Differing Machine Learning Architectures for Histopathological Cancer Detection

ST311 Project Submission

Candidates: 28551 and 33349

Department of Statistics

London School of Economics

Abstract

The following report employs 4 deep neural network architectures to identify cancerous tissue in histopathological scans. We first begin by conducting an EDA of the tissues scans and recognising the computational limitations posed to us during training - mainly the large dimensions of the training data.

We proceeded to define a training function that employs a number of techniques to reduce training time while still maintaining high train and test accuracy. Mainly, we employed mini-batch stochastic gradient descent and an exponential decay learning rate scheduler. We implemented the TensorBoard class to allow for in-depth visualisation and diagnostics of our results

We trained a basic CNN, an interpretation of AlexNet for the classification problem, a pretrained ResNet18 model to demonstrate transfer learning and VGG11 to employ the latest in deep image detection networks. After training the functions, we found transfer learning to be innapropriate for the task. Furthermore, VGG11 handled the training data with the highest train accuracy, however, AlexNet proved most accurate for out of distribution test data.

To conclude the report, we proposed the implementation of the AlexNet model with the sigmoid activation for binary classification. We recognise the computational feasibility of AlexNet compared to other models and proposed further implementations within the clinical setting for other detection problems.

Contents

1 Motivation

Lymphatic node cancer is a common stage of metastatic breast cancer and, if unsuccessfully treated, can lead to stage 4 metastatic cancer. The early detection of cancerous cells in the lymphatic system can stop metastasising into the blood stream, lungs, spinal system or other more sensitive organs.

The lymphatic system is often effected by breast cancer - the most common type of intrusive female cancer. Currently, histopathological images (tissue examined under the microscope) are analysed by a radiologist or oncologist. Areas of unusual density are outlined and a biopsy is performed to assess if the areas is currently affected by cancer. Computer vision and the use of deep neural networks is gaining traction as to expedite this identification process and get rid of human error in the process.

2 Dataset

The 'Histopathologic Cancer Detection' dateset contains 220,025 labeled observations of histopathologic scans of lymph node sections (Kaggle, 2022). These images are labelled as either exhibiting metastatic cancer tissue or clear. A positive label indicates the presence of cancer, precisely that the center 32x32px region of a patch contains at least one pixel of tumor tissue - the other areas of the image are ignored. Tumor tissue in the outer region does not influence the label. This outer region is provided to allow fully-convolutional models that to use padding or stride lengths that would otherwise miss the edge of the images.

Furthermore, the dataset includes 22,003 of labelled test images as well.

3 Proposed Architectures

- **Basic Convolutional Neural Network** A simple CNN implementation employing batch normalisation, max pooling and ReLu to solve the binary classification problem.

- **AlexNet** as a more complicated architecture to achieve more accurate predictions and mitigate overfitting. A slight modification will be made from the original LeCunn architecture to classify binary output rather than 10 classes.

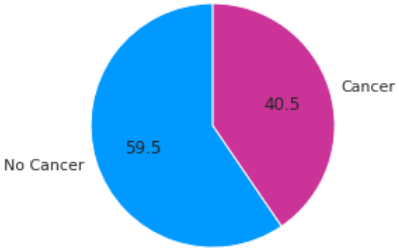
- **RestNet**as an architecture which allows training a large number of layers thereby increasing model complexity without the problem of vanishing gradient which fails to update weights during training). Additionally, it will allow us to train more layers without increasing the training error rate. Considering the large amount of parameters, ResNet will employ pretrained weights and biases.

- **VGG11** architecture as a deeper CNN introducing some non-linearities to better perform pattern detection. VGG11 will be the most computationally heavy model due to the large number of convolutions.

