Detecting metastatic tissue in lymph nodes with deep learning

Marley Sudbury (1838838)

Supervisor: Paul Rosin Moderator: Dave Marshall

Computer Science With a Year in Industry (BSc)

School of Computer Science and Informatics
College of Physical Sciences
Cardiff University

Friday 27th May 2022

Abstract

To track the spread of a tumour, pathologists examine lymph nodes in the surrounding area for metastatic growths. This is done by looking at high resolution scans of slides with tissue taken from the lymph nodes. Deep learning systems have demonstrated a great ability to classify images, and so researchers have looked at applying these techniques to the problem of tissue classification, including lymph node classification, with the aim of creating a system which can aid pathologists. In this project a convolutional neural network (CNN) was used to evaluate two methods for classifying lymph nodes: a whole-slide method, which uses a down scaled version of the entire slide image as its input, and a patch-based method which takes small patches of the image as the input. The effects of data augmentation and stain normalisation were also investigated. Two datasets were used in this project, with one provided by the Head and Neck 5000 Study, and the other being the Camleyon16 training dataset. The best performance obtained from the whole-slide method was an area under the receiver operating characteristic curve (AUC) of 0.727, with a false positive rate (FPR) of 0.36 and a false negative rate (FNR) of 0.22. The patch-based method gave an AUC of 0.890, with an FPR of 0.21 and FNR of 0.24. This level of performance falls slightly short of the state of the art, and suggestions are provided for why this is and how it can be improved in future.

Acknowledgements

I would like to thank Dr Adam Jones and Dr Damian Farnell from the Cardiff University School of Dentistry for their assistance in understanding the background of this project from a medical perspective, the Head and Neck 5000 project for providing one of the datasets used and the organisers of the Camelyon16 project for making the other dataset used publicly available, Cardiff University IT Support for their assistance and for providing the hardware used in this project, the Computer Science Subject Research Ethics Committee (SREC) for answering my queries quickly and effectively, all the researchers in this area whose work I have referenced, Mitko Veta and Geoffrey Schau for making their stain normalisation code available under license and finally my project supervisor Paul Rosin and moderator Dave Marshall for their advice throughout this project.

Contents

List of Figures

List of Tables

1 Introduction

Head and neck cancer is a group of cancers which occur in the head and neck, such as mouth cancer, laryngeal cancer, throat cancers, salivary gland cancer and nasal and sinus cancer. It is a relatively uncommon form of cancer, with 12,422 cases diagnosed each year on average from 2016–2018 in the UK. In the same period, the UK had an average of 4,077 deaths from head and neck cancers per year. 19–59% of people with head and neck cancers survive for more than 10 years (Cancer Research UK 2022), with patient outcomes being dependent on the type of cancer, as well as factors such as age and sex.

Like all cancers, head and neck cancers can spread throughout the body via the lymphatic system (Cancer Research UK 2020). This results in secondary malignant growths, initially in lymph nodes near to the original tumour. This process of cancer spreading is called metastasis and affected lymph nodes are referred to as metastatic lymph nodes (MLNs). Pathologists need to monitor this spread, as spread to lymph nodes can lead to a higher grade of cancer with a lower chance of survival, and inform treatment options (NHS 2019). In mouth cancer, like many cancers, the TNM staging system is used, where T refers to the size of the tumour, N refers to the degree of spread to regional lymph nodes, and M refers to the presence of metastasis in distant organs.

Pathologists can determine if a lymph node is metastatic by removing it and examining a cross sectional image of it taken by a powerful microscope. The aim of this project is to create software which uses deep learning to classify these images of lymph nodes into those which contain metastatic tissue and those that do not. This can then be used to assist pathologists in their work, either by verifying their classification or guiding them to areas which do not appear to be normal tissue.

The primary beneficiaries of this work are pathologists, who could use this software to assist them in making diagnoses. Improvements to the accuracy of diagnoses also benefits patients who will have access to the best treatment depending on the severity of their cancer, therefore increasing their likelihood of survival.

The scope of the project is to evaluate the impact of different techniques on the effectiveness of a deep learning solution to perform metastatic lymph node detection. It will also look at the differences between different datasets, and the ability to generalise this classification across datasets.

There are two key approaches taken in this project. One is to use the entire slide as the input to the network (the whole slide method), and the other is to split the slide into small patches and use these patches as the input (the patch-based method). Of these two, the latter is the most widely explored in previous research, however the former has been used recently with some success.

The assumptions upon which this work is based are that the datasets used are correctly annotated at both the slide level (whether the lymph node shown in the slide contains positive tissue) and where applicable at the lesion level (which parts of the slide contain positive tissue). Another assumption is that the lymph nodes in the head and neck are similar enough to the lymph nodes in the breast that any difference would not affect the models ability to classify. This is as one dataset consists of lymph nodes from the head and neck, and the other from the breast.

2 Background

2.1 Histopathology

In the field of pathology, cancer spread can be diagnosed by checking lymph nodes near the site of the tumour for metastatic tissue. This is done by removing the lymph node, fixing it with a substance such as formalin so it doesn't degrade, dehydrating it by immersing it in alcohol, embedding it in paraffin wax to create a tissue block, cutting a thin slice from the block, putting that onto a slide and staining the slide with dyes (Feldman and Wolfe 2014). These dyes make the different tissues present more easily distinguishable. The principle staining used in histology is hematoxylin and eosin (H&E). Staining makes it possible for trained pathologists to identify the presence of metastatic tissue, although this process is time consuming. In addition to the time for an individual pathologist to examine a slide, more time is taken up by pathology review, which has been shown to be a critical process by Vestjens et al. (2012), who found that it lead to a change in 24% of breast cancer patients' node staging. A high resolution scan is taken of the slide, which is then examined by pathologists on a computer screen. This image is called a whole slide image (WSI), an example of which can be seen in Figure 2.1.

The precise colours present in a WSI can vary due to factors such as the concentration of dye used, the thickness of the tissue, and the use of different brands of slide scanners. To a trained human, the small differences in colour that arise usually have no impact on their ability to diagnose, however even minor lighting or colour shifts can cause issues for neural networks (Folmsbee et al. 2019). This is why it is best practice to normalise the staining before the data is used for training or is classified, and Ciompi et al. (2017) found a 28.7% increase in accuracy from doing so. Stain normalisation is discussed in more detail in section 2.5.

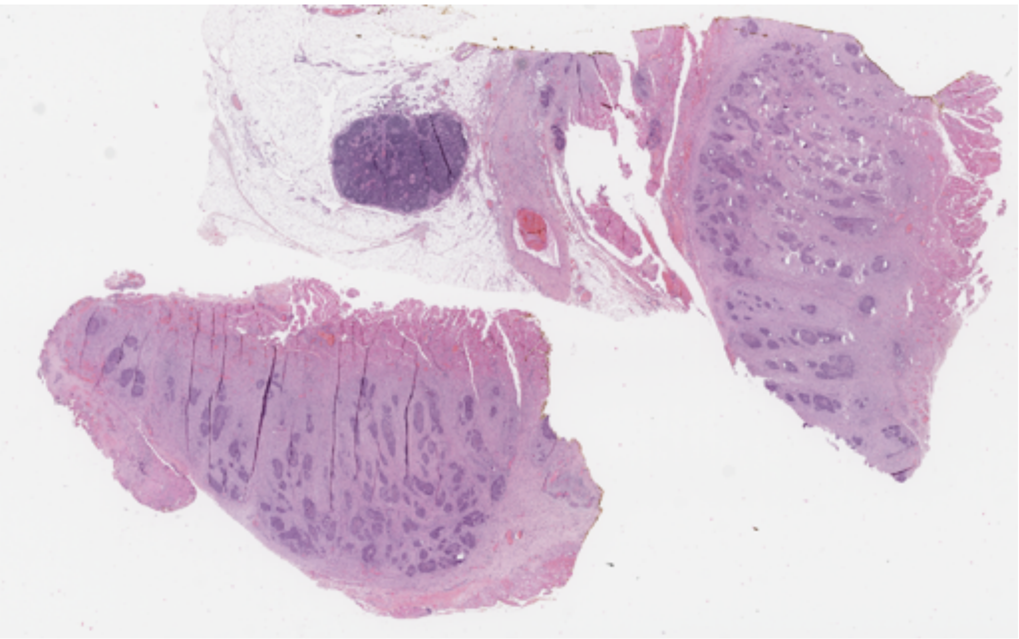

Figure 2.1: Example of a WSI of a lymph node stained with H&E.

2.2 State of the art

Deep learning has been seen as an attractive approach to improve the accuracy and time efficiency of WSI analysis and pathology review. The most common approach to deep learning in this field, such as that employed by X. Wang et al. (2021), is to use slide annotations which have been carefully constructed by a trained pathologist to train a neural network that can classify small patches of a WSI. An example of an annotated WSI can be seen in Figure 2.2, whilst Figure 2.3 shows how these annotation are represented in GeoJSON format.

In that project, their solution consisted of three 'modules'. Firstly, to locate the lymph node within the WSI, they trained a segmentation network. This used the U-Net architecture, which is a CNN that uses up convolution to create a segmentation map. This was trained on masks generated from 1× magnification thumbnails of the slides based on annotation provided by a pathologist. The next stage is a feature extractor which identifies certain features within patches of 768 × 768 pixels taken from slides at a ×20 magnification. Various architectures were tried for this, such as VGG19, AlexNet, Inception V4 and MobileNet V2, although they found

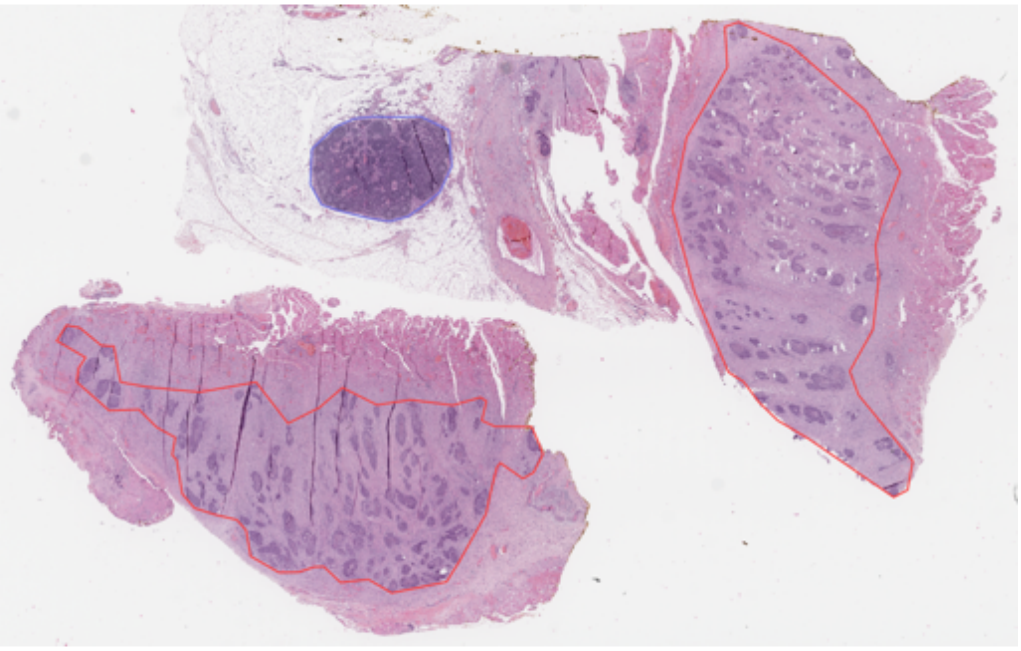

Figure 2.2: Example of a partially annotated WSI of a lymph node. The areas surrounded with red are positive / metastatic tissue, and the area surrounded in blue is negative / normal tissue.

that ResNet-50 achieved the highest classification accuracy. The final part of their solution was the use of the neural conditional random field, which models the spatial correlation of patches to give a set of confidence values for tumour regions.

