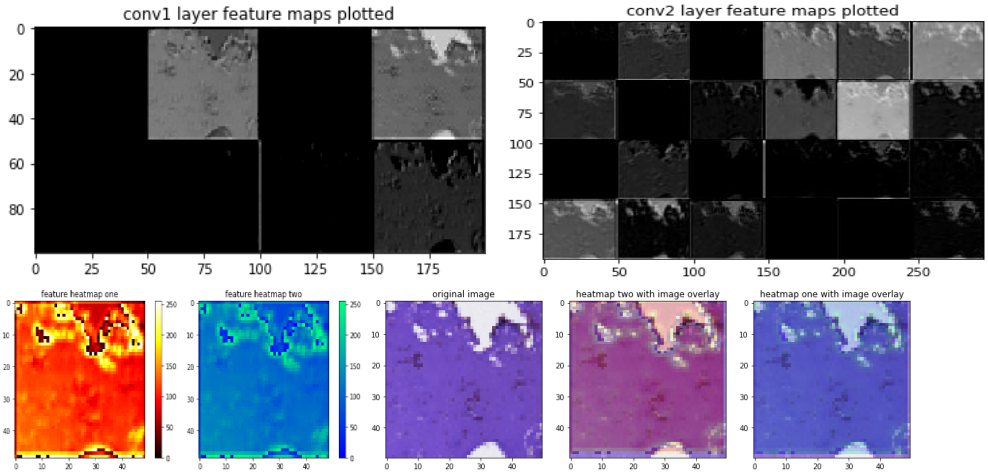

---

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 1.612493178981822e-05 %




---

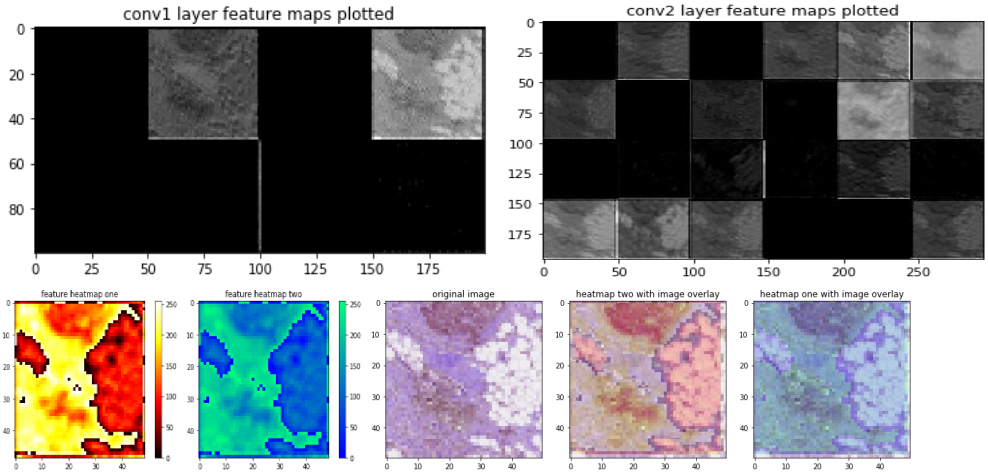
Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 2.083955024205295e-37 %




---

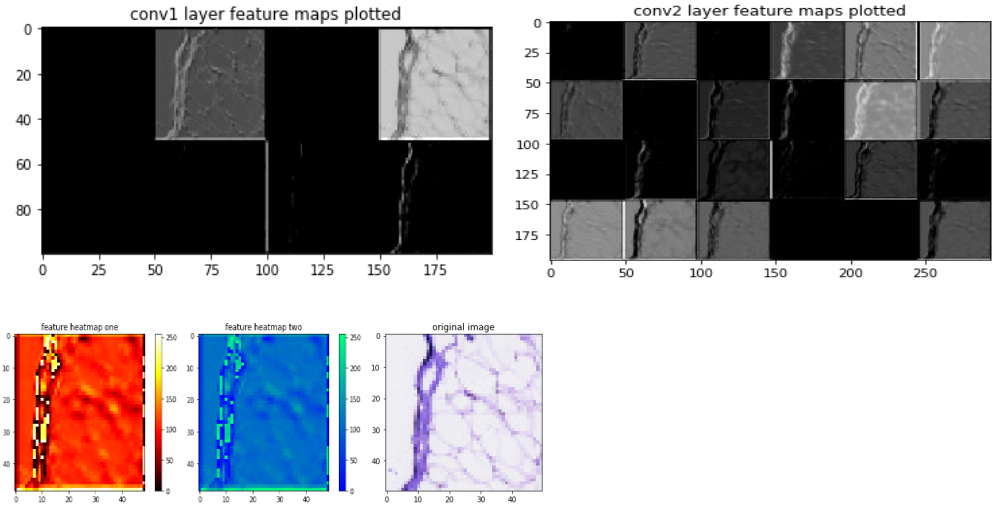
## 8.5.9 Image 1-5 – Training Sample Size 4900 - LR = 0.0022

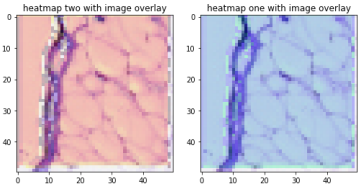
Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.001068221990863094 %




---

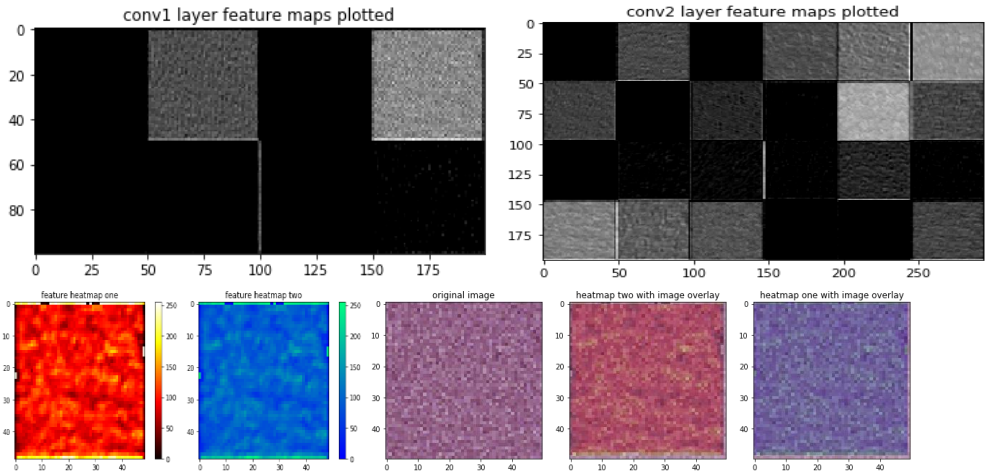
Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 6.587192569185163e-07 %





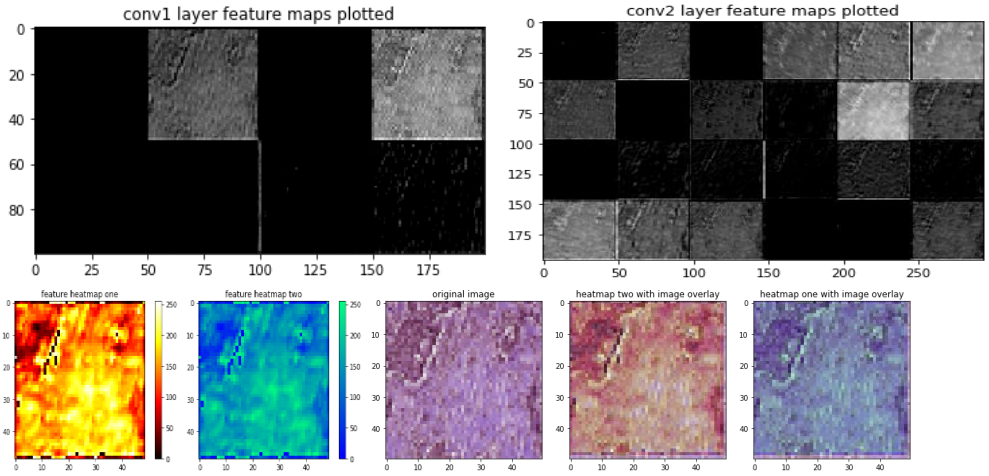

---

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 99.51 %
Confidence of Alternative: 0.4921366926282644 %