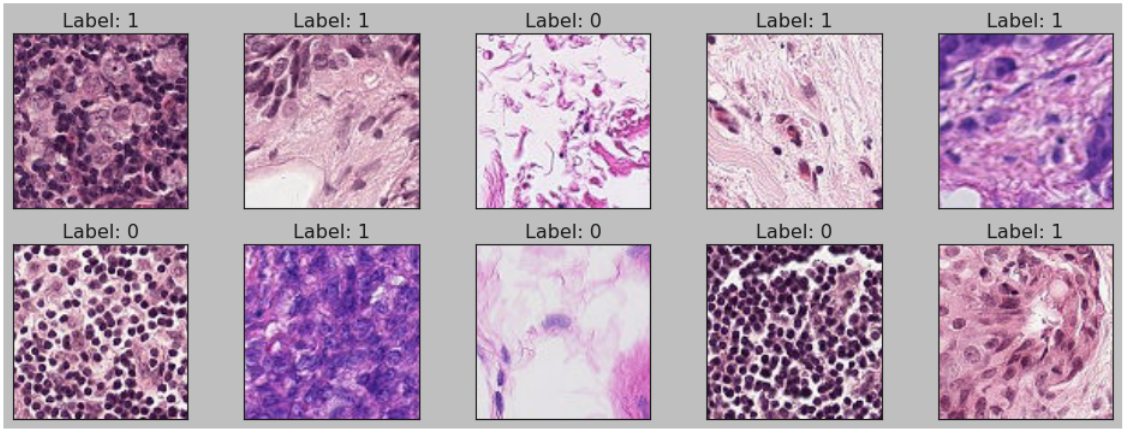
4 Exploratory Data Analysis and Augmentation

Prior to any model fitting, we need to determine the structure of the data alongside any cleaning required. After importing the files and labels, we run a quick count to check the balance of the dataset.

4.1 Data Characteristics



As seen above, approximately 40% of the dataset includes images labelled cancer. While high, we don't feel the need to re-balance the dataset for two main reasons. Firstly, cancer is the second leading cause of death world wide (WHO, 2016) and so we don't feel as if over-representation is of concern. Secondly, considering the clinical setting, tissue is usually only screened if there is suspected cancer and as a result, we would see higher percentages of detection. Inspecting the raw data, we have 3 channel (RGB) images sized 227 x 277 pixels. Below is a sample of what the data would look like, 'Label 1' indicating the positive detection and 'Label 0' otherwise.



4.2 Data Reshaping

We choose not to reshape the data at this time. Each architecture requires a different input size. While possible to change the structure, to allow for authentic comparison, we instead

choose to reshape the data prior to any training to align with the input size.

The reshape function completes 4 main tasks. Firstly, it scales the data to the input size specified as a parameter. Secondly, it applies random flip, rotations and translations to the images. We want to provide a realistic clinical setting and often radiography images will have these minor imperfections. Thirdly, it divides the data into batch sizes of 100. This batch size choice is to be discussed later. Finally, it loads 10,000 entries into a train dataloader and another 5,000 unique entries into a test dataloader. Although the dataset includes over 200,000 entries, computational restrictions made training impossible. This combination of batch size and training datset size records the same level of loss, train accuracy and test accuracy within 2 decimal points as the entire dataset with significantly less computation time.

5 Training Approach

5.1 Training Function Design

Our training function has a similar approach to the standard neural network trainer in most PyTorch models. First a forward pass is constructed, followed by the calculation of gradients then backpropogation to adjust the weights.

We chose to employ stochastic gradient descent. We consider this gradient descent algorithm to be both computationally efficient and accurate. There are more advanced algorithms, however, we recognise the large size of the dataset and the limited resources available to us.

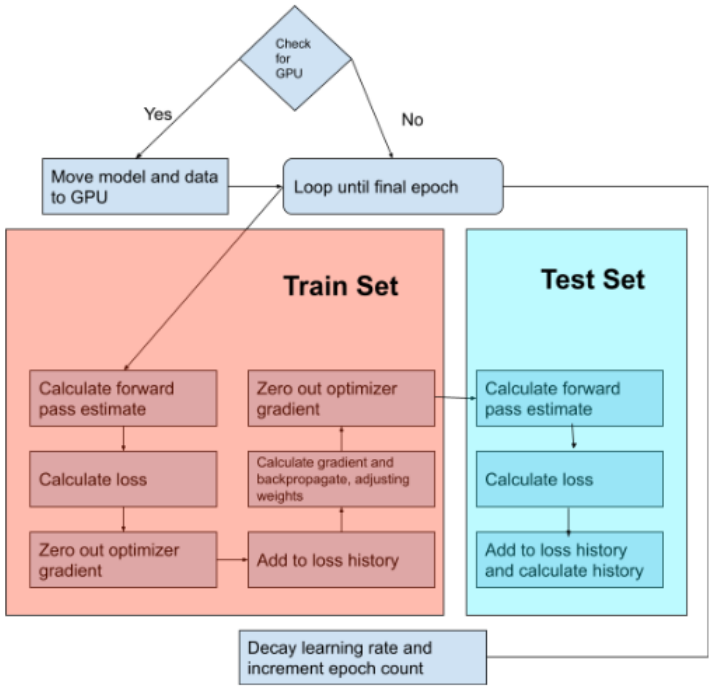
Secondly, we chose to implement exponential learning rate decay. While we are worried about training time, we still hope to converge on accurate results. For this reason, we