Another approach is to train a neural network to diagnose the entire WSI in one go. This is infeasible on most computer systems, as an entire WSI at full resolution may take up 10s of gigabytes. However, it is possible to do on a system with lots of memory using the unified memory technique, where all RAM and VRAM in a system can be accessed as a single address space. This was shown by Chen et al. (2021) to achieve excellent performance (an AUC of 0.9594 on classifying adenocarcinoma), although in that study they were classifying types of lung cancer rather than lymph nodes. The key advantage of this approach is that the training data does not need to be annotated at the lesion level, but only at the slide level. This means that less time and effort needs to be spent in preparing the training data, and so more data can be used. It also allows for the possibility of the model picking up on things which may have been missed by a

```
{
"type":"FeatureCollection",
"features":[{
    "type":"Feature",
    "geometry":{
        "type":"Polygon",
        "coordinates":[[
            [17289, 27134],
            [17241, 27136],
            . . .
            [17337, 27136],
            [17289,27134]
        ]]
    },
    "properties":{
        "object_type":"annotation",
        "classification":{
            "name":"Tumor",
            "colorRGB":-3670016
        },
        "isLocked":false
    }
}]
}
```

Figure 2.3: Example of slide annotation in the GeoJSON format used by QuPath. “colorRGB” value is stored in two’s complement 32 bit integer format. $-3670016 = \underbrace{11111111}_{\text{alpha}}\ \underbrace{11001000}_{\text{red}}\ \underbrace{00000000}_{\text{green}}\ \underbrace{00000000}_{\text{blue}} = (200, 0, 0)$.

human, such as micrometastases. They first downscaled the slides in their dataset from ×20 magnification to ×4 magnification and padded them to 21,500 × 21,500 pixels. They then used the ResNet-50 architecture with fixup initialisation, which they trained as a ternary classifier with the labels adenocarcinoma, squamous cell carcinoma, and non-cancer. The final pooling operations they used were global average pooling (GAP) and global max pooling (GMP), of which they found that GMP performed far better, with an AUC of 0.9594 for adenocarcinoma compared to 0.6506 achieved by GAP. They suggest that this difference is caused by GAP losing subtle information presented in tiny features within the image. They trained on 5,606 slides, and used 1,397 slides for evaluation. They also implemented various multiple instance learning (MIL) approaches to compare against. These are patch-based approaches where training data is not labelled at the patch level, but is labelled as a bag. For example, a bag labelled positive would include at least one patch that is positive, whilst a bag that is negative would contain no positive patches (Babenko 2008). Of these, they found that MIL with max feature aggregation and random forest slide-level aggregation performed the best, but slightly worse than their proposed whole-slide method. The results for these methods are included in Table 2.1.

The Camelyon16 Challenge which ran in 2016 was designed to find the best performing algorithms at classifying lymph nodes in breast cancer patients, and compare these with the performance of pathologists. In the paper published after the conclusion of the challenge, Ehteshami Bejnordi et al. (2017) reported that the best performing algorithm (Harvard Medical School and MIT Method 2) achieved an AUC of 0.994, whilst a pathologist with no time constraint achieved 0.966 and a team of 11 pathologists with a flexible time constraint of 2 hours to classify 129 slides achieved an average AUC of 0.810. This best performing algorithm used the GoogLeNet architecture proposed by Szegedy et al. (2014), which is a 27 layer deep CNN which includes ‘inception modules’, where the input layer goes through different sizes of convolution simultaneously, the outputs of which are concatenated together for the output. The solution also used stain normalisation and ‘extensive’ data augmentation.

Please see Table 2.1 for a summary of the performance for various existing solutions in this area. This table uses the terms AUC, accuracy, sensitivity and specificity, which shall be defined here. Accuracy is the percentage of classifications which were correct. Specificity, also called the true negative rate, is the percentage of negative samples which were correctly labelled. Sensitivity, also called the true positive rate, is the percentage of positive samples which were correctly labelled. AUC stands for area under the receiver operating characteristic curve. This curve represents the relationship between the sensitivity and the specificity at various classification thresholds. By taking the area underneath the curve we get a value which can indicate how successful the model is at classifying generally, regardless of the threshold.

These same values for the solutions made in this project will be given in chapter 5. "—" indicates that corresponding value was not published.

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

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

2.3 Selection of datasets

To evaluate the transferability and robustness of any solution produced in this project, two datasets will be used. This allows for one dataset to be used for training, and the other to be used for evaluation. Therefore, any overfitting to the training data should be shown in the evaluation.

The first dataset was kindly provided by the Head and Neck 5000 study. This is anonymised secondary data, and was provided with only slide-level annotation. Lesion-level annotation for 10 positive slides in this dataset was provided by Dr Adam Jones (an example is shown in Figure 2.2) so that the patch-based method could be applied to this data. There are 65 positive slides and 45 negative slides in this dataset.

No publicly available lymph node slide datasets associated with head and neck cancer were found, however since the structure of lymph nodes around the body are similar, any dataset of lymph nodes should be useable. For the second dataset, the Camelyon16 training dataset was chosen, containing WSIs of lymph nodes from breast tissue. This comes with both slide-level and lesion-level annotations. The full Camelyon16 training dataset contains 160 positive slides and 110 negative slides, however due to the amount of storage space available only the first 50 negative slides and the first 50 positive slides were used in this project.

2.4 Data augmentation

It is common practice when training a neural network to classify images, to apply some form of augmentation to the images being used as training data. This is done to prevent over fitting. J. Wang and Perez (2017) showed that

Paper	**AUC**	**Accuracy**	**Specificity**	**Sensitivity**
Harvard Medical School and MIT, Method 2 (Ehteshami Bejnordi et al. 2017)	0.994	—	—	—
X. Wang et al. (2021) (slide level)	0.990	96.9%	96.1%	98.5%
X. Wang et al. (2021) (patch level)	0.986	—	—	—
Pathologist without time constraint (Ehteshami Bejnordi et al. 2017)	0.966	—	98.7%	93.8%
Chen et al. (2021) (whole-slide)	0.9594	—	—	—
Chen et al. (2021) (MIL)	0.9345	—	—	—
Mean of 11 pathologists with time constraint (Ehteshami Bejnordi et al. 2017)	0.810	—	98.5%	62.8%

Table 2.1: The performance of various existing solutions.

affine transformations such as scaling the images, rotating them, cropping them, reflecting them and translating them are effective in increasing the validation accuracy of a deep learning model. By using these traditional augmentation techniques they were able to improve validation accuracy of classifying dogs vs cats from 0.705 to 0.775. Data augmentation is also used to prevent a model from overfitting to the training data. Transformations such as the above are built in to Tensorflow as Keras pre-processing layers which can be added to the beginning of the model. When used for classification, these layers are not activated.

2.5 Stain normalisation

As discussed in section 2.1, the result of staining a slide can vary slightly between labs and even in the same lab over time. As well as the differences in staining, the colours captured by different scanners can vary considerably, even when scanning the same slide (Vahadane et al. 2016). To a trained human pathologist, these slight differences usually make no impact on their ability to accurately diagnose, however for a neural network it can potentially cause misclassification.

The stain normalisation method used in this project is the one proposed by Macenko et al. (2009). When this method was evaluated by Ciompi et al. (2017), they found that it lead to a significant increase in accuracy (more than 20% when applied to both training data and evaluation data), although they also experienced some issues where it failed to normalise patches that were of an unexpected tissue type (adipose tissue). This method was not implemented as part of this project, as a Python implementation has already been created and published by Veta and Schau (2020).

This method for stain normalisation has two key steps. Firstly, ‘stain vectors’ are automatically calculated for each of the stains. These are vectors which describe the proportion of red, green and blue light wavelengths that are absorbed by the stain. To obtain these vectors, the pixels are transformed from RGB space to optical density (OD) space using the following equation:

$$OD = -log_{10}(I)$$

where I is the RGB colour vector of the pixel, with each component normalised to between 0 and 1. This transformation is important, because in OD space the two stains can be separated by a straight line, which is not possible in RGB space. A single value decomposition (SVD) is then calculated for each pixel in the OD space, which is a representation of the OD value using eigenvalues and eigenvectors. A plane is created from the two largest SVD values. The data is projected onto this plane, and the angle is calculated of each point from the first SVD direction. The α^{th} and $100 - \alpha^{th}$ percentiles of the angle are converted back to OD space and returned as the optimal stain vectors (the authors recommend that $\alpha = 1$, but state that good results are obtained from a range of values).

The next step is to normalise the stains. For each stain, the intensity histogram is calculated of all pixels which have a majority of that stain.

The 99$^{\text{th}}$ percentile of the intensity histogram is taken to be an approximation of the maximum intensity. By scaling the histograms to have the same ‘pseudo-maximum’, the stain is then normalised. This method also allows separation of the stains into separate images by deconvolving with the determined stain vectors, although this will not be used in this project.

3 Specification and design

3.1 Requirements

- Read in an image in WSI format (such as .tif or .svs).
- Return a label for the image, and a set of labels for regions within the image.
- Return region labels as an annotation file so that pathologists can view the results on the slide.
- When run on evaluation data return accuracy, specificity, sensitivity and AUC.
- Be capable of running within a reasonable amount of time.

3.2 Reading WSIs

WSIs come in several different file formats, such as .svs and .tif. Fundamentally, they are all similar in that they contain a compressed high resolution scan of a slide, as well as several down-sampled versions of the same image. This is done so that slide viewing software such as QuPath can show the slide at different zoom levels without running out of memory. These layers can be thought of as an image pyramid (see Figure 3.1).

Most standard image processing libraries struggle with these WSI formats, as they usually attempt to load the whole image into memory at the same time, and full-resolution WSIs are usually 10s of GBs. One way of interacting with WSIs is the OpenSlide library. This allows for loading the slide and extracting regions of it from the desired scale level. Alternatively, a lower resolution of the slide can be extracted from the file into memory. These two approaches will be the bases of the methods used in this project, and therefore OpenSlide makes up the first part of the pipeline.

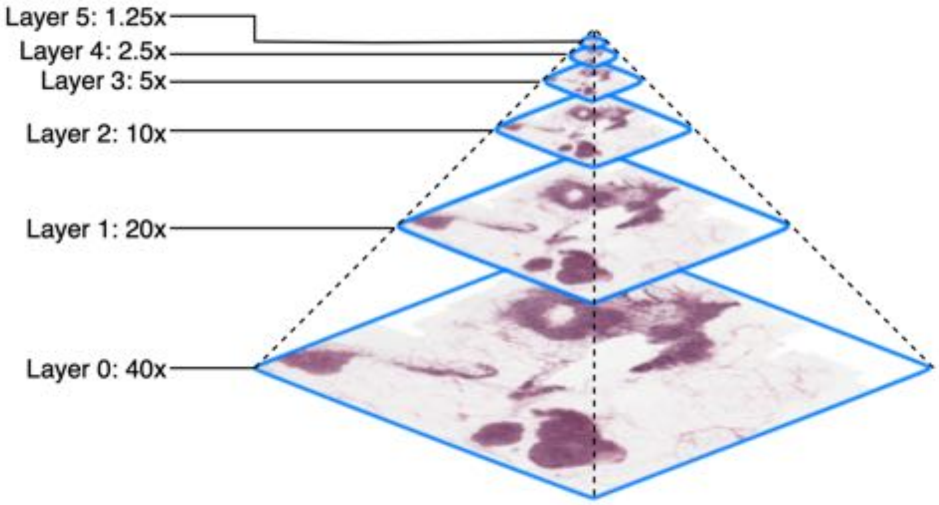

Figure 3.1: Visual representation of WSI layers in the Camelyon16 dataset.