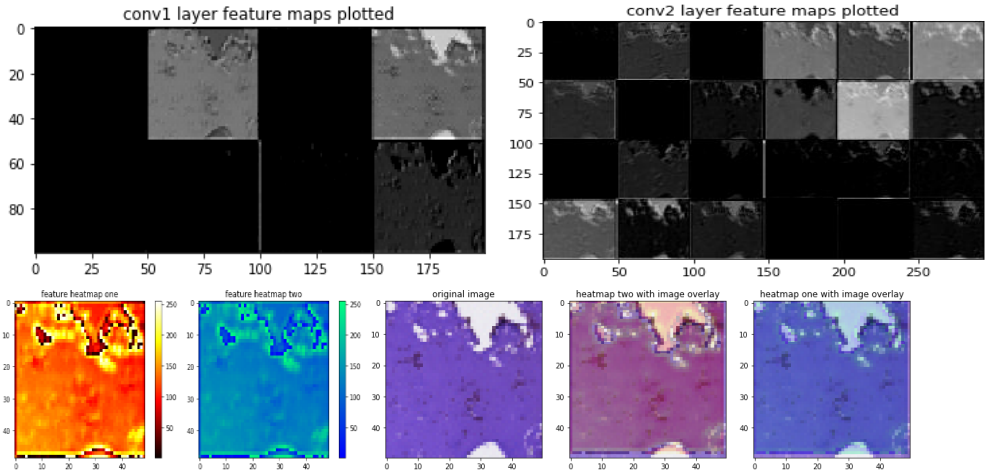

---

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 1.4868780873200649e-06 %




---

Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 9.323827084428492e-23 %




---

# 8.6 Appendix VI – Displaying the image and prediction outputs

### *8.6.1.1 6000 - LR = 0.0027*

---

---

Showing successes for cancer: 0 to 9 with confidence <= 100 %

---

---

Image Sample Number: 0

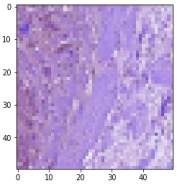


Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %

---

---

Image Sample Number: 3

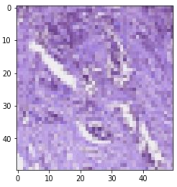


Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %

---

---

Image Sample Number: 7

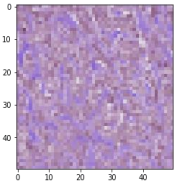


Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 12

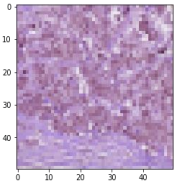


Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 13

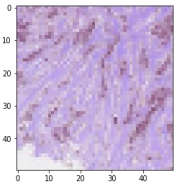


Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 17

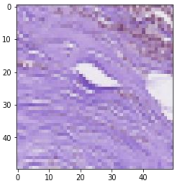


Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 20

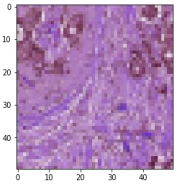


Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 23

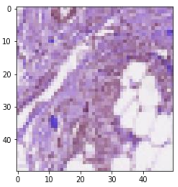


Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 24

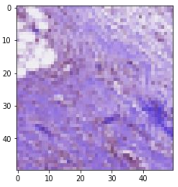


Image Label: 1
Image Prediction: 1
Confidence of Prediction: 97.5 %
Confidence of Alternative: 2.5 %
---
---
Image Sample Number: 25

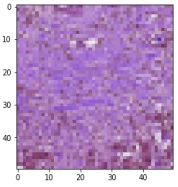


Image Label: 1
Image Prediction: 1
Confidence of Prediction: 99.28 %
Confidence of Alternative: 0.72 %

---
---
Showing successes for normal: 0 to 9 with confidence <= 100 %
---
---
Image Sample Number: 1

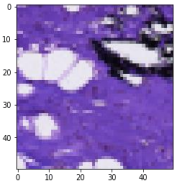


Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 6

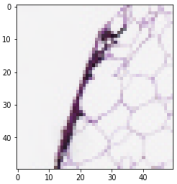


Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 8

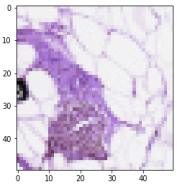


Image Label: 0
Image Prediction: 0
Confidence of Prediction: 99.99 %
Confidence of Alternative: 0.01 %
---
---
Image Sample Number: 9

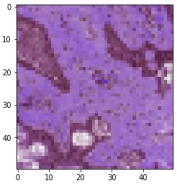


Image Label: 0
Image Prediction: 0
Confidence of Prediction: 98.5 %
Confidence of Alternative: 1.5 %
---
---
Image Sample Number: 11

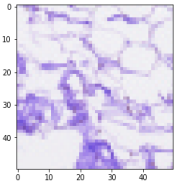


Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 15

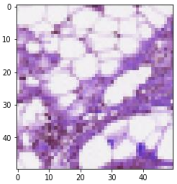


Image Label: 0
Image Prediction: 0
Confidence of Prediction: 99.96 %
Confidence of Alternative: 0.04 %
---
---
Image Sample Number: 19

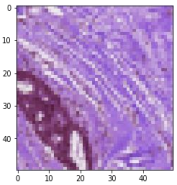


Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 22

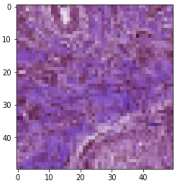


Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 27

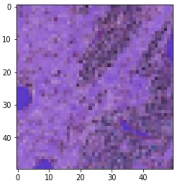


Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 30

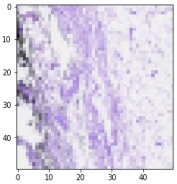


Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %

---
---
Showing failures for cancer: 0 to 9 with confidence <= 100 %
---
---
Image Sample Number: 2

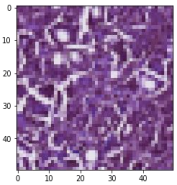


Image Label: 1
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 4

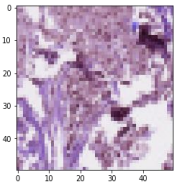


Image Label: 1
Image Prediction: 0
Confidence of Prediction: 96.14 %
Confidence of Alternative: 3.86 %
---
---
Image Sample Number: 5

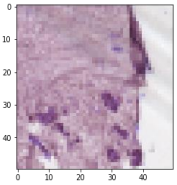


Image Label: 1
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 10

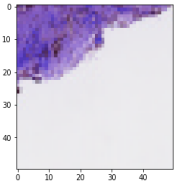


Image Label: 1
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 14

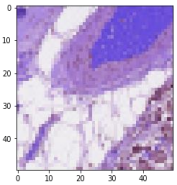


Image Label: 1
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 16

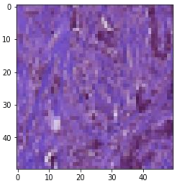


Image Label: 1
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 18

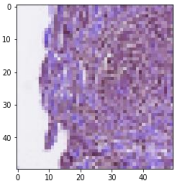


Image Label: 1
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 21

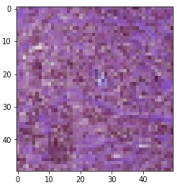


Image Label: 1
Image Prediction: 0
Confidence of Prediction: 87.62 %
Confidence of Alternative: 12.38 %
---
---
Image Sample Number: 28

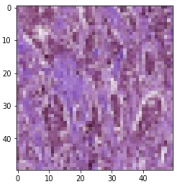


Image Label: 1
Image Prediction: 0
Confidence of Prediction: 74.81 %
Confidence of Alternative: 25.19 %
---
---
Image Sample Number: 39

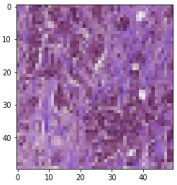


Image Label: 1
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Showing failures for normal: 0 to 9 with confidence <= 100 %
---
---
Image Sample Number: 26

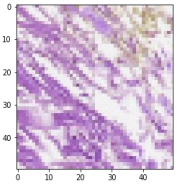


Image Label: 0
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 41

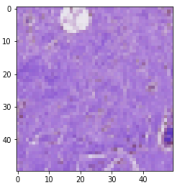


Image Label: 0
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 44

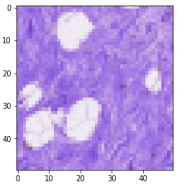


Image Label: 0
Image Prediction: 1
Confidence of Prediction: 99.97 %
Confidence of Alternative: 0.03 %
---
---
Image Sample Number: 50

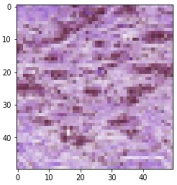


Image Label: 0
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 65

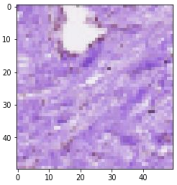


Image Label: 0
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %
---
---
Image Sample Number: 69