deemed it necessary to include a learning rate scheduler. After each epoch, the learning rate decreases exponentially at a rate γ defined as a hyper parameter (discussed further below). At epoch i, the learning rate l_i is given by the formula:

$$l_i = \alpha_0 \times e^{\gamma \times i}$$

Where α_0 is the initial learning rate discussed below.

Finally, after each iteration we log the loss, train accuracy and test accuracy using TensorBoard. The TensorBoard visualtion tool lets us accurately inspect the progress of our models and saves after each run. This way, we were able to tune our parameters without doing computationally costly cross validation. During the training function, the current loss and accuracy metrics are added to a scalar diagram, incrementing with each epoch.



5.2 Choice of hyperparameters

There are a number of hyperparameters we require for our model:

- **Batch Size**: Due to the large number of images, we clearly need to work with batches. In order to increase the speed at which training can take place, we chose a batch size of 100. We commonly see mini-batch stochastic gradient descent with batch sizes of 128 (Aremu, 2022), so choosing a batch size of 100 shouldn't reduce our training accuracy and help us achieve faster training times.
- **Epoch Number**: We chose to train over 10 epochs. Ideally, this would be larger. The more complicated models (VGG in particular) can take close to 30 minutes of training time per epoch. In choosing a relatively low epoch, it becomes important to implement lower learning rates - as to increase accuracy.
- **Initial learning Rate** α_0: Considering the choice for 10 epochs, we start the initial learning rate at 0.001 for all models. This value is already small by most industry metrics. However, with the further addition of learning rate decay, we can approach even higher levels of accuracy given we are working with batches over such low epoch counts.
- **Learning Rate Decay** γ: We chose an industry standard gamma decay rate of 0.96. With a high decay rate, we see smaller learning rates after the initial epochs.

6 Metrics for Analysis

To interpret the performance of the models, we use:

- Training accuracy
- Test accuracy
- Computational time

The training accuracy measures the amount of correct classifications made on the total number of training data points. Although indicative of the goodness of fit, the training accuracy is not a reliable metric of the model performance as a high accuracy may be signaling that the model is overfitting on the training set, and therefore poorly generalises on unseen data. The performance on unseen data is more pertinent to the clinical setting.

To get a better idea of how the model performs, we refer to the test accuracy which measures the number of correct classifications made by the model on the test data points. Since the test data points are not used to train the model, the test accuracy is more reflective of the external validity of the model - that is, how the model adapts to unseen data. A significant difference between the training and test accuracy gives strong evidence of overfitting, in which case, the model must be regularised (using regularisation techniques such as L1 and L2 regularisation in the loss function, higher dropout, early stopping ie. stopping the training process once the performance on the validation set gets worse).

Lastly, we compare the models based on computational time, that is the time every model takes to train. Although secondary to the metrics on predictive accuracy, the computational time can be an important factor to take into consideration in practical applications. In this project, we have trained our models using the GPU on a Google Colab notebook without which the training would have been significantly slower giving the size of the dataset and the nature of our task (convolutions rely on a lot computations and image classification models involve a large number of parameters to learn).

7 Architectures

7.1 Convolutional Neural Network

At first, a naive CNN architecture was trained as a baseline model for our analysis. The simplified structure of the architecture provides us with a starting point against which the performance of more complicated models will eventually be compared.

Our basic CNN consists of 5 blocks, each consisting of a convolutional layer, a batch normalisation layer, a ReLU function and finally, a maximum-pooling layer preceding the fully connected MLP giving the output. Since our project involves a binary classification task (either detecting cancer or not in 32x32 crops of digital pathology scans), we chose to include a Sigmoid activation function in the output layer for a greater stability of results, ensuring that the output always lies between 0 and 1. In particular, having the output lie in that range will ensure that loss is not dramatically large due to the output values being significantly different to the data labels (0 or 1). In turn, we thereby ensure that the accuracy and the losses are reflective of the true model performance.