3.3 Deep learning architecture

An overview the architectures used by different researchers for deep learning approaches to lymph node classification is given in section 2.2. The architecture used in this project, shown in Table 3.1, is based on that used by Ciompi et al. (2017) in their work evaluating the effectiveness of stain normalisation on tissue classification.

7 weight layers										
conv5-32	MP	conv5-64	MP	conv3-128	MP	conv3-256	MP	conv9-1024	conv1-512	SM

Table 3.1: The neural network structure used in this project, based on Ciompi et al. (2017). MP = max pooling layer, SM = softmax layer.

Ciompi et al. based this architecture on work by Simonyan and Zisserman (2014), who showed that by having more layers with a smaller filter size, higher discrimination with less parameters can be achieved compared to fewer layers with a larger filter size. The architecture consists of 11 layers, alternating convolutional layers with max-pooling layers, except for between the last two convolutional layers. The final layer in the architecture is a softmax layer which gives a probability for each class.

The convolutional layers use rectified linear unit (ReLU) activation. The first layer has 32 filters, with the number of filters doubling for each convolutional layer. The filters are of size 5 in the first two convolutional layers, with size 3 in the third and fourth, size 9 in the fifth layer and size 1 in the final layer. This was chosen because Ciompi et al. attained an accuracy of 93.8% at classifying tissue types within a 150 × 150 pixel patch. The architecture uses the ADAM algorithm for updating parameters during training.

3.4 Hardware

Training a machine learning model is most efficiently done when done using graphics cards. In this project an NVIDIA GeForce RTX 2070 graphics card was used, in an Ubuntu workstation from the Linux Lab of the Cardiff University School of Computer Science (Abacws room 5.44). The OpenSlide library referred to in section 3.2 could not be installed on that workstation as this would require adminstrator rights. Therefore, generating training data at training time, or during each epoch of training, as has been done in some previous research (X. Wang et al. 2021), was not possible, and training data had to be generated in advance of training.

3.5 Interface

The user interfaces with the system exclusively using the command line. If this software were to be packaged into a commercial application, it would be important to provide a graphical user interface, as pathologists may not be used to working with the command line.

The patch-based classification approach can create a lesion level annotation which can be imported into QuPath. This interface is user friendly and familiar to pathologists.

3.6 Code structure

Many of the image processing elements, such as scaling, extracting patches and applying stain normalisation, are needed in both the training and classification sections of the project. For this reason, the common code was put into an image pipeline class. Other useful utilities such as a path handler are contained within a utils module, so they can be accessed by all parts of the program. The architecture of the machine learning model (shown in Table 3.1) is given its own class within the models module. This is so that the code can be loaded from both the trainer and classifier programs. It was initially planned to try several different architectures, but due to the limited time available for this project, only one was implemented.

All code for this project was written in Python 3, as there are powerful machine learning libraries available, such as the TensorFlow library which is used in this project, and it has a convenient virtual environment to manage all the libraries involved in the project. It also has an historical precedent in this field of research, being the language used by X. Wang et al. (2021) in their research into the use of deep learning to predict gastric cancer outcome, although they used the PyTorch machine learning library.

3.7 Whole-slide approach

The training data for the whole-slide approach, generated from the image pipeline, consists of 100 × 100 px images with the lymph node in the middle. The edges are padded with white pixels. One problem of down-scaling the slides and putting them within a square like this, is that it could lead to most of the image being empty, such as in Figure 3.2. This is not such a big problem for the data from the Head & Neck 5000 study, as the slides are already cropped to only include the lymph node section. This cropping is not perfect, as in some cases it has included logos or writing on the slides that are not part of the lymph node. However, the Camelyon16 dataset is not cropped like this, and so when they are scaled down a larger amount of detail is lost than for the Head & Neck 5000 data. The annotation data available for the Camelyon16 dataset only specifies the presence of lesions, and so cannot be used to determine the lymph node section of the slides. The approach to this problem taken by X. Wang et al. (2021) was to train another neural network to locate the lymph node within the slide. It may also be possible to use other computer vision techniques such as SIFT to locate the lymph node area within the slide, however in this project no solution to this issue was implemented. Therefore a difference in scale between the training data and evaluation data will likely have an effect on models using this approach in the evaluation.

The purpose of this approach can be seen as evaluating whether there are significant enough structural differences at the whole slide level to allow a neural network to differentiate between positive and negative nodes.

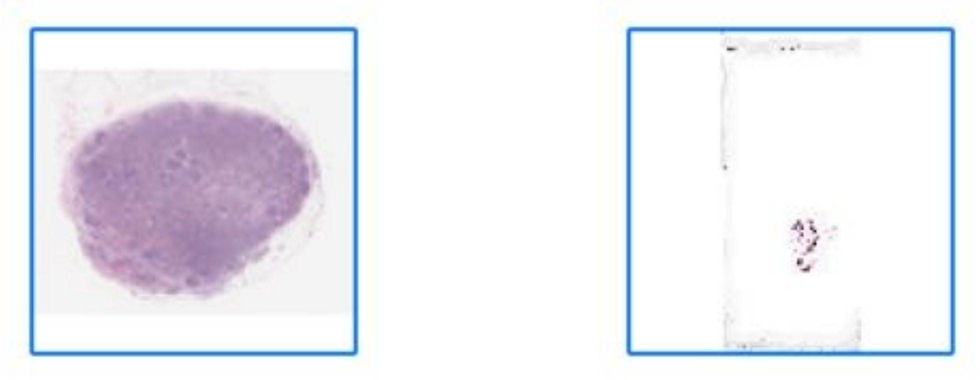

Figure 3.2: Left, maximum information retained due to smart cropping (Head and Neck 5000). Right, unnecessary information loss (Camelyon16).

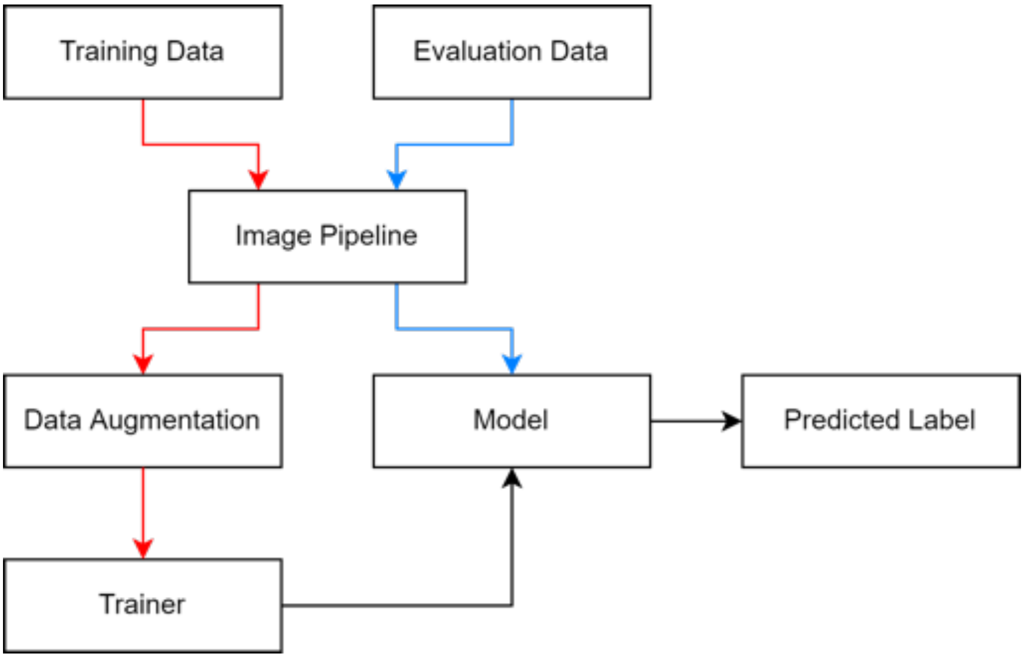

Figure 3.3: The flow of data through the system.

3.8 Patch-based approach

The patch-based approach to slide classification is more popular than the whole-slide approach, as the input to the network can be at a low resolution whilst still capturing important details from the slide. The first step in this approach is to have WSIs which have been hand annotated by a pathologist using a program such as QuPath or Aperio ImageScope. This annotation process yields a file with coordinates describing one or more polygons within the slide, and the classification for the tissue within each polygon. Then patches can be extracted from the slides using the OpenSlide library and labelled with the correct classification of either positive or negative. These are then used by a neural network to train. This network can then be given a patch from a slide it hasn't seen before and return a predicted classification. Chen et al. (2021) showed that the classification process can be spread up considerably by ignoring all patches with RGB values larger than 220, as these would likely be background. This will also be applied in this project.

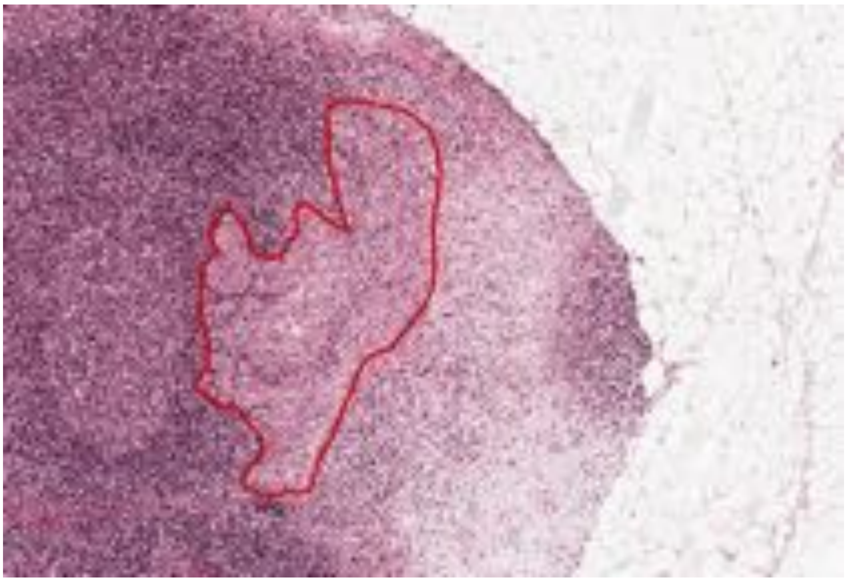

Figure 3.4: An example of a positive tissue annotation provided with the Camelyon16 dataset, displayed with QuPath.

Once every patch within a new slide has been classified by the model, this can be converted into a new annotation file which can be provided to a pathologist for review. This annotation would draw lines around the areas in the slide that the model has classified as positive.

In Table 3.2 and Table 3.3, the uncompressed size of each layer in gigabytes is calculated like so:

$$\text{Size (GB)} = \frac{\text{width} \times \text{height} \times 3}{1024^3}$$

Layer	**Ratio**	**Resolution**	**Size**
0	1:1	97792 × 221184 px	60.43GB
1	1:2	48896 × 110592 px	15.11GB
2	1:4	24448 × 55296 px	3.777GB
3	1:8	12224 × 27648 px	0.9443GB
4	1:16	6112 × 13824 px	0.2361GB
5	1:32	3056 × 6912 px	0.05902GB
6	1:64	1528 × 3456 px	0.01475GB
7	1:128	764 × 1728 px	0.003689GB
8	1:256	382 × 864 px	0.0009221GB
9	1:512	192 × 438 px	0.0002350GB

Table 3.2: The pixel ratio and uncompressed size of WSI file layers in the Camelyon16 dataset as read by OpenSlide. Layer 0 is scanned at 40× magnification.