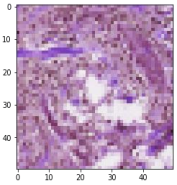


Image Label: 0
Image Prediction: 1
Confidence of Prediction: 93.41 %
Confidence of Alternative: 6.59 %
---
---
Image Sample Number: 77

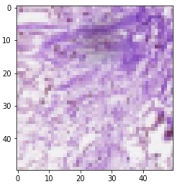


Image Label: 0
Image Prediction: 1
Confidence of Prediction: 81.72 %
Confidence of Alternative: 18.28 %
---
---
Image Sample Number: 87

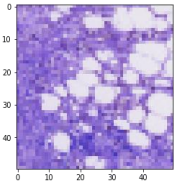


Image Label: 0
Image Prediction: 1
Confidence of Prediction: 98.5 %
Confidence of Alternative: 1.5 %
---
---
Image Sample Number: 97

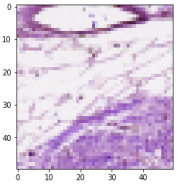


Image Label: 0
Image Prediction: 1
Confidence of Prediction: 99.09 %
Confidence of Alternative: 0.91 %
---
---
Image Sample Number: 98

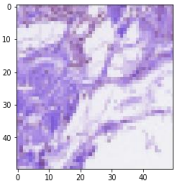
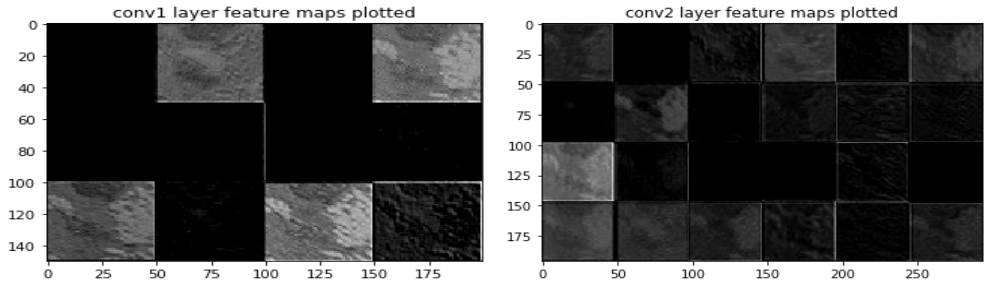


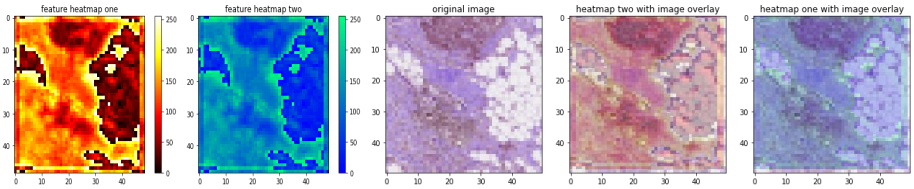
Image Label: 0
Image Prediction: 1
Confidence of Prediction: 98.41 %
Confidence of Alternative: 1.59 %

# 8.7 Appendix VII – Attention maps and predictions for stage 4.4.3

### *8.7.1.1 8100 - LR = 0.0022 – 12 filters in convolution layer 1*

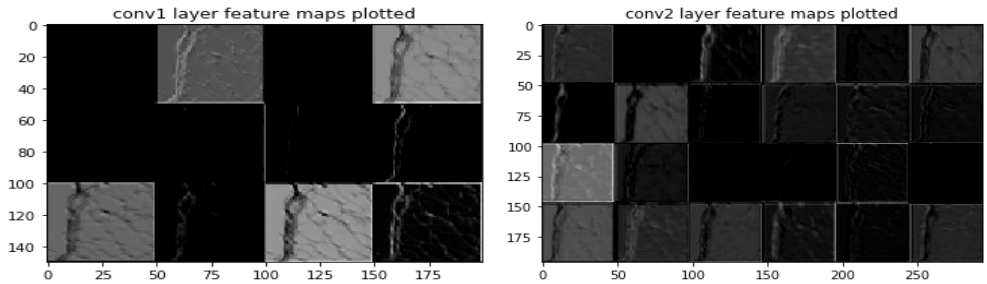
Image Label: 1
Image Prediction: 0
Confidence of Prediction: 99.77 %
Confidence of Alternative: 0.23199552670121193 %

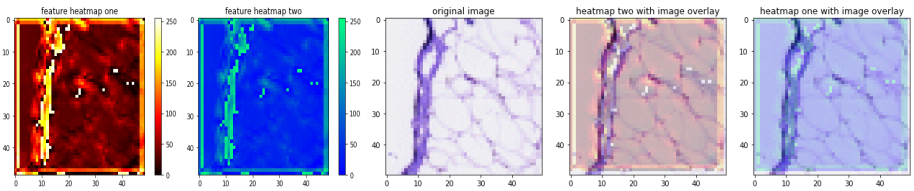



---

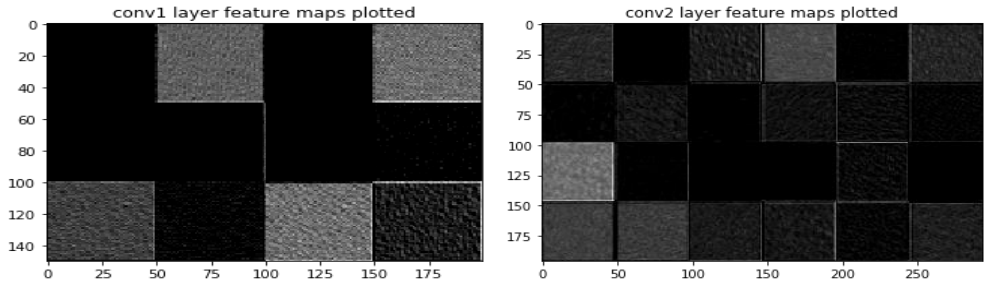
Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %

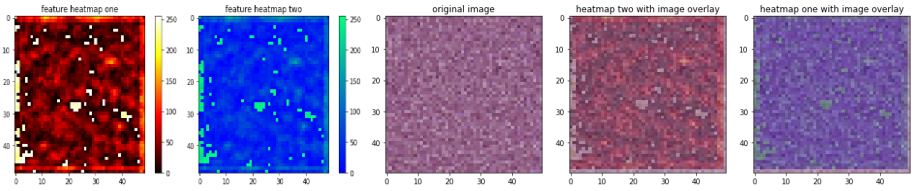



---

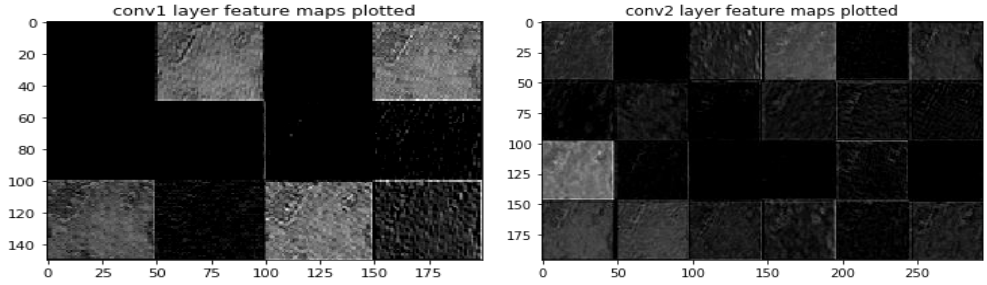
Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 9.866660101567528e-21 %

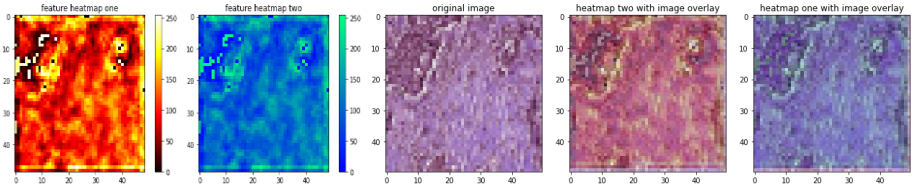



---

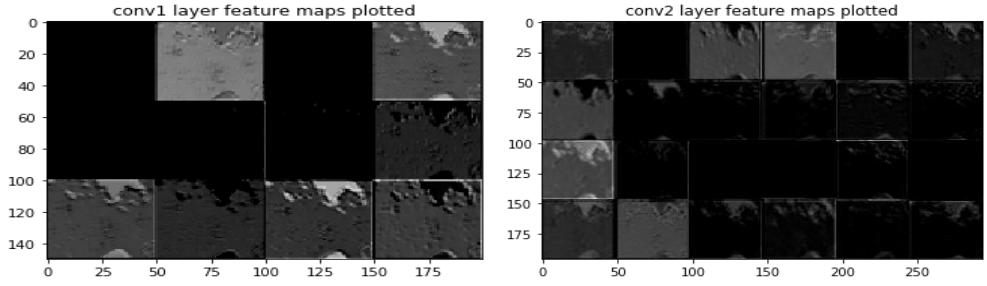
Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 1.4872506055172242e-31 %