With regards to the blocks themselves, each consists of a 2D convolutional layer which extracts features from the input image using a 32 x 3 dimensional filters with stride and padding equal to 1. The filters operate by sliding through the input image, eventually giving us the convolved feature maps. As the name suggests, a feature map consists of an ensemble of features that the network has learned through the first layer in a way that preserves the spatial relationship between pixels (this is important to ensure that further convolutions can be applied while image integrity is preserved).

Following the convolution, comes a batch normalisation process that serves the purpose of making the feature maps more uniform. In essence, normalising allows each feature map

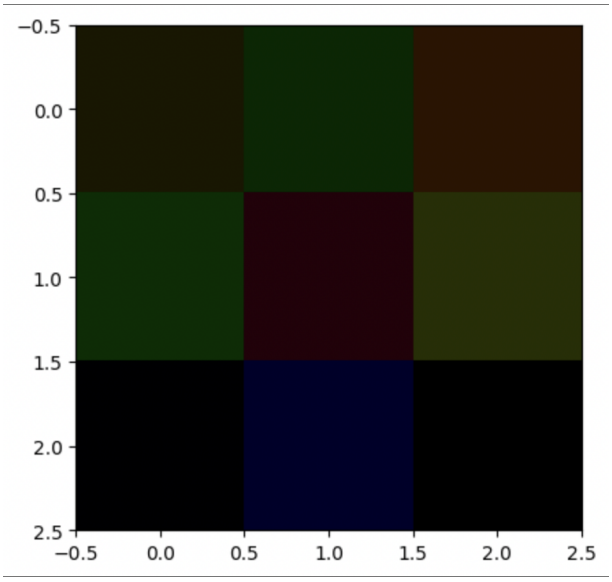


Figure 1: Image of a filter from the first layer of our CNN architecrure

to be standardised (with a mean of 0 and a standard deviation of 1), so that each feature learned by every feature map will be treated the same way regardless of its location in the image. Normalising batches inside blocks of a CNN appears to be a very popular practice in the deep learning community: Kaiming He, et al. in their 2015 paper titled "Deep Residual Learning for Image Recognition" cited: *"We adopt batch normalization (BN) right after each convolution and before activation"*.

After our feature maps have been normalised, we apply a ReLU function (Rectified Linear Unit) to introduce some non-linearities in the data. The convolution operations done precedingly rely on linear operations (such as matrix multiplication), and so using ReLU allows to regain non-linear relationships which better fit the structure of images (resulting in a more adapted and less biased model). Finally, our block concludes with a max pooling layer which reduces the size of the input representation according the following formula (for either height or width):

$$output = \frac{inputsize - poolingsize + 2 \times padding}{stride} + 1$$

l = length of the feature map,

w = width of the feature map,

f = dimensions of the filter,

c = number of channels of the feature map,

s = stride

In our case, the max pooling layer has a 2x2 window and stride 1. It downsamples the input along its spatial dimensions (height and width) by taking the maximum value over the 2x2 input window for each channel of the input. Apart from substantially reducing the number of parameters in the learning process - which allows for faster training - pooling ensures our network is insensitive to spatial variations of the features. This is particularly important in our task since cancerous cells appear randomly and in various locations of a tissue, and a crucial functionality of our model is to detect them regardless of their location.

7.2 AlexNet

As a second model in our analysis, we chose to train an implementation of AlexNet architecture by suitably adapting it to our binary classification task. Compared to our naive CNN, AlexNet is a much deeper network with far more filters and parameters to train (Kizhevsky, Sutskever and Hinton, 2017). As a far deeper and more complicated structure, we expect it will better detect the small scale and complex patterns of cancerous cells. The extent to which AlexNet performs better than our naive CNN will determine the scope for using more complicated models for which we can get better predictions but will need to mitigate overfitting. AlexNet was one of the first architectures developed for RGB images showing a tredemendous success in performance after LeNet-5 which is fundamentally more suited to grey-scale images, hence our motivation to use it here.