Layer	**Ratio**	**Resolution**	**Size**
0	1:1	53784 × 52470 px	7.885GB
1	1:4	13446 × 13117 px	0.4928GB
2	1:16	3361 × 3279 px	0.03079GB
3	1:32	1680 × 1639 px	0.007693GB

Table 3.3: The pixel ratio and uncompressed size of WSI file layers for a slide in the Head and Neck 5000 dataset as read by OpenSlide. Layer 0 is scanned at 40× magnification. Note that unlike the Camelyon16 dataset, the resolution varies slightly from slide to slide. However, the magnification and order of layers is consistent across the dataset.

and is given to four significant figures. The actual files are much smaller than the numbers given in the tables, as the images are stored with compression.

4 Implementation

4.1 Model architecture

For both the whole-slide and patch-based approaches, the network structure shown in Table 3.1 was used. This structure was used by Ciompi et al. (2017) with good results. The differences are that in this project the input size is 100 × 100 px rather than 150 × 150 px, and the output is a 2 element vector rather than a 9 element vector. This is because the aim of this project is to identify tumour tissue within a lymph node, whereas in that work they classified all the types of tissue present in a slide taken from a tumour.

In this project, the structure was implemented in Tensorflow using the `keras.Sequential` class. The models were trained on their input data for 1000 epochs before being evaluated, and a batch size of 64 was used. Training was ended early in some instances where the loss metric of the model did not change for three or more epochs. Different pre-processing steps (outlined in chapter 3) were applied to the data for different models, so that their effect on the classification performance could be evaluated. In each case, a validation split was used of 0.2, meaning that 20% of the training data would be used for validation during training rather than for the training itself. The metrics achieved during training can be found in Appendix B. The code for the architecture is available in `models/second_model.py`.

As mentioned in section 2.4, data augmentation can be easily implemented in Tensorflow using Keras layers. Three layers were used: `layers.RandomFlip`, `layers.RandomRotation` and `layers.RandomZoom`. These layers flip the input with a random probability, rotate the input by a random amount (between 0 and 1π radians clockwise or anti-clockwise) and zoom the input in or out by a random amount between 0 and 100%. Bilinear interpolation is used, and any blank space is filled by reflecting the input about the edge. The data augmentation layers do not activate when the model is being used for classification.

The data augmentation layers were implemented in `utils/data_augmentation.py`

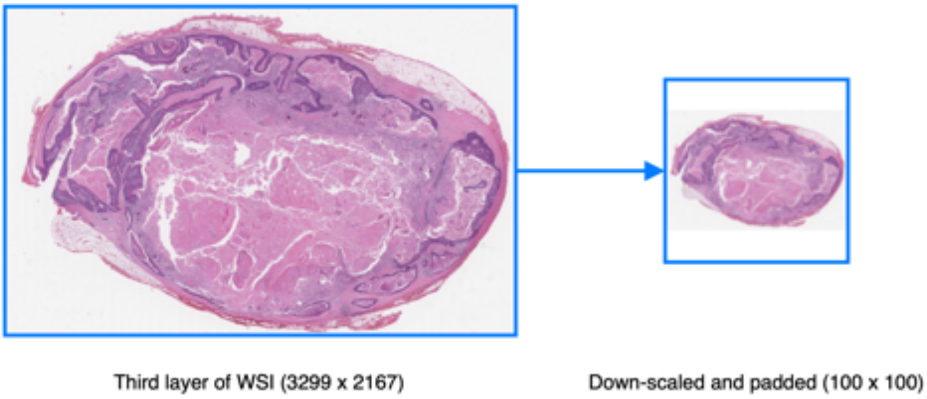
Third layer of WSI (3299 x 2167)
Down-scaled and padded (100 x 100)

Figure 4.1: The process for generating training data for the whole-slide approach.

so that the same augmentation could be used by multiple architectures, although in the end only one architecture was used. Keras offers additional augmentation such as `layers.RandomBrightness`, `layers.RandomContrast` and `layers.RandomTranslate`, which were not used in this project.

4.2 Image pipeline

As mentioned in chapter 3, loading the WSI into memory at the highest resolution is not feasible on the systems being used. Therefore, an image pipeline was implemented in `utils/image_pipeline.py`. This loads the third layer of the slide into memory. This layer is a down-sampled version of the first layer and can easily be loaded. Then it is down-sampled further to 100 × 100 px and saved as a PNG to be used as training data for a model. The aspect ratio is maintained, and the image is padded with white pixels to make it square, rather than being cropped. The image pipeline also has a function to normalise the stains in a given image. This was used to turn the datasets into training data for the whole-slide method.

4.3 Patch generators

Using patches at the full resolution / magnification (40×) would be too slow to be practical, as a single slide from the Camelyon16 dataset would yield 21,630,025,728 100 × 100 px patches, taking approximately 20 years for classification of all patches in a single slide. Since it was not practical to use the full resolution, it made sense to use one of the lower resolution layers in the image pyramid within each WSI file. $\frac{1}{16}$ of the highest resolution was chosen as a good balance between detail and time. In the Head and Neck 5000 slides, this is the second layer in the image pyramid, while in the Camelyon16 data set this was the fourth layer. For the Camelyon16 dataset, this now yields 8,418 100 × 100 px patches (minus any that are thresholded), with an approximate time of 4 minutes to classify them all.

From the Head and Neck 5000 dataset, patches were extracted in the following categories: positive tissue, negative tissue, and background. The first of these was much scarcer than the others, therefore, undersampling was employed, i.e., from each slide used for patch generation, all positive patches were extracted, whilst only some of the negative and background patches were extracted. During training, the negative tissue and background patches were combined into a single class called negative, with 50% of those patches taken from the negative tissue and 50% taken from the background. The patch training data extracted from the Head and Neck 5000 dataset consisted of 1,626 positive patches and 1,626 negative patches (813 taken from negative tissue and 813 taken from the background). When extracting normalised patches, however, these numbers decreased as some patches returned an error message. This is described in section 4.10. Therefore, when training with the normalised data from the Head and Neck 5000 dataset, there were 810 positive patches and 810 negative patches (405 from negative tissue and 405 from background).

When it came to extracting patch data from the Camelyon16 dataset, the annotations only showed the locations of lesions (positive tissue) within the slide, therefore it was not possible to discriminate between negative tissue and background to ensure that they both had adequate representation within the dataset. In an attempt to make sure that the negative patch dataset was balanced, the negative samples were taken starting from the middle of the slide. This accounts for the fact that the top left of a slide is almost guaranteed to be empty. The data extracted from the Camelyon16 dataset consitsed of 254 positive patches and 254 negative patches. When extracting normalised patches, the numbers remained the same.

As mentioned in section 3.8, the patch classifying process can be significantly sped up by ignoring background patches. This is done by checking the RGB values for every pixel in the patch and ignoring the patch if they are all above a certain threshold. Chen et al. (2021) used 220 for this value, but in this project that did not yield a good result, so it was lowered to 200.

To get training data from the slides for the patch-based method, two

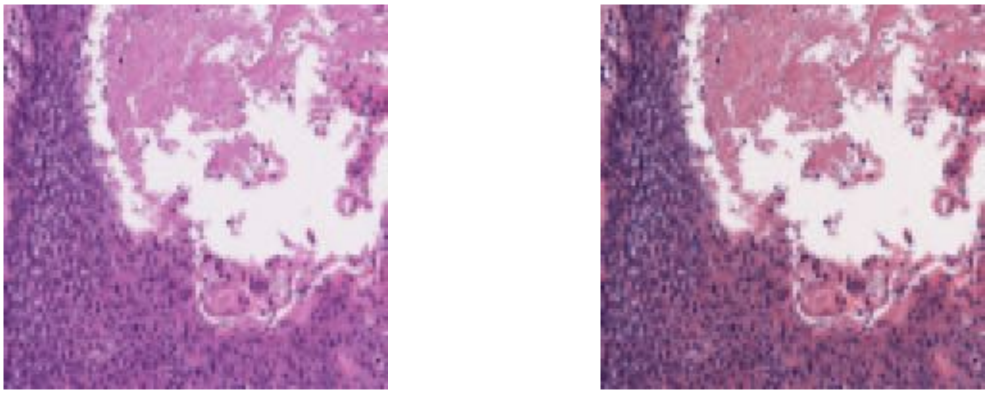

Figure 4.2: A 100 × 100 px patch from a slide before stain normalisation (left) and after (right).

programs were implemented. This is because the method for obtaining patches was different for the two datasets, as the Camleyon16 dataset did not have negative tissue annotated. The program for extracting patches from the Head and Neck 5000 dataset was `annotation_to_polygon.py`, and the program for extracting patches from the Camelyon16 dataset was `generate_patches_camelyon.py`. They both work in a similar way, loading the slide using OpenSlide and reading in the provided annotation in GeoJSON.

Since the annotations for the Camelyon16 dataset were provided in XML format, the program `xml_to_geojson.py` was written which converts these into GeoJSON annotations which can be used by the rest of the program. Then the program iterates over each patch of the slide and uses the Shapely library to check if the patch falls within a positive annotation polygon, and then saves the patch in the appropriate directory. The program has a binary constant which determines if the output should have the stains normalised or not.

4.4 Trainer

The program `trainer.py` creates a model based on the architecture, loads the training data provided into batches and trains the model for 1000 epochs, saving it to a specified checkpoint folder after each one. To control the use of data augmentation, the augmentation layer can be commented out in the model file.

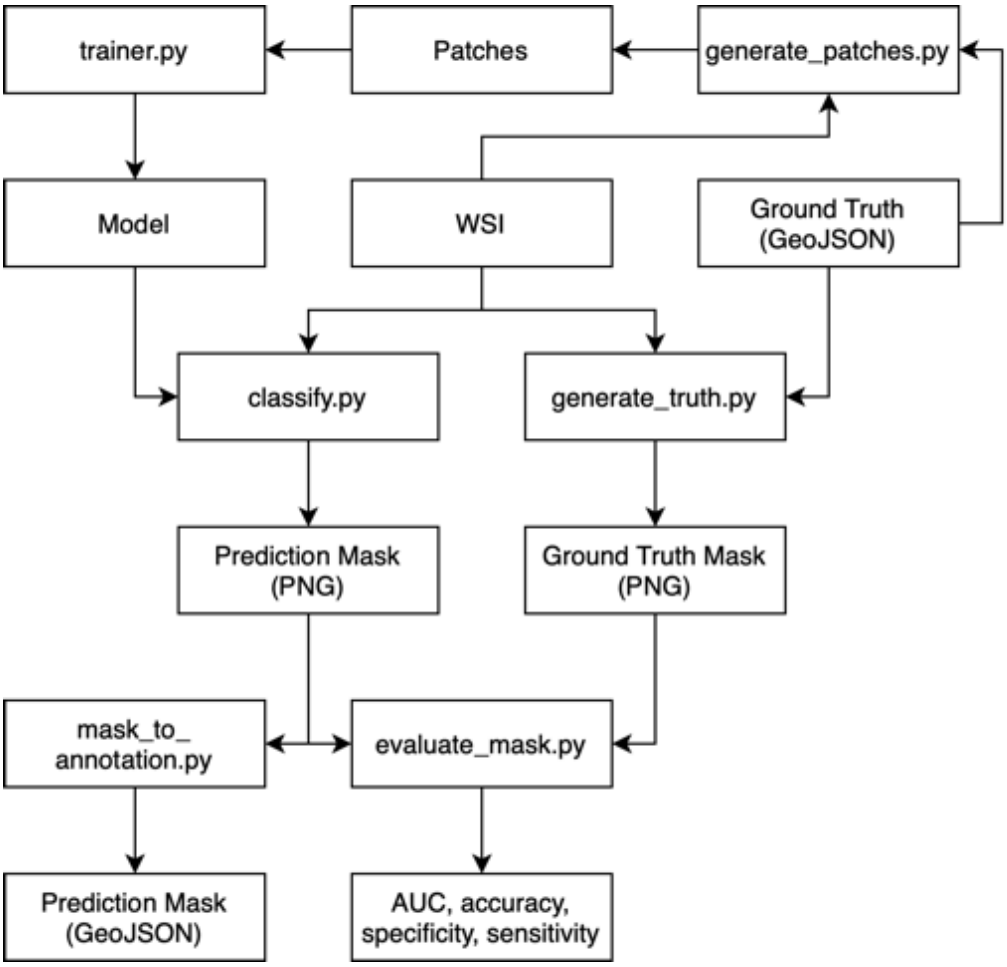

Figure 4.3: Diagram showing how the data sources interact with the programs implemented.