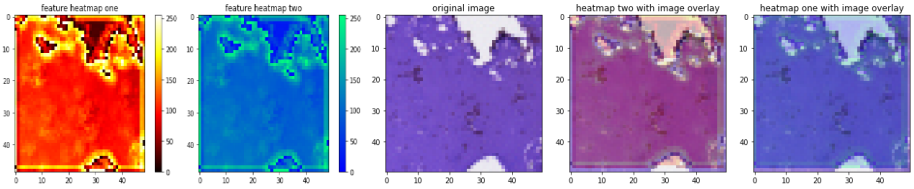



---

Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %

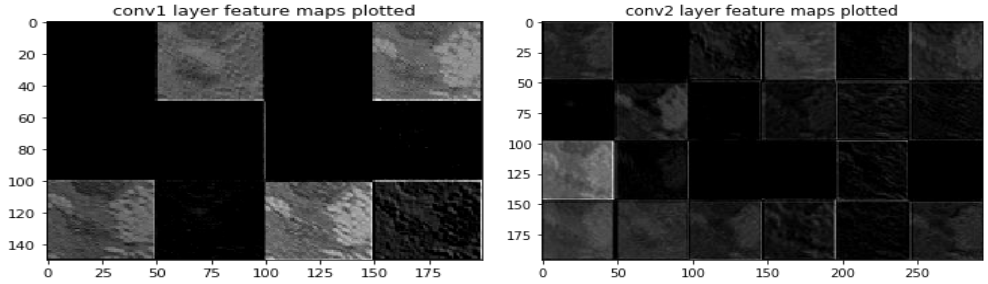


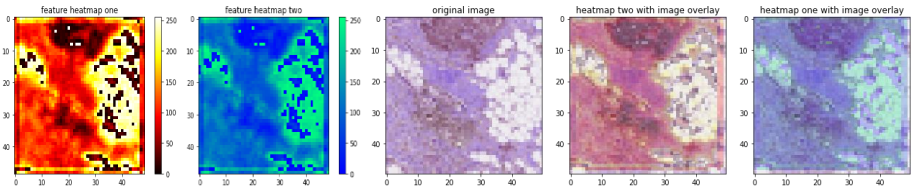



---

### *8.7.1.2 6000 - LR = 0.0027 – 12 filters in convolution layer 1*

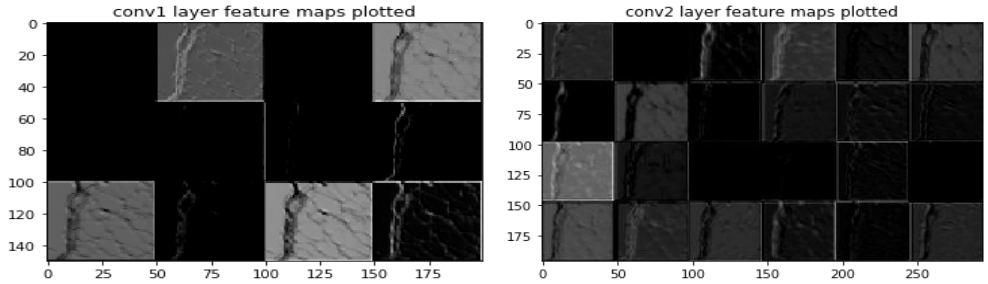
Image Label: 1
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 8.84735271711179e-06 %

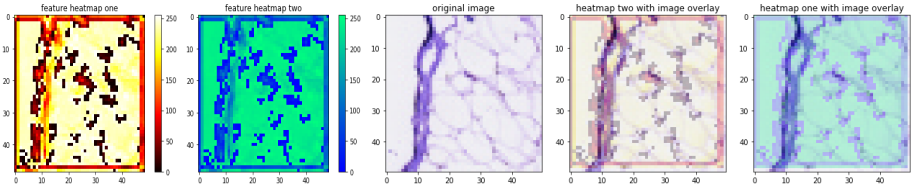



---

Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %

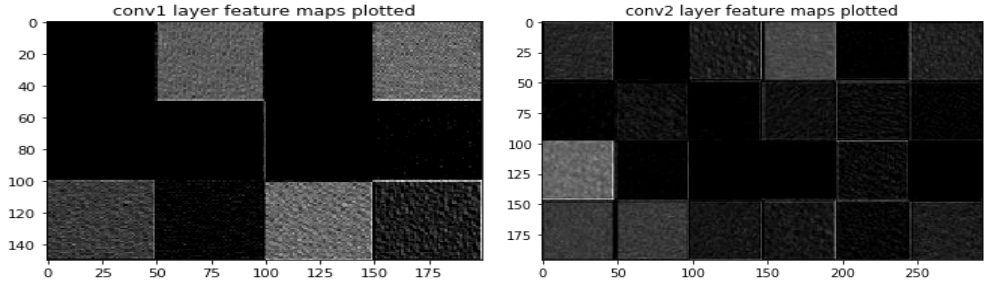


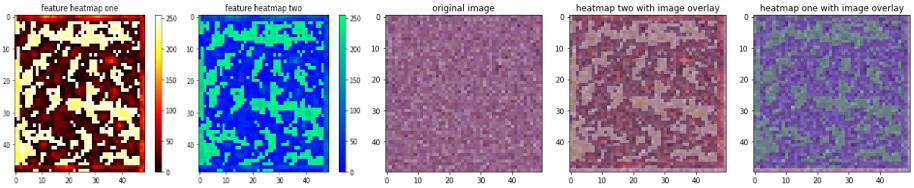




---

Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %

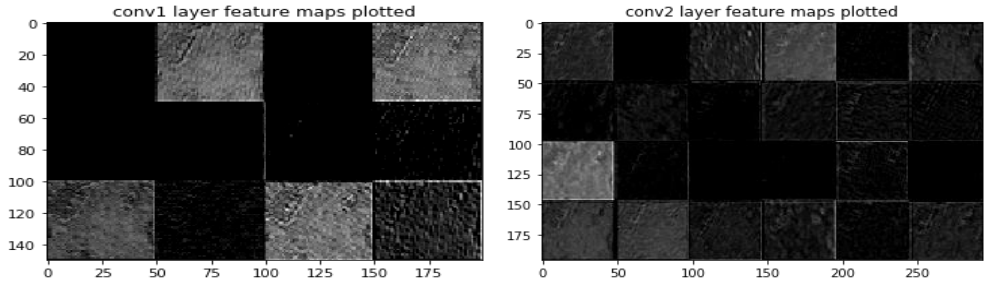
Confidence of Alternative: 1.1995501218760338e-19 %

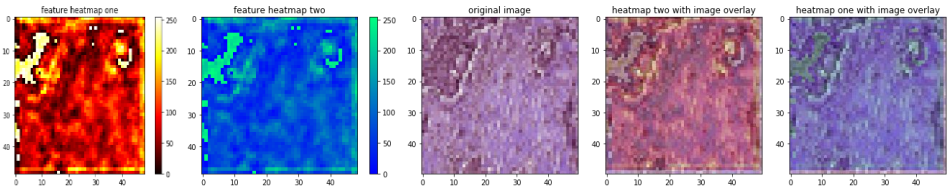



---

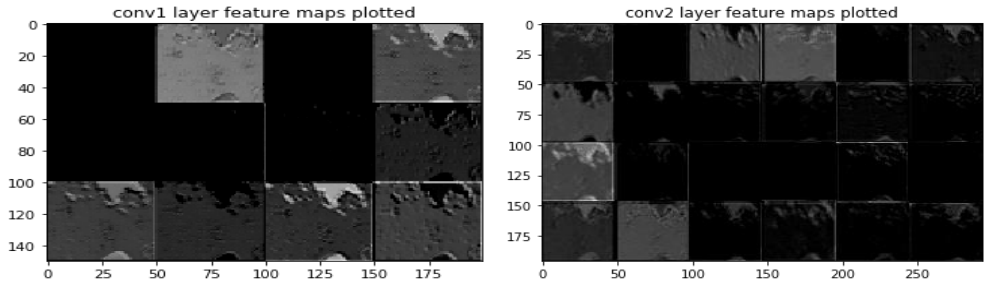
Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 4.610049704293016e-31 %