A glimpse of AlexNet, as we have implemented it, is the following:

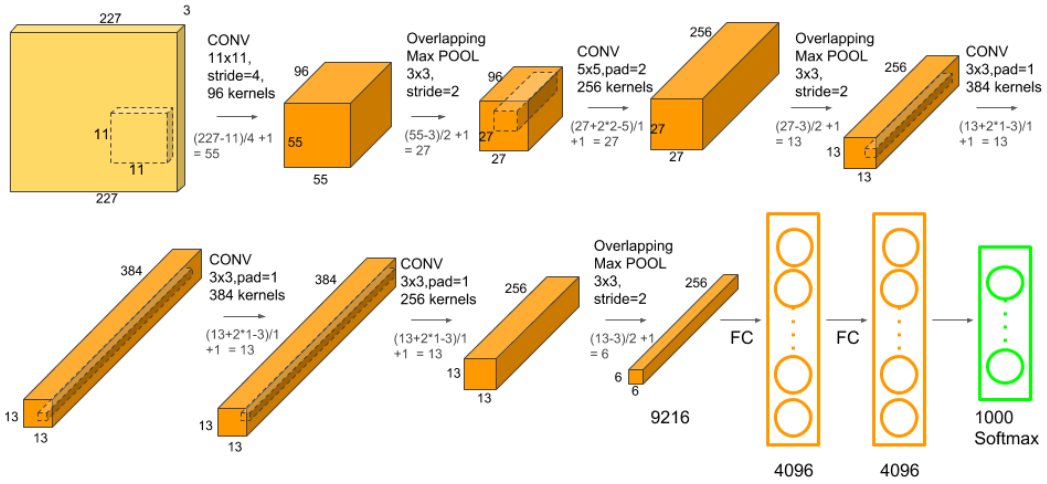


Figure 2: AlexNet architecture

7.3 ResNet

We chose to explore how Transfer Learning can relate to the detection of cancer using state of the art image classification algorithms. By definition, 'Transfer Learning' refers to the ability of recycling a model performing well on one task to solve another different and unrelated task (TensorFlow, 2021). It is a popular approach in deep learning where pre-trained models are used as the starting point on computer vision tasks given how costly it is, in both time and computing resources, to develop neural network models from scratch every time we are given a new research problem.

As such, we were interested to investigate the performance of Pytorch's pre-trained ResNet18 architecture on our histopathological cancer dataset. The ResNet18 model provided by Pytorch has been pre-trained on the ImageNet dataset containing over 14,197,122 images. Given how large and diverse the training set is, we would expect ResNet18 to peform well on our dataset.

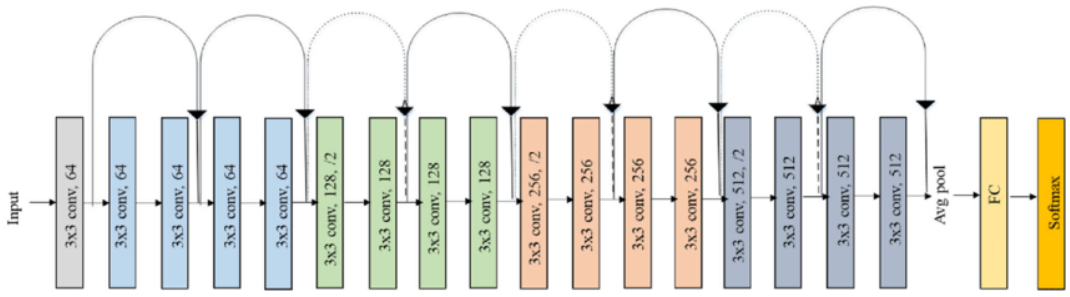


Figure 3: ResNet18 architecture

7.4 VGG11

We decided to use the VGG11 model primarily to compare its performance with AlexNet given the similarities in their architectures. Compared to AlexNet, VGG has has a smaller receptive field as it uses a kernel of size 3x3 (compared to the kernel of size 11x11 used in AlexNet) but this as a result leads to more weights, so the training is considerably slower. Most importantly, VGG features more ReLU activations in its structure, precisely: 8 ReLU functions in the main sequential block and 2 ReLU functions in the linear block. This as a result makes the decision function of the model more non-linea,r allowing it to exploit important features which could have been lost in linearity. A glimpse of the VGG11 architecture is given below:

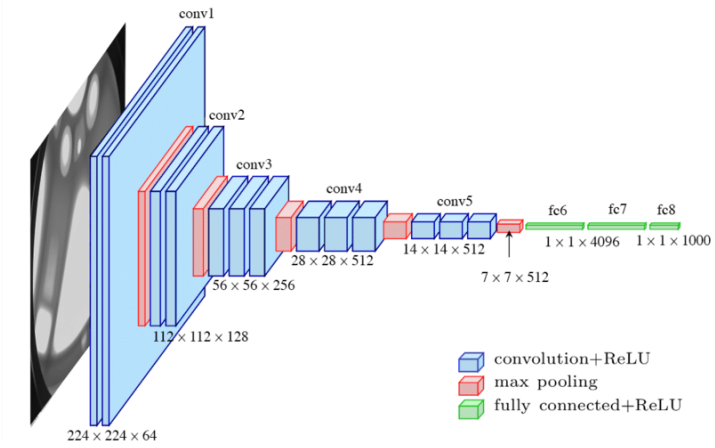
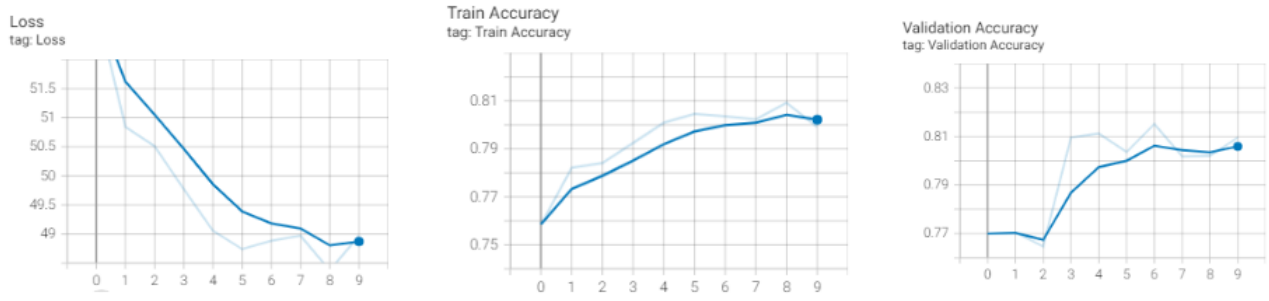


Figure 4: VGG11 architecture

8 Results

8.1 CNN

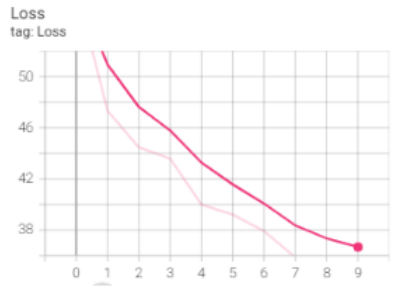
The basic CNN's performance can be considered our baseline model. Being the most simplistic, we expect it to have the lowest test accuracy rate alongside the quickest training time.

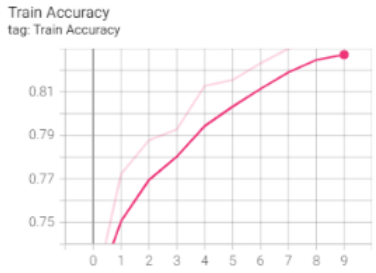


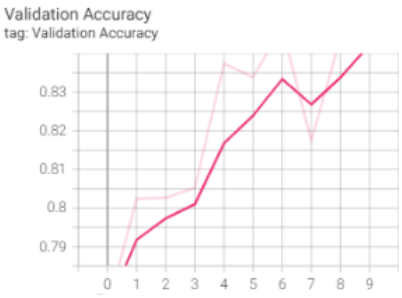
After a relatively quick training period (approximated 5 minutes for 10 epochs), we reach 80+% test and train accuracy. We can improve this accuracy with greater models but initially, this is a good result considering the simplicity of the model.

8.2 AlexNet

As a more complex implementation of a CNN, we expect a marginally better test accuracy. However, we also expect a much longer computational time.