4.5 Classifier

The program `classify.py` takes as arguments an input file or directory, input size and Y/N for stain classification. It loads the model weights from a checkpoint specified by the config file, which also tells it if it should operate in patch-based mode or whole-slide mode. These two methods of operation are outlined below.

4.5.1 Whole-slide

When classifying a single WSI in whole-slide mode, the classifier will return a label (positive or negative) and a confidence for that label from 0 to 100%. When classifying a directory of WSIs classifies every file within that directory, creating as it does so a list of which is positive and which is negative, as well as a confidence score for each classification. The program separately tracks the correct label for each input, and at the end will provide the user with information for evaluation purposes, such as the overall accuracy, the sensitivity and the specificity. The confidence values are output which allow for the calculation of AUC as described in section 4.6.

4.5.2 Patch-based

For the patch based classification, the system outputs a classification mask PNG, such as that shown in Figure 4.4. Each pixel in this mask represents a 100 × 100 px patch in the slide at the specified level. The pixel's colour value is used to represent the confidence of the system, with the red value corresponding to negative and the green value corresponding to positive. Therefore a pixel with value (83, 17, 0) would correspond to a patch that the model classified as negative with 83% confidence. If the patch has been thresholded due to having no RGB values lower than 200, the patch is represented with a black pixel (0, 0, 0). This mask can be used to generate a GeoJSON file which can be imported as an annotation into QuPath. The ground truth annotation can also be turned into a mask PNG by the program `generate_truth.py`, and then the program `evaluate_mask.py` can use the pairs of masks to calculate the accuracy, sensitivity, specificity and AUC and output these to the console for the purposes of evaluation.

4.6 Evaluation

As mentioned above, some of the evaluation for the whole-slide method is handled by the classifier program. The exception to this is the AUC which is calculated by the program `calculate_auc.py`. This takes the confidences of a model for each slide, and the true value for each slide, and calculates the AUC based on those values using `tf.keras.metrics.AUC()`.

To evaluate the patch-based method more programs were required. The program `generate_truth.py` takes as input a WSI and a GeoJSON annotation file for that WSI, and produces a PNG mask where patches of 100 × 100 px of positive tissue are coloured (0, 100, 0) and patches that are negative are coloured (100, 0, 0). An example of this is given in Figure 4.4. The truth mask can then be given to the program `evaluate_mask.py` along with the confidence mask given by `classify.py` for the same slide, and it will return a reciever operating characteristic (ROC) graph, AUC, accuracy, specificity and sensitivity for the model on the input. It can also receive a list of slides to evaluate.

The program `flock_eval.py` takes the confidences for a lists of models on a list of slides, as well as the truth values for those slides or the patches within them (depending on which method is being evaluated). It returns a ROC graph, AUC, accuracy, specificity and sensitivity for the flock of models on the data provided.

4.7 Annotation generator

The mask PNG output by `classify.py` can be turned into a GeoJSON annotation file using the program `mask_to_annotation`. This draws a box around every patch that was classified as positive tissue. An example of this annotation can be seen in Figure 4.5, contrasted with a ground truth annotation.

4.8 Data availability

The dataset provided by the Head & Neck 5000 study is not publicly available, but contact details for the organisers of the study are available here: http://www.headandneck5000.org.uk/contact-us/. The Camelyon16 dataset which was used for evaluation in this project is available here: https://camelyon17.grand-challenge.org/Data/.

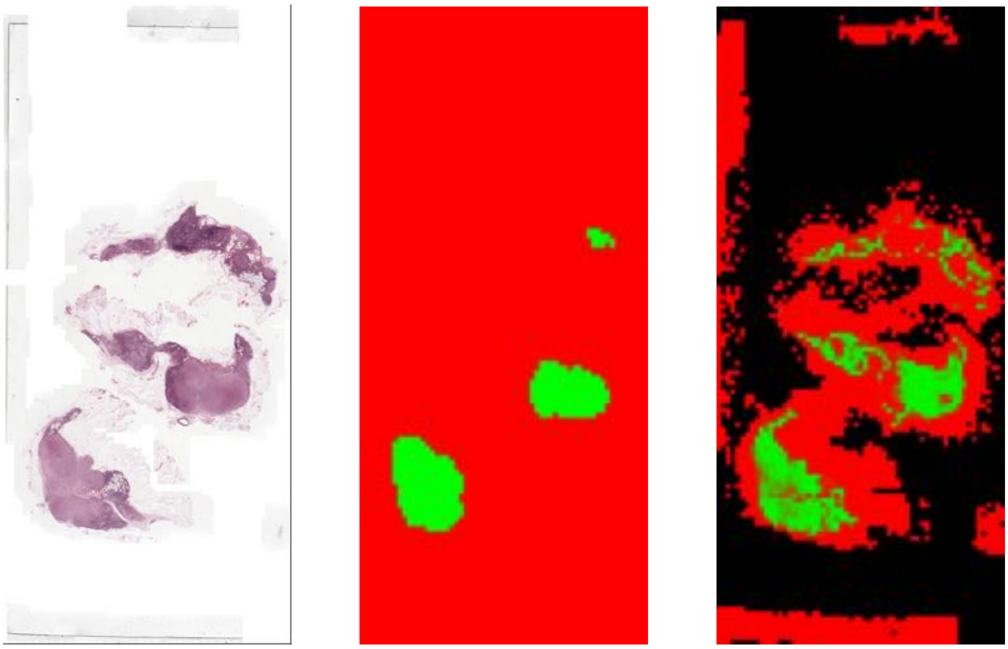

Figure 4.4: Left: whole slide image. Centre: binary ground truth mask generated from GeoJSON annotation file, showing three lesions in green. Right: confidence mask generated by patch classifier. The colours here are boosted to the range 0–255 to make them easier to distinguish.

4.9 Code availability

The source code for this project is available at the following Github repository: https://github.com/marleysudbury/detect-diagnose-tumours/. The trained models are not available. The stain normalisation code written by Github user schaugf is available separately at https://github.com/schaugf/HEnorm_python/blob/master/normalizeStaining.py.

4.10 Challenges

This section discusses challenges encountered during implementation which have not already been addressed above.

The first challenge that arose in the implementation was using the WSIs as training data. The first approach was to load the files directly into Tensorflow, as can be done with PNGs, JPEGs and other image formats. However, Tensorflow currently has limited and experimental support for TIFFs, and it is required to load the data elsewhere and pass it to Tensorflow

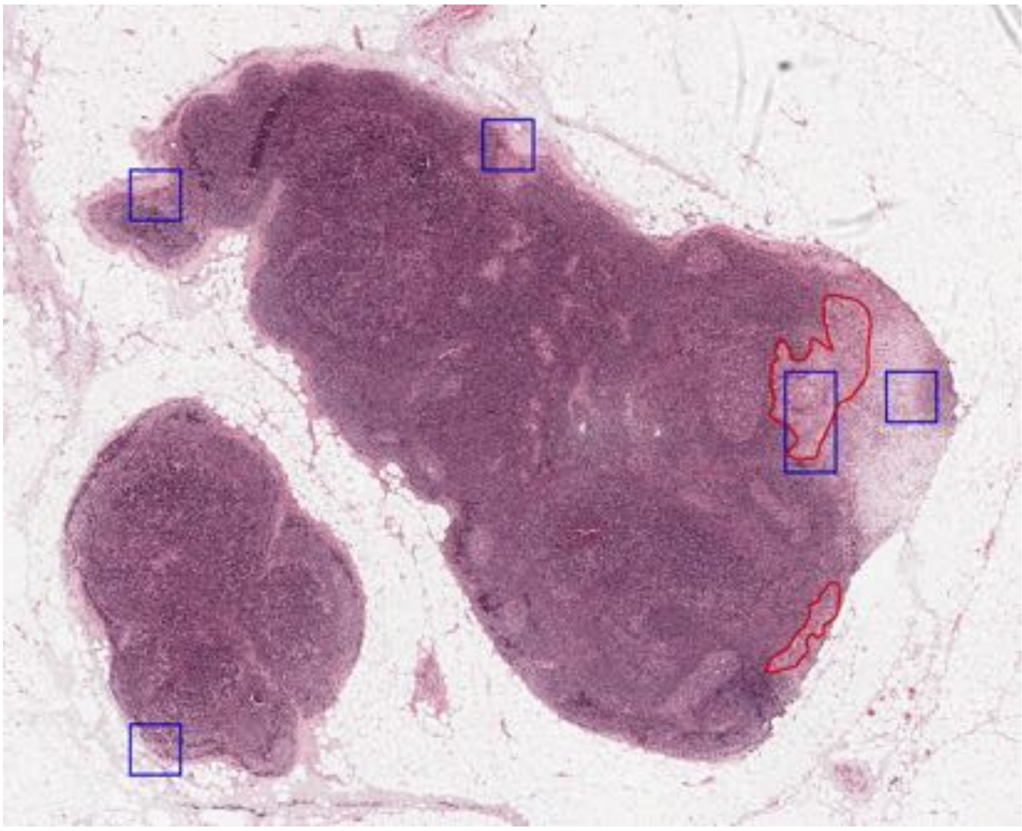

Figure 4.5: Using the annotation file generated by the program, it is possible to compare the patches it has classified as positive (blue) with the ground truth of which tissue is actually positive (red).

either as a Numpy array or as a PNG. To do this, the Pyvips library was used, as it supports multi-layer TIFFs, such as the WSIs.

When it came to implementing the patch-based approach, however, Pyvips was not capable. This is because it does not allow the loading of a patch from the WSI, and loading the whole image into memory and cropping it down was not feasible due to the size of the images. To resolve this challenge, the OpenSlide library was used, which was capable of extracting patches from the layer without loading the whole thing into memory.

Introducing the OpenSlide library brought another challenge—it is not compatible with the Pyvips library. This is because the Pyvips library includes some of the functionality from OpenSlide, meaning that the namespace contains clashes. This meant that the implementation for the patch-based approach had to be revisited and altered.

With the patch-based approach, during training the error ``Unknown image file format. One of JPEG, PNG, GIF, BMP required.'' was encountered, even though the input data was in PNG format. This was caused by a single image file that was corrupted in the training data. This file was located and deleted by iterating over all of the files and using the `verify()` method from the Python imaging library (PIL) (it could not be located manually due to the large number of files, and due to the file being hidden). Once the file was deleted, the model trained as expected.

There was a further error with the patch-based approach when it came to generating and classifying normalised patches. The normalisation software would return an error messages for some of the patches given to it. These error messages were `LinAlgError: Eigenvalues did not converge` and `RuntimeWarning: overflow encountered in exp`. As this occurred late in the project, and the implementation was done by others, this issue was unable to be resolved. In instances where the code is not able to normalise the patch, it is treated the same as if it was filtered by the RGB threshold. If that leads to positive patches being skipped over during classification, that is reported to the user during evaluation, and reported in chapter 5.