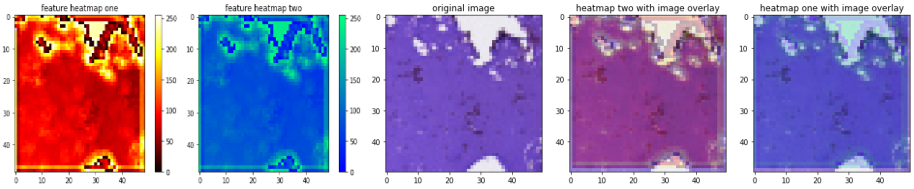



---

Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %

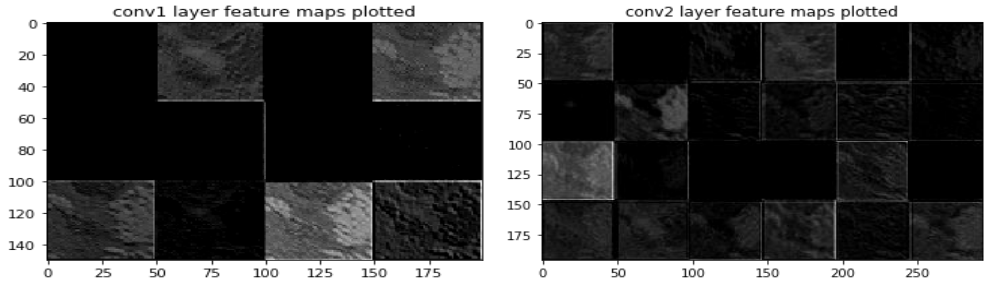


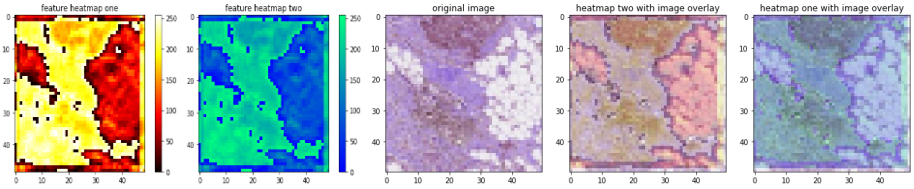



---

### *8.7.1.3 4000 - LR = 0.0053 – 12 filters in convolution layer 1*

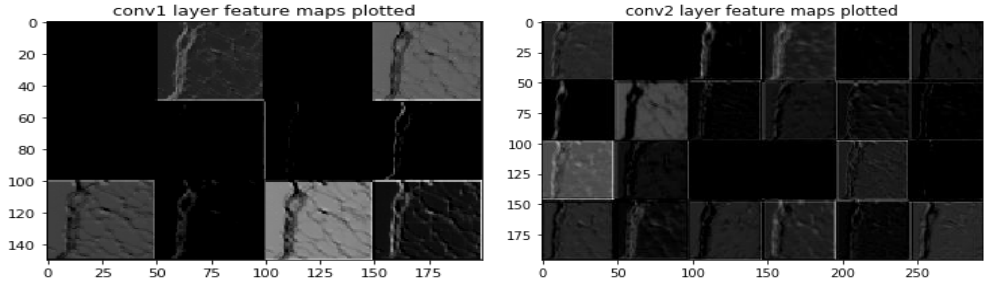
Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 4.135574945932373e-24 %

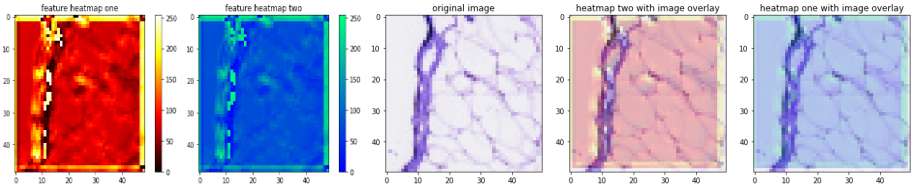



---

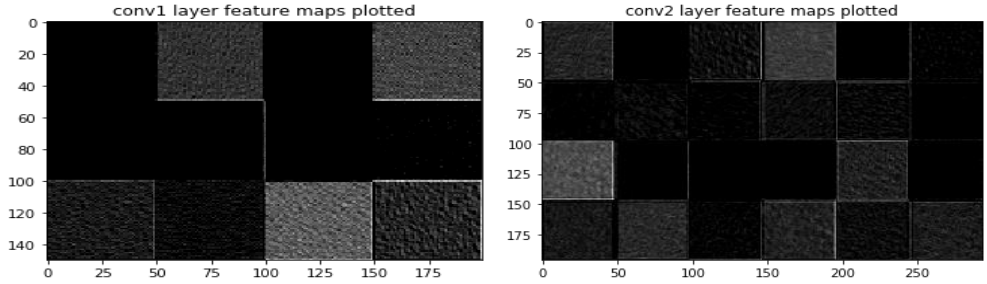
Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %

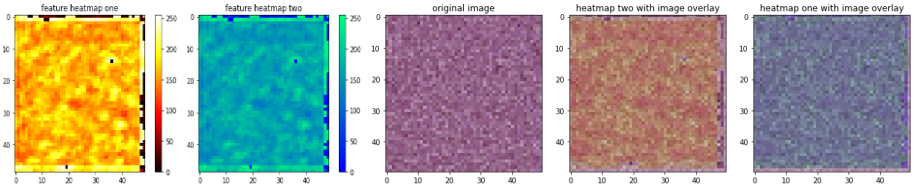



---

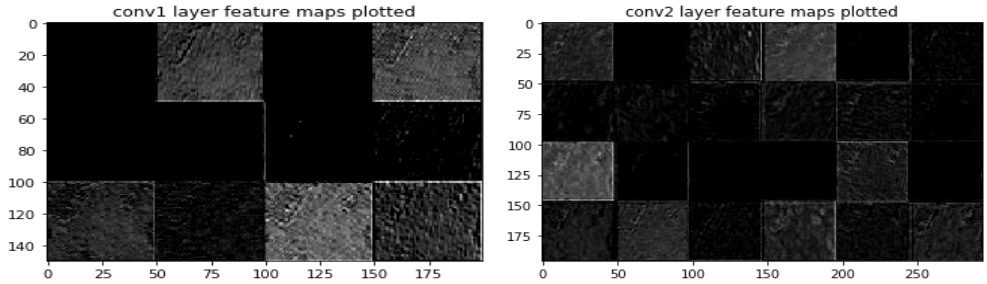
Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %

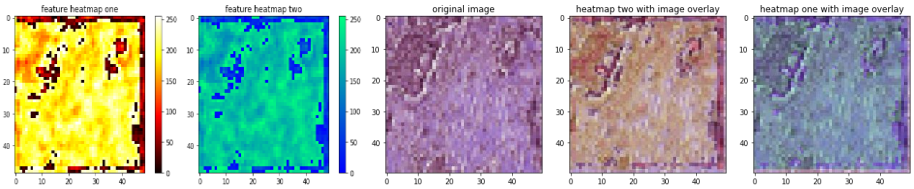



---

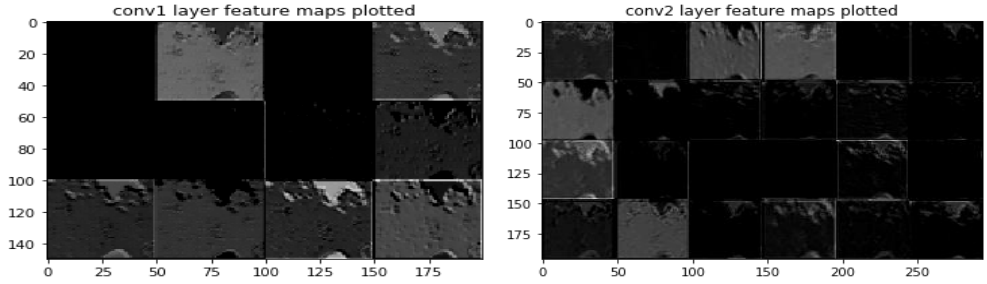
Image Label: 1
Image Prediction: 1
Confidence of Prediction: 100.0 %
Confidence of Alternative: 0.0 %

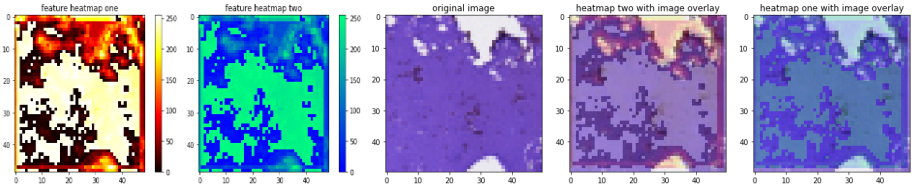



---

Image Label: 0
Image Prediction: 0
Confidence of Prediction: 100.0 %
Confidence of Alternative: 6.653353126951842e-31 %