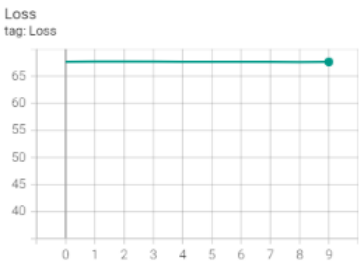


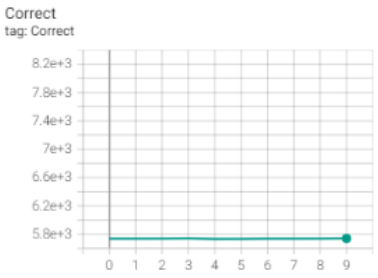


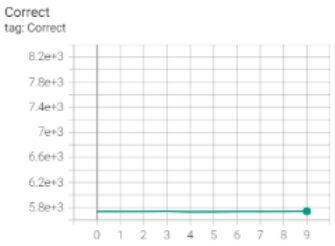
As we can see, the test and train accuracy of AlexNet surpases the basic CNN relatively quickly. With 84+% accuracy rates, this is clearly a succesful module. Furthermore, the training time was relatively quick compared to the other models (around 9 minutes for 10 epochs).

8.3 ResNet18

ResNet is an incredibly complex architecture and it's depth makes it computationally demanding. Being pretrained, we expect some of this complexity to be alleviated. As a result, we hope to see high test accuracy with similar training time as AlexNet.



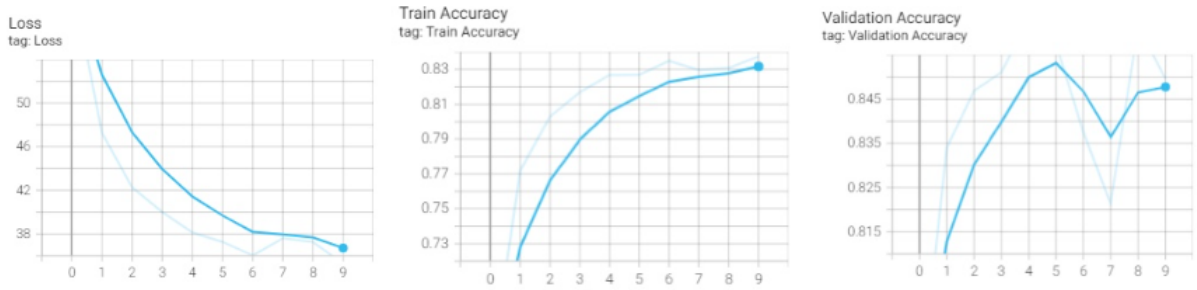




Upon inspection, we notice quick convergence to a 60% test accuracy rate within the first epochs. We can conclude that the pre-trained weights and biases for this model was not appropriate for the dataset. Potentially, the cancer tissue was too specific and the classification task was too niche. Furthermore, our addition of the sigmoid activation function and changing the final connected layer to 2 output classes may not be appropriate for this architecture.

8.4 VGG

Finally, we have VGG. Although the number of layers in VGG is comparable to AlexNet, the smaller convolutions result in a greater number of operations. In total, it was the slowest model to train (approximately 60 minutes for 10 epochs).



We notice a stable decreasing of the loss and relatively similar values for test and train accuracy of around 84%. We compare this more complex model with the results from more simplistic models in the following section.

8.5 Summarised Metrics

Below is a summary of loss and accuracy for all 4 models:

Performance Metrics			
Model	Final Loss	Train Accuracy	Test Accuracy
Basic CNN	0.430	80.2%	80.9%
AlexNet	0.361	82.7%	85.49%
ResNet18	0.683	59.6%	59.6%
VGG11	0.356	83.1%	84.6%

Firstly, we notice that ResNet18 stands out as having the highest loss and lowest accuracy in both categories. As discussed above, we recognise that transfer learning is not applicable in this classification task. The dataset images were too niche and the datset used

to train ResNet18 was not sufficent. Seecondly, we notice that VGG11 has the highest train accuracy rate while AlexNet has the highest test accuracy rate. Considering the size of the dataset, this difference could be explained by only a few observations. We would consider this two models as relatively similar in their predictive abilities.

9 Proposed Model and Conclusion

When proposing a final model we need consider the balance between accuracy and appropriateness for a clinical setting. While a complex and deep network may provide high test and train accuracy, we recognise the need for widespread deployment across healthcare centres and clinicians' devices.

Initially, we disregard the ResNet model. The weights and biases with the pretrained architecture have been overfitted to datasets not comparable to the histopathological tissue scans. To train this model for the datasets would be time consuming and computationally demanding. Furthermore, the deep nature of the network may not be suitable for a clinical setting, where devices may not have the memory requirements for the number of parameters.