A challenge which came about as a result of the cross-evaluation between the two datasets was that they did not have the same layers. In the Camelyon16 dataset, as shown in Table 3.2, the layers go 40×, 20×, 10×, etc., whilst in the Head and Neck 5000 dataset (Table 3.3) there is no 20× layer or 5× layer. This meant that to access the same scale in both datasets, the index had to be manually chosen. This was only significant for the patch-based method, as the whole-slide method down sized the images. Alternatively, it would have been possible using OpenSlide to calculate the correct index for the desired scale, but this was not considered a good use of time since there were only two datasets. If there were additional datasets with varying scales, then it would be sensible to implement that.

5 Results and evaluation

During evaluation, the default classification threshold used to generate the accuracy, specificity and sensitivity is 0.5. When calculating the AUC, 100 thresholds are used from 0.0 to 1.0. If $\mathrm{Input}_\mathrm{n}$ is Yes, then the results are of the model classifying inputs which have had stain normalisation applied.

In this project, cross-evaluation is used to see to what extent the learning of the networks is transferable between datasets. Information on the datasets used can be found in section 2.3. Where a model is trained on the Head and Neck 5000 data, it is denoted as α, and where it is trained on the Camelyon16 data, it is denoted as β. Where the patch-based method was used rather than the whole-slide method, a $_p$ will be used. Where data augmentation was used in the training process (see section 2.4), the model name will have an $_a$. Where the training data has had stain normalisation applied, the model name will have a $_n$.

The AUC of each model is calculated by the `tf.keras.metrics.AUC()` method from Tensorflow, which recieves as arguments a list containing the confidences generated by the model for each image and a list of the correct values (for both lists 0 represents 100% confidence that the input is negative, and 1 represents 100% confidence that it is positive). The AUC refers to the area under the ROC, a curve which plots the true positive rate (TPR) against the FPR (1 - true negative rate (TNR)) at various confidence thresholds. An AUC of 0.5 would be equivalent to a model which decides the classification label by tossing a coin. According to Mandrekar (2010), an AUC of 0.7–0.8 is acceptable, 0.8–0.9 is excellent and more than 0.9 is outstanding. It is useful to look at the performance of an image classifier in terms of its AUC because it is threshold invariant. If a model is very good at classifying, but only when the threshold is something different from 0.5, such as 0.3, this will be reflected in a good AUC, whilst the model may give a bad accuracy, specificity or sensitivity. In these cases, an optimal threshold can be calculated using the ROC using for example the method described by Hong (2009), although this was not done as part of this project.

5.1 Whole slide approach

The results for all models using the whole-slide method are shown in Table 5.1. α models were trained on 103 slides from the Head and Neck 5000 dataset (60 positive and 43 negative). β models were trained on 100 slides from the Camelyon16 dataset (50 positive and 50 negative). They were evaluated each other's training data.

Model	**Input**$_n$	**AUC**	**Accuracy**	**Specificity**	**Sensitivity**
α	No	0.659	63%	70%	56%
α	Yes	0.645	63%	58%	68%
β	No	0.595	73%	33%	78%
β	Yes	0.521	52%	49%	54%
α_a	No	0.719	69%	92%	46%
α_a	Yes	0.710	64%	88%	40%
β_a	No	0.589	52%	57%	48%
β_a	Yes	0.439	42%	75%	18%
α_n	No	0.540	54%	100%	8%
α_n	Yes	0.540	54%	100%	8%
β_n	No	0.678	63%	16%	96%
β_n	Yes	0.606	62%	11%	98%
α_{an}	No	0.686	67%	88%	46%
α_{an}	Yes	0.727	71%	64%	78%
β_{an}	No	0.738	63%	13%	98%
β_{an}	Yes	0.695	64%	15%	98%

Table 5.1: Results for different models using the whole-slide approach.

5.1.1 Dataset comparison

α models on average scored an AUC of 0.65325, whilst β models scored an average of 0.607625. When the models are paired up so that each α is paired up with the β which has the same features, α outperforms β in terms of AUC in 5 out of 8 instances. Therefore it seems that the Head and Neck 5000 dataset leads to slightly better performance in this area, but the difference is not large.

Model	**AUC**	**Accuracy**	**Specificity**	**Sensitivity**
α	0.65325	63%	83%	44%
β	0.607625	59%	34%	74%

Table 5.2: Averages for the two datasets using the whole-slide approach.

5.1.2 Stain normalisation only

Table 5.3 shows that across all whole-slide method models, use of stain normalisation has a very minor effect on the accuracy, with the exception of when models are trained without stain normalisation but tested on data that has had stain normalisation applied. In that case, there is a 9% drop in accuracy compared to no normalisation at all. This suggests that there is a large difference between the ordinary datasets and their stain normalised counterparts, whilst there is not a large difference in the staining between the two datasets. These results do not match what was shown by Ciompi et al. (2017), who found that apply stain normalisation in testing and training gave an accuracy of 79.66% compared to 50.96% with no stain normalisation at all. Of course, the effect of stain normalisation will be dependent on the differences in staining between the datasets.

	Training with SN	Training without SN
Testing with SN	63%	55%
Testing without SN	62%	64%

Table 5.3: Average accuracy from training and testing with and without stain normalisation (SN).

	α_{an}	α_a
Testing with SN	71%	64%
Testing without SN	67%	69%

Table 5.4: α_a accuracy from training and testing with and without stain normalisation (SN).

5.1.3 Data augmentation only

In the models without data augmentation there is a generally poor performance, which can be attributed to overfitting to the training dataset. Across all models, adding data augmentation leads to an increase in AUC of 0.064875. However, this effect seems to be greater in α models, which had an increase of 0.1145, whilst β models only saw an increase of 0.01525. Since the cropped images from the Head and Neck 5000 dataset generally contain more information than those from the Camelyon16 dataset, the data augmentation process will be more effective for α models, as the augmentation methods used cannot add information that is missing, but can alter existing information.

5.1.4 Stain normalisation and data augmentation

The results so far suggest that stain normalisation has a negligible effect, whilst data augmentation has a large effect, particularly on α models. Models with both stain normalisation and data augmentation achieved an AUC improvement of 0.1205 compared to stain normalisation alone, and of 0.09725 compared to data augmentation alone. Therefore, it is clear that combining both of these techniques is key to making the best classifier.

5.1.5 Overall winner

The best performing model for the whole-slide approach in terms of AUC only, was β_{an}, with 0.738 on non normalised input data. However, this model has a specificity of 13% and a sensitivity of 98%, which shows that it has a large bias towards positive classification. It is therefore not a useful classifier, which is unsurprising because it exhibited signs of overfitting during training, where it had a validation accuracy of only 0.5500 (Table B.1). The next highest AUC was α_{an} with 0.727 on normalised input data. This model seems to be more well rounded, with a specificity of 64% and a sensitivity of 78%. This result reinforces the suggestion in chapter 3 that the patch-based training data generated from the Head and Neck 5000 dataset retains more useful information than that generated from the Camelyon16 dataset, as a result of the slides being cropped to include only the important information.

5.2 Patch-based approach

The results for all models using the patch-based method are shown in Table 5.5. The α models not trained on stain normalised data were trained on 3,252 patches (1,626 positive and 1,626 negative) generated from 10 slides in the Head and Neck 5000 dataset, whilst the α_n models were trained on 1,620 patches (810 positive and 810 negative) generated from the same

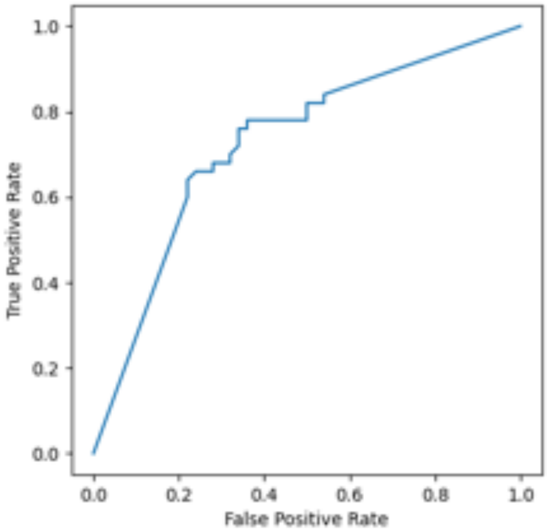

Figure 5.1: ROC of α_{an} on normalised data, with an AUC of 0.727.

slides. The β models were trained on 508 patches (254 positive and 254 negative) taken from 10 slides in the Camelyon16 dataset.

The patch-based approach can be evaluated by comparing the label given for each patch to the annotation provided with the dataset. As with the whole-slide approach, the confidence values given by the model can be used to calculate the AUC for the various patch-based models.

Since the purpose of this system is to identify metastases within lymph nodes, it is not relevant to evaluate the performance of it on differentiating negative lymph node tissue from background. Furthermore, it would not be possible to evaluate that performance as the Camelyon16 dataset only provides annotations for positive tissue (lesions). This means that a binary ROC will be used for this as well.

When evaluating this approach, cross-evaluation between the datasets is again employed. Each model is tested on every patch from 10 positive annotated slides from the other dataset. The evaluation metrics, such as accuracy, are given for the classification of patches. The system as it currently stands does not make a conclusion for a given slide based on its predictions for the patches within the slide.

As mentioned in chapter 3, X. Wang et al. (2021) employed an RGB threshold to filter out background patches when classifying. In this project, the threshold was set at 200. When testing the α_p models, this lead to 18

Model	Input$_n$	AUC	Accuracy	Specificity	Sensitivity
α_p	No	0.477	80%	80%	76%
α_p	Yes	0.633	50%	50%	74%
β_p	No	0.501	52%	99%	1%
β_p	Yes	0.356	44%	86%	7%
α_{pa}	No	0.392	92%	93%	2%
α_{pa}	Yes	0.456	79%	80%	4%
β_{pa}	No	0.532	56%	84%	26%
β_{pa}	Yes	0.534	49%	87%	15%
α_{pn}	No	0.500	2%	0%	100%
α_{pn}	Yes	0.544	6%	5%	100%
β_{pn}	No	0.504	52%	99%	2%
β_{pn}	Yes	0.496	47%	95%	4%
α_{pan}	No	0.617	90%	90%	67%
α_{pan}	Yes	0.890	79%	79%	75%
β_{pan}	No	0.683	71%	76%	65%
β_{pan}	Yes	0.469	60%	79%	43%

Table 5.5: Results for different models using the patch-based approach.

positive patches being ignored due to the RGB threshold, whilst 13 positive patches were ignored due to errors in the normalisation code. When testing the β_p models, the RGB filter lead to 53 positive patches being ignored, whilst 27 patches were ignored due to normalisation errors.

5.2.1 Dataset comparison

α_p models had an average AUC of 0.563625, whilst β_p models had an average AUC of 0.509375. This difference in performance can be attributed to the larger training dataset used for the α_p models. When the models are paired up so that each α_p is compared to the β_p with the same features, α_p outperforms β_p in 3 of the 8 instances.

5.2.2 Stain normalisation only

Unlike in subsection 5.1.2, using the patch-based method the results indicate that stain normalisation has a negative impact on the ability to classify. The average accuracy of patch-based models with stain normalisation (SN) on training and testing data was 48%, compared to 70% for models with no stain normalisation. Despite this trend, the best performing model (subsection 5.2.5) employed stain normalisation on both training and testing data. This suggests that while stain normalisation is an important element, it is only useful in a model which is good.

	Training with SN	Training without SN
Testing with SN	48%	56%
Testing without SN	54%	70%

Table 5.6: Average accuracy from training and testing with and without SN in the patch-based method.

5.2.3 Data augmentation only

Patch-based models without data augmentation achieved an average AUC of 0.501375, whilst those with data augmentation achieved an average AUC of 0.571625. β_p, α_{pn} and β_{pn} show particular signs of overfitting, although α_{pa} does as well, despite using data augmentation. The best performing model does use data augmentation (subsection 5.2.5).