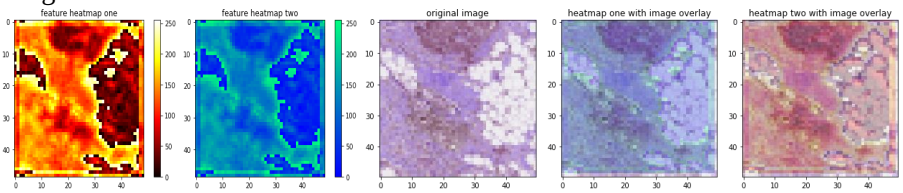



---

# 8.8 Appendix VIII – Attention maps and predictions for stage 4

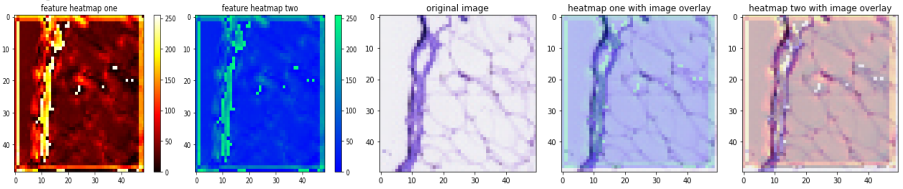
## 8.8.1 Test Case 1.1

Image Label: 1
Image Prediction: 0



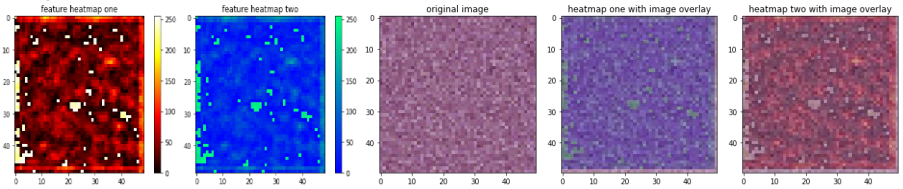

---

Image Label: 0
Image Prediction: 0



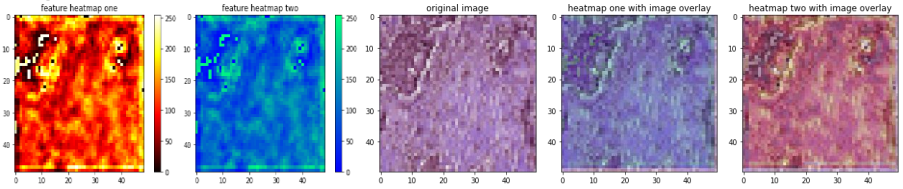

---

Image Label: 1
Image Prediction: 1



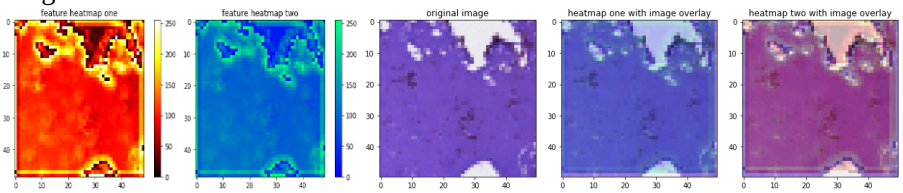

---

Image Label: 1
Image Prediction: 1



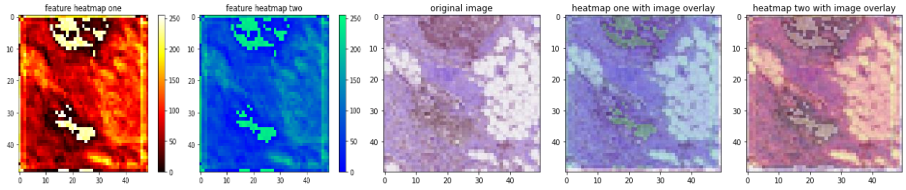

---

Image Label: 0
Image Prediction: 0



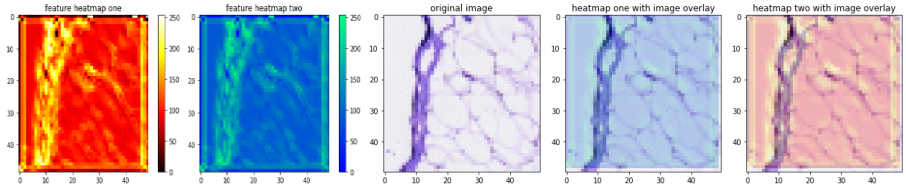
## 8.8.2 Test Case 1.2

Image Label: 1
Image Prediction: 1



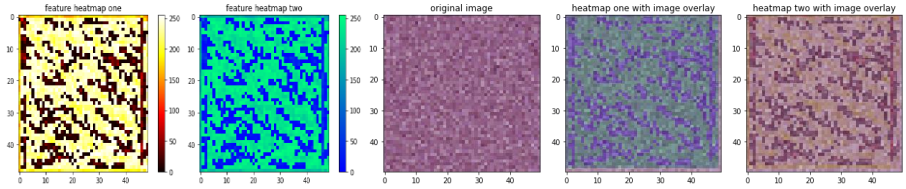

---

Image Label: 0
Image Prediction: 0



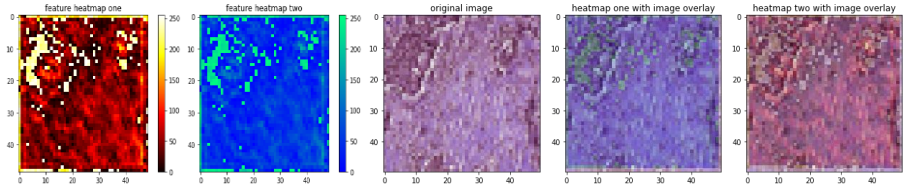

---

Image Label: 1
Image Prediction: 1



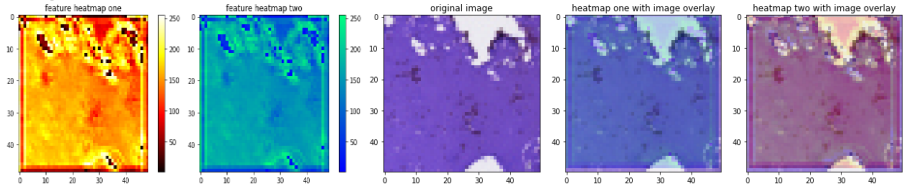

---

Image Label: 1
Image Prediction: 1



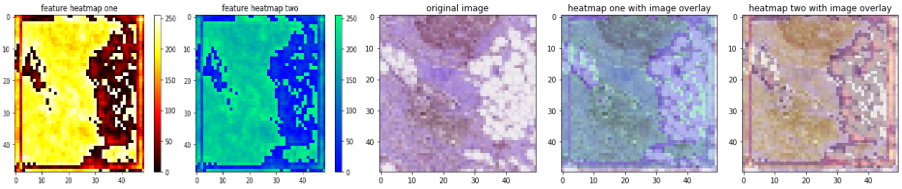

---

Image Label: 0
Image Prediction: 0



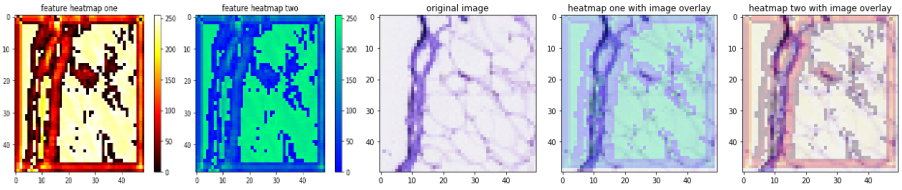

## 8.8.3 Test Case 1.3

Image Label: 1
Image Prediction: 0



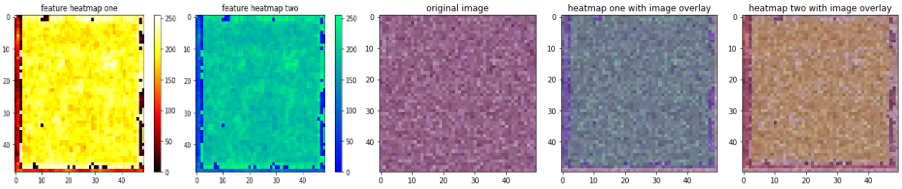

---

Image Label: 0
Image Prediction: 0



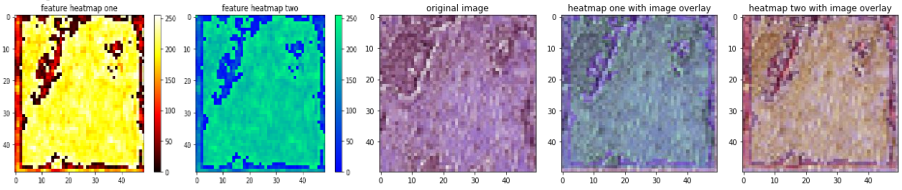