Secondly, we can consider VGG unnecessarily complicated. While promising results, VGG is clearly overly-deep for this type of classification problem. The large amount of 1x1 convolutions lead to unnecessarily long computational time. With results similar to our first 2 models, we disregard this model. Furthermore, the test and train accuracy was improved upon with the simpler models.

As a proposed model for the clinical setting, we recommend the AlexNet with the 2 output class modification. The training time was relatively quick. We could see clinicians

loading large amounts of tissue scans for more niche areas that potentially don't have a pretrained model. Furthermore, with an accuracy rate above 84% in test, we can be certain that the model is not over-fitted and we are not concerned for both in distribution and out of distribution bias.

We see clinicians having a mix of pretrained models and custom architectures for diagnostic purposes. For example, a clinician could use a pretrained AlexNet model to identify breast cancer using this dataset. However, for more niche tissue scans, potentially the images would display different indicators. As a benefit of choosing a relatively simplistic model, AlexNet can be trained in a matter of minutes with a custom dataset and deployed across the healthcare center. As a result, we are able to provided targeted models for differing cancer detection or other radiological problems.

To conclude, we have been able to test 4 models in deep neural network architecture to identify breast cancel cells in histopahtolical tissue scans. With varying levels of accuracy, we were able to make slight modifications to the models to better be suited for the binary classification problem. The resulting final model, a binary classifier modification of AlexNet, had a high test accuracy rate of 85.49%. Furthermore, it had reasonable training time. We identified that these two characteristics make it plausible to be implemented in a clinical context.

For future improvements, we could see models being deployed with similar architecture for other radiological scan datasets (other cancers, fracture x-rays, etc). By choosing a model that was computationally feasible for most devices, were able to offer a solution that was applicable for healthcare providers. Furthermore, we developed a training function as well as data transforms that improved the overall training process despite the memory requirements of the dataset. We would look forward to the implementation of deep neural network architecture in the diagnosis regime of healthcare providers and see it as a potential tool to assist clinicians battle aggressive but often unseen illnesses.

References

2022. [online] Available at: <https://www.cdc.gov/nchs/products/databriefs/db293.htm#:~:text=In%202016%2C%20the%2010%20leading,although%20two%20causes%20exchanged%20ranks.> [Accessed 5 May 2022].

Aremu, T., 2022. *AlexNet: A simple implementation using Pytorch*. [online] Medium. Available at: <https://medium.com/analytics-vidhya/alexnet-a-simple-implementation-using-pytorch-30c14e8b6db2> [Accessed 5 May 2022].

Aremu, T., 2022. *VGG-16: A simple implementation using Pytorch*. [online] Medium. Available at: <https://medium.com/@tioluwaniaremu/vgg-16-a-simple-implementation-using-pytorch-7850be4d14a1> [Accessed 5 May 2022].

Cruz, J. and Wishart, D., 2006. Applications of Machine Learning in Cancer Prediction and Prognosis. *Cancer Informatics*, 2, p.117693510600200.

Develop Paper. 2022. *Example of pytorch implementing alexnet - Develop Paper*. [online] Available at: <https://developpaper.com/example-of-pytorch-implementing-alexnet/> [Accessed 5 May 2022].

Kaggle.com. 2022. *Histopathologic Cancer Detection | Kaggle*. [online] Available at: <https://www.kaggle.com/c/histopathologic-cancer-detection> [Accessed 5 May 2022].

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

Ramzan, F., Khan, M., Rehmat, A., Iqbal, S., Saba, T., Rehman, A. and Mehmood, Z., 2019. A Deep Learning Approach for Automated Diagnosis and Multi-Class Classification of Alzheimer's Disease Stages Using Resting-State fMRI and Residual Neural Networks. *Journal of Medical Systems*, 44(2).

TensorFlow. 2022. *Transfer learning and fine-tuning | TensorFlow Core*. [online] Available at: <https://www.tensorflow.org/tutorials/images/transfer_learning> [Accessed 5 May 2022].

University, S., (BCM), B., Paso), T. and (UW), U., 2022. *How AI is improving cancer diagnostics*. [online] Nature.com. Available at: <https://www.nature.com/articles/d41586-020-00847-2> [Accessed 5 May 2022].