5.2.4 Stain normalisation and data augmentation

Patch-based models with both stain normalisation and data augmentation achieved an average AUC increase of 0.18625 compared to stain normalisation alone, and of 0.15375 compared to data augmentation alone. They achieve an average AUC increase of 0.173 compared to models with neither stain normalisation or data augmentation.

5.2.5 Overall winner

With an AUC of 0.890 on normalised input data, α_{pan} far exceeds the performance of any other model created in this project. It beats the second highest AUC from a patch-based method (β_{pan} on non-normalised data) by 0.207, and it beats the best whole-slide method (α_{an} on normalised data) by 0.107.

5.3 Wisdom of crowds

This section is inspired by the work of Levenson et al. (2015), who showed not only that pigeons can be trained to classify pathology and radiology images of breast cancer, but that when voting as a flock they can achieve

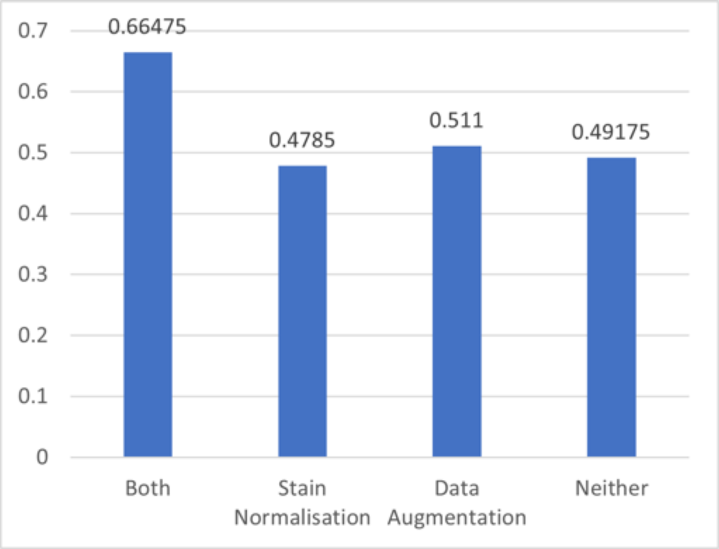

Figure 5.2: The average AUC of patch-based models by features.

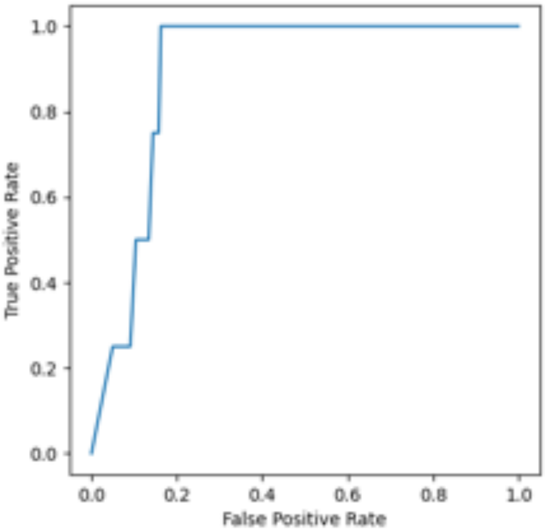

Figure 5.3: ROC of α_{pan} on normalised data, with an AUC of 0.890.

an AUC of 0.99, surpassing any of the individual pigeon's ability. Below are the results of applying this 'flock-sourcing' to the models created in this project.

The method for reaching the flock score is as follows: the sum of classifications from each model for each stimulus are taken, where positive is 1 and negative is 0. This is then divided by the number of models to give a flock confidence. A flock confidence of 0 would mean that all models said the stimulus was negative. A confidence of 1 would mean that they all said it was positive. The resulting number can be used to calculate accuracy, specificity, sensitivity and AUC as was done for the individual models. These values are given in Table 5.7.

Method	**AUC**	**Accuracy**	**Specificity**	**Sensitivity**
α	0.758	63%	92%	34%
α_p	0.725	88%	89%	35%
β	0.587	75%	36%	95%
β_p	0.649	52%	99%	1%

Table 5.7: Flock results for the two methods.

The flock achieves a better AUC than the individuals for α models. However, for α_p, β and β_p models there were individuals who performed better than the flock. The reason that this does not match the effect observed by Levenson et al. is that in that experiment all of the classifiers (pigeons) were on roughly equal footing, whereas here each flock contains classifiers that are deliberately lacking features, and therefore have been shown to be worse.

6 Future work

It is common in machine learning to think that more data leads to greater accuracy, and indeed many well performing solutions to this problem have used larger datasets than were used in this project. Therefore, future work should involve larger datasets. However, in the case of WSIs it must be remembered that the size of the data is very large. Therefore, the training and classification processes would need to be more efficient before using more data becomes practical.

The patch based approach took approximately 5.5 minutes to diagnose every patch of the slide. This process is parallelisable, and doing so would yield an improvement to the run time. Doing this would also make it more practical to use patches from a higher zoom level, which would likely increase the accuracy of the system. Additionally, if this process was sped up it could allow for a "sliding window" approach, which would give a label to each pixel in the input rather than a label to every 100 × 100 px patch.

As mentioned in chapter 2, research has been done on using unified memory techniques to train neural networks on the the whole slide at a much higher scale than was used in this project. Future work on classifying lymph nodes should certainly explore if it is possible to achieve good results with with this technique, since research by Chen et al. (2021) has shown it can be highly successful at classifying types of lung tumour.

The stain normalisation method proposed by Macenko et al., and described in section 2.5, has two important variables, α which determines the percentiles used to find the optimal colour vectors, and β which determines the minimum OD required for a pixel to be normalised. In this project, the default values of $\alpha = 1$ and $\beta = 0.15$ recommended in the paper were used, but it is possible that in future work better suited values for this task could be found.

A wide variety of deep learning architectures have been used to approach this problem and related problems with success, and it was initially planned to implement and evaluate several different architectures. However, due to the limited time available in this project, only one was implemented.

Future work should explore if better results can be achieved by using a different architecture, in particular by using ResNet-50 which was discussed in section 2.2 and was used by both X. Wang et al. and Chen et al.

Future work should also explore more methods of data augmentation than those that were used in this project. For example, Chen et al. (2021) used colour augmentation methods such as randomising contrast, brightness, hue and value

The weights in the neural network architectures used in this project were all initialised randomly, as is the default mode of operation in Tensorflow. This lead to some situations where networks didn't learn effectively, and the process needed to be restarted. This could be avoided by using a smart initialisation method, such as the method proposed by He et al. (2015), designed for use with ReLU activation.

During the evaluation for the patch-based method, only positive slides were used as data, since these have both positive and negative patches to classify. Future work should include experimentation to see if patch-based models are more or less effective at classifying negative slides.

In addition to those carried out in chapter 5, there are other test which could be carried out in the future. Firstly, the code produced was not thoroughly tested, and therefore bugs may exist within the system that are not handled and could result in crashes or incorrect results.

Secondly, evaluation of performance on a validation subset of the data from the training dataset was not carried out. Therefore, it is not possible to distinguish between a model which performed well on the training data but did not generalise to the other dataset, and one which did not perform well on any data at all. This means that we miss part of the whole picture in terms of the effect of the different processes used. To get some idea of this performance, please see the complete table of training results in Appendix B.

7 Conclusions

The primary aim of this project was to build a system which could identify metastasis in lymph nodes. To this end, it has been moderately successful, whilst not achieving the performance of some other existing solutions listed in section 2.2. Another aim was to identify the effect of data augmentation and stain normalisation on deep learning solutions to this problem, which was observed to be significant, but not as prominent on average as anticipated. A further aim was to evaluate the difference in performance of the whole-slide and patch-based methods, and it was found as expected that the patch-based method produced the best model as it had a lot more information to work with.

The fact that the α models largely outperform the β models suggests that either the Head and Neck 5000 dataset has higher variability, making it more generalisable, or that the Camleyon16 dataset was prone overfitting. Of these, the second is the most likely, particularly for the patch-based method. This is because the differences between the datasets are most apparent at the whole-slide level, where one is cropped and the other is not, and also because researchers such as X. Wang et al. (2021) have managed to achieve high transferability between tissue classifiers from one dataset to another using a patch-based approach.

The results of the whole-slide approach suggest that there are some differences between metastatic and normal lymph nodes when viewed at a scale of 100 × 100 px that a computer can pick up on, however these are only enough to achieve an AUC of 0.727, and therefore there is likely not enough information conveyed in such as small resolution for this to be usefully be applied. The principle of whole-slide classification is good in that an annotation-free approach to training a tissue classifier reduces the workload of pathologists who would have to otherwise painstakingly produce the annotations by hand, and it also allows for the possibility of the machine picking up on differences which may be missed by a human. Chen et al. (2021) showed that with a significantly higher resolution of input, and the necessary computational power, this principle can be applied with even higher accuracy than a patch-based approach.

A higher AUC was achieved using the patch-based method, of 0.890, however this was only the case with one particular model: α_{pan}. This model employed both data augmentation and stain normalisation, and compared to β_{pan} it had a significantly higher amount of training data. As can be seen by comparing this result to Table 2.1 in chapter 2, α_{pan} achieved 0.08 more AUC than the mean of 11 pathologists with a time constraint reported by Ehteshami Bejnordi et al. (2017). It also had 12.2% higher sensitivity, although it had 19.5% less specificity. This shows that the model is comparable with the performance of people under time pressure, however it comes nowhere close to the 0.966 AUC of a pathologist without time constraint reported in the same paper. The model also took a longer amount of time than the pathologists, who took an average of 120 minutes to diagnose 129 slides (1.075 minutes per slide), whilst α_{pan} took 55 minutes to diagnose 10 slides (5.5 minutes per slide).

The fact that the patch-based methods in general did not perform better than the whole-slide methods, indicates that either there was insufficient training data used to capture the variation that occurs in unseen data, or that too much data was being lost due to the down-scaled resolution. The data used in the patch-based methods of this project was taken from a low-resolution layer of the slide, where the width and height are both $\frac{1}{16}$ of their full size. Some investigation was carried out into taking the data at a higher resolution, where the width and height are $\frac{1}{4}$ of their original size, which can be found in Appendix A. An AUC of 0.806 was achieved by one model, with the rest performing slightly above or below 0.5.

It is clear from the results obtained in this project that stain normalisation does indeed have an effect. The highest performing models were where the training and testing data both had stain normalisation applied. On average across all models, those with no stain normalisation at all scored an AUC of 0.558, whilst those with stain normalisation on either the training data or the testing data, but not both, scored an AUC of 0.5775, and those with stain normalisation applied to both had an AUC of 0.620875.

As expected, data augmentation lead to more robust systems. This is because the models that did not have data augmentation during training were overfitting to the training dataset, and were exposed to less variation. On average across all models, those without data augmentation scored an AUC of 0.549688, whilst those with data augmentation scored an AUC of 0.61725.

8 Reflection

Coming into this project, I had made the naïve assumption that using a small input size, a neural network with enough training would still be able to discriminate between positive and negative tissue samples. This assumption came from having implemented an image classifier which was capable of determining the species of a flower from only a 100 × 100 px image, as well as a lack of understanding of the diagnosis process, and a lack of exposure to the slides. However, this assumption is completely inappropriate as there are far fewer low-frequency distinguishing characteristics between metastatic and non-metastatic lymph node tissue samples than between different flowers. This fundamental misunderstanding meant that much of the early work on this project was spent exploring avenues which did not lead anywhere, although as shown in chapter 5 there is some capability to distinguish at low resolutions, albeit at a less than acceptable rate of success. I now understand that there is not a one-size fits all approach to image classification, and that a thorough understanding of what the data is and what parts of it are important is required before setting out to design a system which can tackle the problem.