---

Image Label: 1
Image Prediction: 1



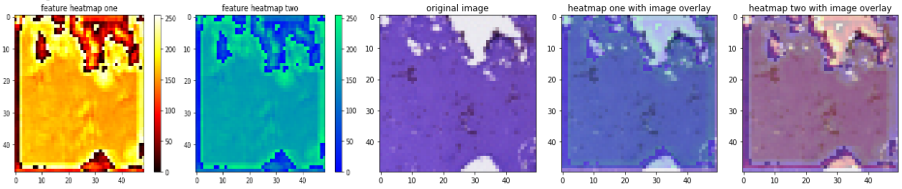

---

Image Label: 1
Image Prediction: 1



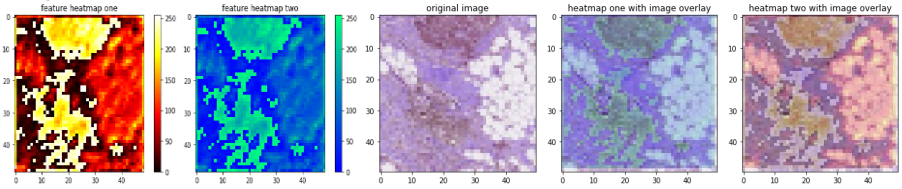

---

Image Label: 0
Image Prediction: 0



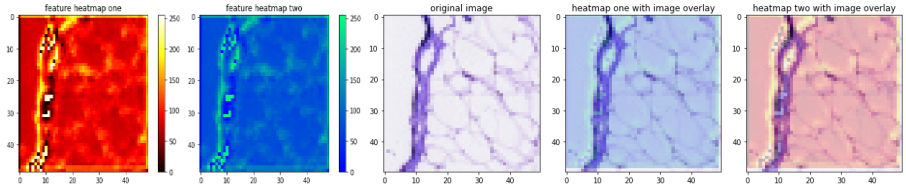

## 8.8.4 Test Case 1.4

Image Label: 1
Image Prediction: 0



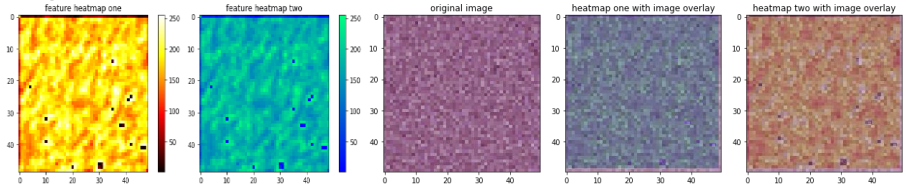

---

Image Label: 0
Image Prediction: 0



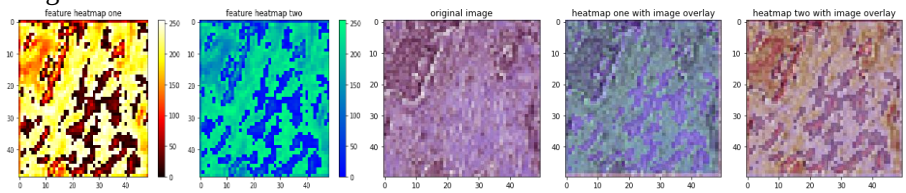

---

Image Label: 1
Image Prediction: 1



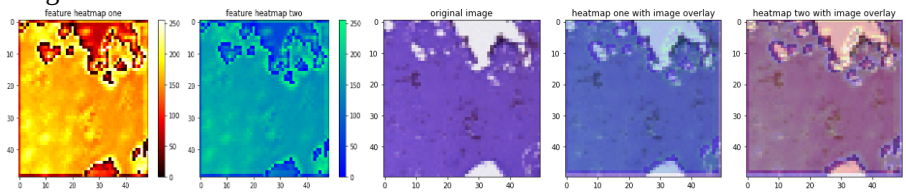

---

Image Label: 1
Image Prediction: 1



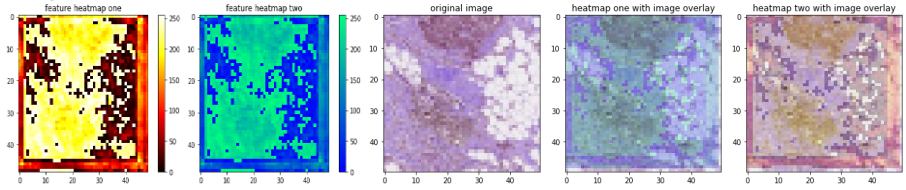

---

Image Label: 0
Image Prediction: 0



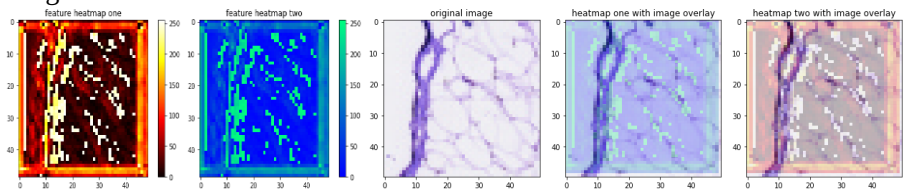

## 8.8.5 Test Case 1.5

Image Label: 1
Image Prediction: 0



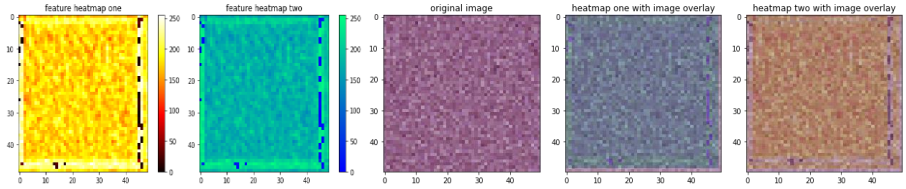

---

Image Label: 0
Image Prediction: 0



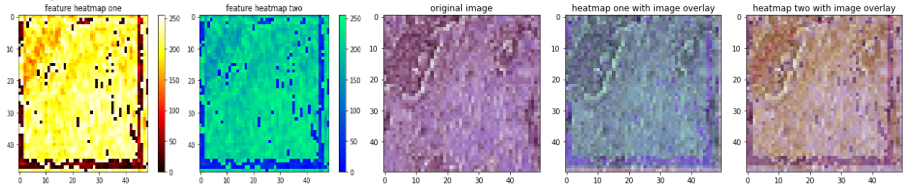

---

Image Label: 1
Image Prediction: 1



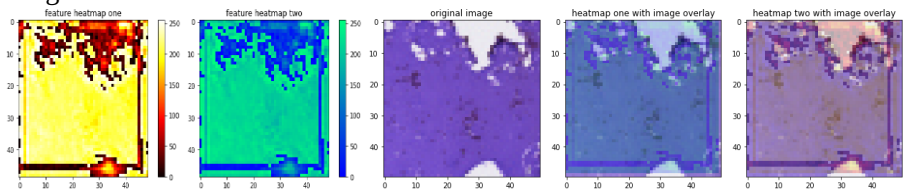

---

Image Label: 1
Image Prediction: 1



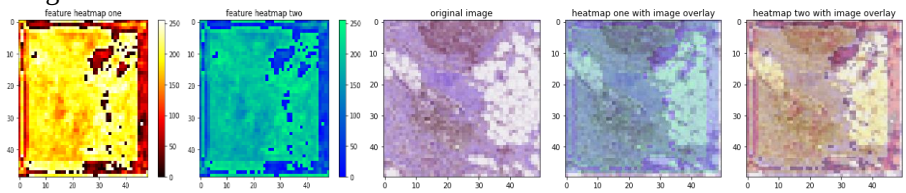

---

Image Label: 0
Image Prediction: 0



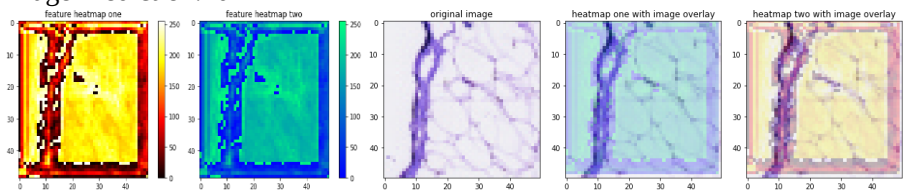

## 8.8.6 Test Case 1.6

Image Label: 1
Image Prediction: 0



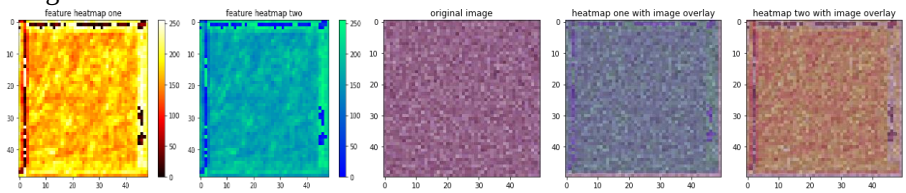

---

Image Label: 0
Image Prediction: 0



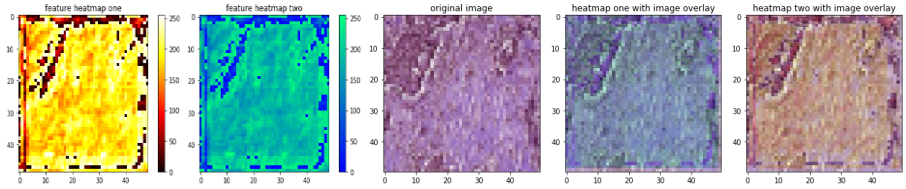

---

Image Label: 1
Image Prediction: 1



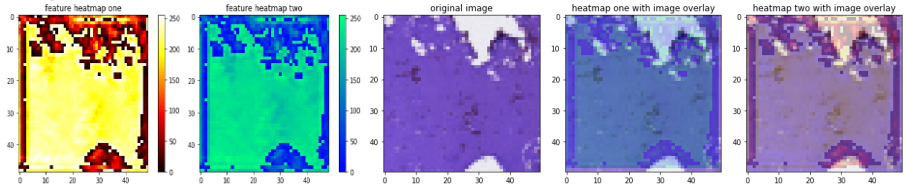

---

Image Label: 1
Image Prediction: 1




---

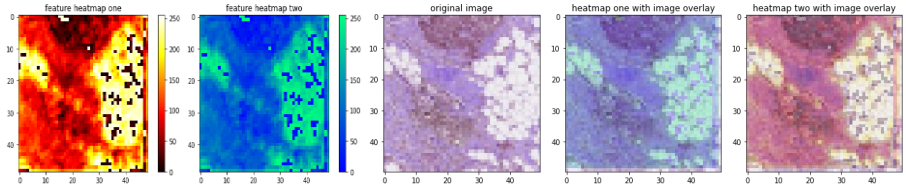
Image Label: 0
Image Prediction: 0




---

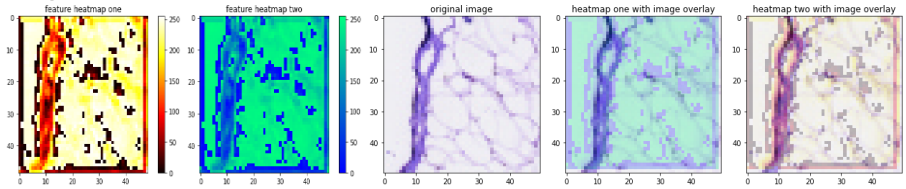
## 8.8.7 Test Case 1.7

Image Label: 1
Image Prediction: 0



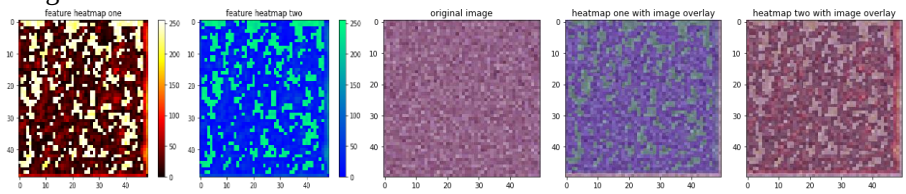

---

Image Label: 0
Image Prediction: 0



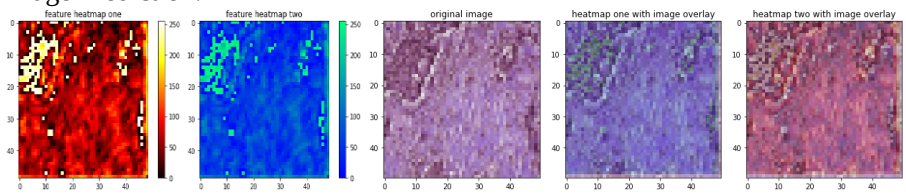

---

Image Label: 1
Image Prediction: 1




---

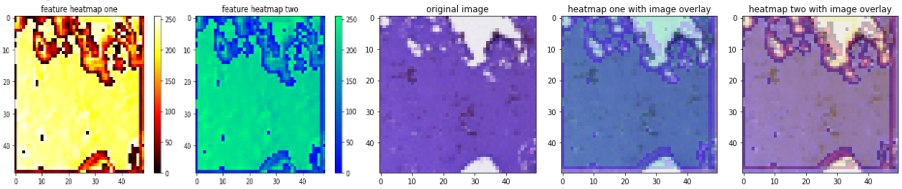
Image Label: 1
Image Prediction: 1




---

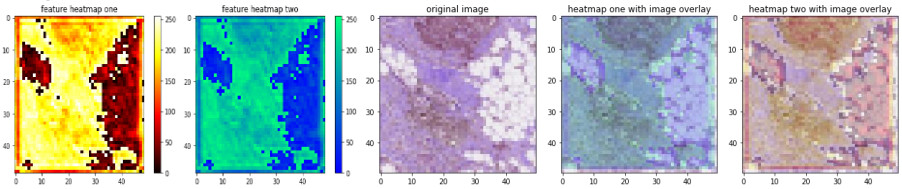
Image Label: 0

Image Prediction: 0



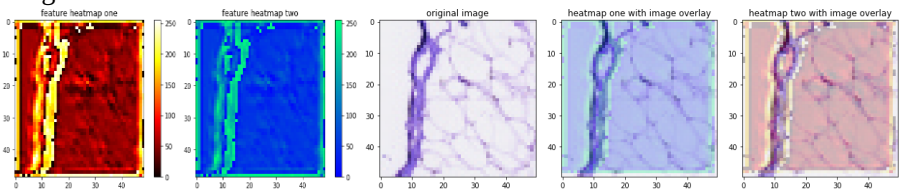

## 8.8.8 Test Case 1.8

Image Label: 1
Image Prediction: 0



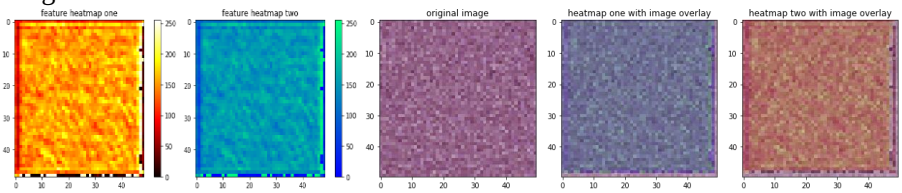

---

Image Label: 0
Image Prediction: 0



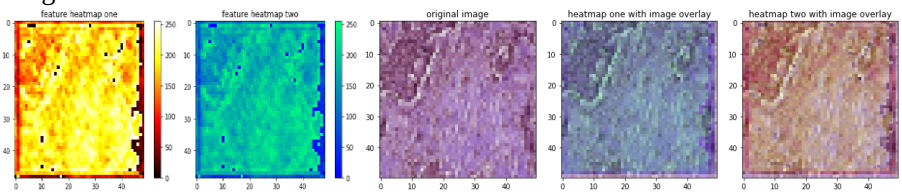

---

Image Label: 1
Image Prediction: 1



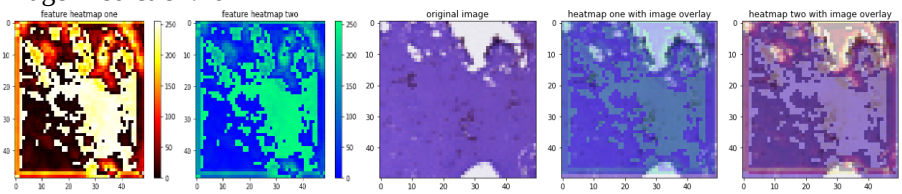

---

Image Label: 1
Image Prediction: 1




---

Image Label: 0
Image Prediction: 0




---