My initial planning for this project proved to have been too optimistic in terms of the timescale. This is because I underestimated the challenge that would be posed by the size of the WSIs (see chapter 4) and by the evaluation. As a result of this, I had to construct additional features that I did not originally plan. I also found that being limited to lab opening times to carry out model training made it difficult to manage my time effectively at some points of the project.

I have learned a lot about machine learning, which I had studied during my time at university but never used before. I also gained a lot of knowledge about processes, techniques and tools used in pathology. Another thing that I have developed during this project is the ability to understand and explore a topic from scratch independently.

If I were to do this project again, I would approach it differently as a result of the knowledge I have gained. I would take more time at the start to understand the solutions put together by others for similar problems, and to carefully plan out my approach to creating a solution, rather than running straight into it and getting lost amongst the practical challenges outlined elsewhere in this report. I would also ensure that I was more organised with the different models created and evaluations carried out, as towards the end of the project I had to retrain and reevaluate several models due to inconsistencies in the training and evaluations carried out, which added to the pressure of time. Another thing which added to the time pressure was the lack of access to the labs in the evenings and weekends. To avoid this, cloud computing would be the best option, although that may not have been possible in this particular case due to the requirement to keep the Head & Neck 5000 dataset on a password encrypted hard drive.

The timeline of work completed on this project ended up looking quite significantly different from the timeline I set out in my initial plan. The only thing which wasn't completed from that initial plan was "Implement non deep learning approach to tumour detection and diagnosis", which was not done as there wasn't enough time to explore this. There was also some work done which was not included in the initial plan, as at the time I did not know it would be required.

Although notes were taken throughout the project, the report writing only started towards the end, and later than planned in the initial plan. If I were to do this project again, I would start writing the report earlier, as in the process of writing and putting everything together I found that a lot of good ideas occurred to me which would have been more useful if they had happened earlier in the project. By starting the writing process earlier, it would also avoid it having been so rushed, which in turn would have likely meant a better end product.

I really enjoyed this project, and I hope to utilise the skills and knowledge developed throughout it to do more work in the crossover area of heath care and computer science.

Table of abbreviations

AUC area under the receiver operating characteristic curve

CNN convolutional neural network

FNR false negative rate

FPR false positive rate

GAP global average pooling

GMP global max pooling

H&E hematoxylin and eosin

MIL multiple instance learning

OD optical density

PIL Python imaging library

ReLU rectified linear unit

ROC reciever operating characteristic

SN stain normalisation

SVD single value decomposition

TNR true negative rate

TPR true positive rate

WSI whole slide image

A Higher resolution patches

There was not time in this project to fully explore or evaluate the patch-based method at higher resolutions, where it is likely to perform better. This is because using a higher resolution means there are more patches to be classified in each slide, taking longer to evaluate. To give some hint of the potential that this way lies, some models were trained on Camelyon16 patches from layer 2 (1:4). These are denoted as $\beta_{p\frac{1}{4}}$. These models were trained on 10,320 patches (5,160 positive and 5,160 negative) taken from 10 slides in the Camelyon16 dataset. They were evaluated on 45,499 patches (24,405 positive, 21,094 negative) from 1 slide in the Head and Neck 5000 dataset. The results are shown in Table A.1. The results may not be reflective of a comprehensive performance, as the slide used for evaluation had an unusually high tumour to metastatic lymph node ratio, with a very small amount of normal tissue.

Method	**Input$_\mathbf{n}$**	**AUC**	**Accuracy**	**Specificity**	**Sensitivity**
$\beta_{p\frac{1}{4}}$	No	0.476	29%	89%	5%
$\beta_{p\frac{1}{4}}$	Yes	0.459	39%	49%	36%
$\beta_{p\frac{1}{4}a}$	No	0.806	78%	73%	81%
$\beta_{p\frac{1}{4}a}$	Yes	0.507	49%	69%	42%
$\beta_{p\frac{1}{4}n}$	No	0.521	34%	96%	9%
$\beta_{p\frac{1}{4}n}$	Yes	0.569	40%	96%	21%
$\beta_{p\frac{1}{4}an}$	No	0.583	49%	95%	31%
$\beta_{p\frac{1}{4}an}$	Yes	0.501	26%	99%	0%

Table A.1: Results of partially tested high-res patch-based method.

The model $\beta_{p\frac{1}{4}a}$ achieved a good AUC of 0.806 on non-normalised input data, however the rest of the models performed poorly despite having high accuracy on the training and validation data (Table B.2). These results

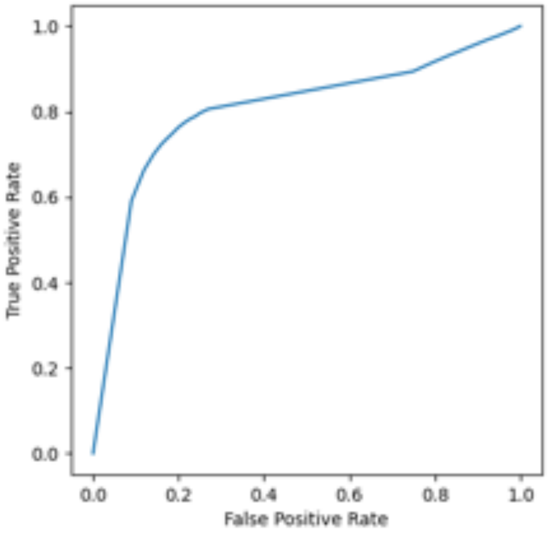

Figure A.1: ROC of $\beta_{p\frac{1}{4}a}$ on non-normalised input. AUC of 0.806.

could perhaps be improved by increasing the input size and increasing the number of slides used to generate the training data.

B Loss and accuracy of models on training and validation data

Model	**Loss**	**Accuracy**	**Validation loss**	**Validation accuracy**
α	5.7450e-08	1.0000	2.1773	0.8000
α_p	0.0030	0.9996	1.1914	0.8723
α_a	0.2271	0.8675	0.2899	0.8500
α_n	2.7289e-08	1.0000	1.3923	0.8000
α_{pa}	0.2757	0.8939	0.2861	0.9015
α_{pn}	0.6840	0.5139	0.6846	0.5093
α_{an}	0.1988	0.9277	0.6965	0.9500
α_{pan}	0.2214	0.9151	0.4364	0.8519
β	6.3628e-07	1.0000	0.5715	0.9500
β_p	2.5189e-07	1.0000	2.3103e-05	1.0000
β_a	0.4038	0.8125	1.0139	0.6000
β_n	5.0664e-08	1.0000	5.3902	0.5500
β_{pa}	0.0141	0.9975	0.0108	1.0000
β_{pn}	8.5179e-06	1.0000	3.5844e-05	1.0000
β_{an}	0.2925	0.8875	2.3814	0.5500
β_{pan}	0.0770	0.9803	0.0584	0.9802

Table B.1: Training and validation metrics for each model from the final epoch.

Model	**Loss**	**Accuracy**	**Validation loss**	**Validation accuracy**
$\beta_{p\frac{1}{4}}$	3.4059e-05	1.0000	2.2133e-05	1.0000
$\beta_{p\frac{1}{4}a}$	0.0704	0.9746	0.1315	0.9540
$\beta_{p\frac{1}{4}n}$	1.5171e-05	1.0000	0.0010	0.9995
$\beta_{p\frac{1}{4}an}$	0.0526	0.9821	0.2796	0.9104

Table B.2: Training and validation metrics for the models referenced in Appendix A, from the final epoch.

References

Babenko, B. 2008. *Multiple Instance Learning: Algorithms and Applications*. Available at: http://vision.ucsd.edu/~bbabenko/data/bbabenko_re.pdf [Accessed: 17 May 2022].

Cancer Research UK. 2020. *How cancer can spread*. Available at: https://www.cancerresearchuk.org/about-cancer/what-is-cancer/how-cancer-can-spread [Accessed: 25 April 2022].

Cancer Research UK. *Head and neck cancers statistics*. Available at: https://www.cancerresearchuk.org/health-professional/cancer-statistics/statistics-by-cancer-type/head-and-neck-cancers#heading-One [Accessed: 6 May 2022].

Chen, C.-L. et al. 2021. An annotation-free whole-slide training approach to pathological classification of lung cancer types using deep learning. *Nature Communications* 12(1193), doi: 10.1038/s41467-021-21467-y.

Ciompi, F. et al. 2017. The importance of stain normalization in colorectal tissue classification with convolutional networks. 2017 IEEE 14th International Symposium on Biomedical Imaging (ISBI 2017).

Ehteshami Bejnordi, B. et al. 2017. Diagnostic Assessment of Deep Learning Algorithms for Detection of Lymph Node Metastases in Women With Breast Cancer. *JAMA* 318, pp. 2199–2210. doi: 10.1001/jama.2017.14585.

Feldman, A. and Wolfe, D. 2014. chap. Tissue Processing and Hematoxylin and Eosin Staining. In: Day, C. eds. *Histopathology*. New York, NY: Humana Press. pp. 31–43.

Folmsbee, J., Johnson, S., Liu, X., Brandwein-Weber, M. and Doyle, S. 2019. Fragile neural networks: the importance of image standardization for deep learning in digital pathology. Medical Imaging 2019: Digital Pathology. Available at: https://doi.org/10.1117/12.2512992.

He, K., Zhang, X., Ren, S. and Sun, J. 2015. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification.

2015 IEEE International Conference on Computer Vision (ICCV), pp. 1026–1034. doi: 10.48550/ARXIV.1502.01852.

Hong, C. S. 2009. Optimal Threshold from ROC and CAP Curves. *Communications in Statistics - Simulation and Computation* 38, pp. 2060–2072. doi: 10.1080/03610910903243703.

Levenson, R. M., Krupinski, E. A., Navarro, V. M. and Wasserman, E. A. 2015. Pigeons (Columba livia) as Trainable Observers of Pathology and Radiology Breast Cancer Images. *PLOS ONE* 10, pp. 1–21. doi: 10.1371/journal.pone.0141357.

Macenko, M. et al. 2009. A method for normalizing histology slides for quantitative analysis. *2009 IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, pp. 1107–1110.

Mandrekar, J. N. 2010. Receiver Operating Characteristic Curve in Diagnostic Test Assessment. *Journal of Thoracic Oncology* 5, pp. 1315–1316. doi: 10.1097/JTO.0b013e3181ec173d.

NHS. 2019. *Mouth cancer diagnosis*. Available at: https://www.nhs.uk/conditions/mouth-cancer/diagnosis/ [Accessed: 25 April 2022].

Simonyan, K. and Zisserman, A. 2014. Very Deep Convolutional Networks for Large-Scale Image Recognition., doi: 10.48550/ARXIV.1409.1556.

Szegedy, C. et al. (2014). *Going Deeper with Convolutions*. doi: 10.48550/ARXIV.1409.4842. Available at: https://arxiv.org/abs/1409.4842.

Vahadane, A. et al. 2016. Structure-Preserving Color Normalization and Sparse Stain Separation for Histological Images. *IEEE Transactions on Medical Imaging* 35, pp. 1962–1971. doi: 10.1109/TMI.2016.2529665.

Vestjens, J. H. M. J. et al. 2012. Relevant impact of central pathology review on nodal classification in individual breast cancer patients. *Annals of Oncology* 23(10), doi: 10.1093/annonc/mds072.

Veta, M. and Schau, G. F. 2020. *Staining Unmixing and Normalization in Python*. Available at: https://github.com/schaugf/HEnorm_python [Accessed: 11 May 2022].

Wang, J. and Perez, L. 2017. The Effectiveness of Data Augmentation in Image Classification using Deep Learning., doi: 10.48550/ARXIV.1712.04621.

Wang, X. et al. 2021. Predicting gastric cancer outcome from resected lymph node histopathology images using deep learning. *Nature Communications* 12(1637), doi: 10.1038/s41467-021-21674-7.