Histopathologic Cancer Detection

Geetha Kamath Koteshwar
Department of Computer Science
University of Florida
Gainesville, FL 32611
gkamathkoteshwar@ufl.edu

Madhura Basavaraju
Department of Computer Science
University of Florida
Gainesville, FL 32611
m.basavaraju@ufl.edu

Abstract

This paper discusses the various strategies used for detection of Breast Cancer in Histopathologial slides using Convolutional Neural Networks or CNNs. A CNN works as a classifier mapping a given input image to it's desired class. We conduct experiments by tuning the hyperparameters such as the learning rate, activation function and objective functions. CNNs are widely used in the field of Computer Vision, we also use a SVM for the purpose of classification.We find that for supervised Image Classification problems, CNNs are the most suitable option.

1 Background and Motivation

Metastasis is the spread of cancerous cells from a site of origin to new body parts usually through lymph systems or bloodstream. Metastatic tumor is the medical term given to the tumor that has spread from primary site of origin in the body to different areas of the human body.

Histopathology can be described as the branch of Biology that studies the symptoms and implications of the disease using a part of the tissue that is excised, processed and fixed onto glass slides to be studied under a microscope. Different parts of the specimen may react to different dyes.

Breast cancer is known to be the most common and deadly cancers in women worldwide. The primary step in detecting metastatic cells is to examine the regional lymph nodes. Lymph nodes are small glands that filter the fluid in the lymphatic system, and they are the first place a breast cancer is likely to spread. Histological assessment of lymph node metastases is part of determining the stage of breast cancer in TNM classification which is a globally recognized standard for classifying the extent of spread of cancer[4]. The data consists of histopathological images which are glass slide microscope images of lymph nodes that are stained with one of the most widely used staining method, hematoxylin and eosin (H&E). Dark blue hematoxylin binds to nuclei while eosin binds to cytoplasm and extracellular parts of the cell staining them pink.

Lymph node metastases can have the following features:

- Foreign cell population - key feature (Classic location: subcapsular sinuses)
- Cells with cytologic features of malignancy
- Nuclear pleomorphism (variation in size, shape, and staining)
- Nuclear enlargement.
- Irregular nuclear membrane.
- Irregular chromatin pattern, esp. asymmetry.
- Large or irregular nucleolus.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

- Cells in architectural arrangements seen in malignancy
- Gland formation.
- Single cells.
- Small clusters of cells.

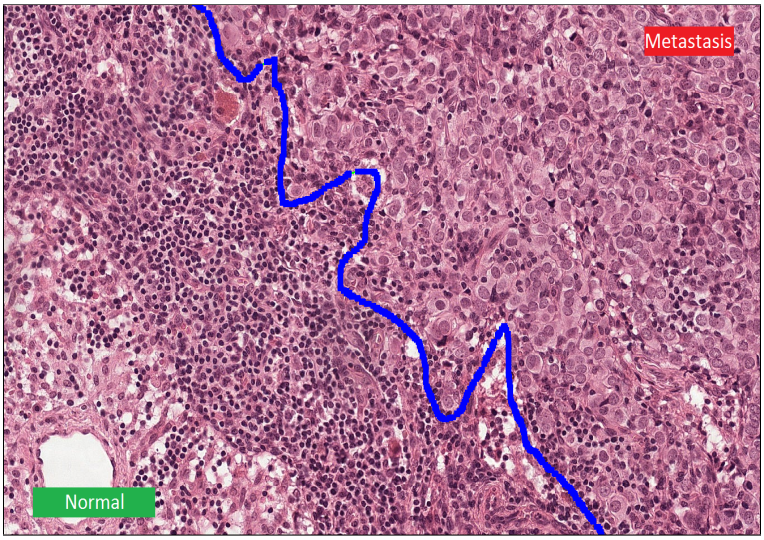

Figure 1: Metastasis

In summary, it can be said that irregular nuclear sizes, shapes, and staining shades can often indicate metastases. Diagnostic procedure for pathologists is tedious and time-consuming as a large area of tissue must be examined and small metastases can be easily missed.

Thus, the utilization of deep learning in the diagnostic process can be a great choice in terms of both ease of usability and precision.

2 Problem formulation

2.1 Data

The dataset is a subset of the PCam dataset comprising of 220,025 training images and 57,468 evaluation images and with all the duplicate images being removed. The PCam's dataset including this one uses 10x undersampling to increase the field of view, which gives the resultant pixel resolution of 2.43 microns. This dataset combines the collection of two datasets collected independently at Radboud University Medical Center (Nijmegen, the Netherlands), and the University Medical Center Utrecht (Utrecht, the Netherlands).

Table 1: Image file descriptors

Feature	Description
Format	TIF
Size	96 x 96
Channels	3
Bits per Channel	8
Data Type	Unsigned char
Compression	Jpeg

Visualizing the statistics of the distribution in the training data shows a ratio of 60 positive samples to 40 negative samples.

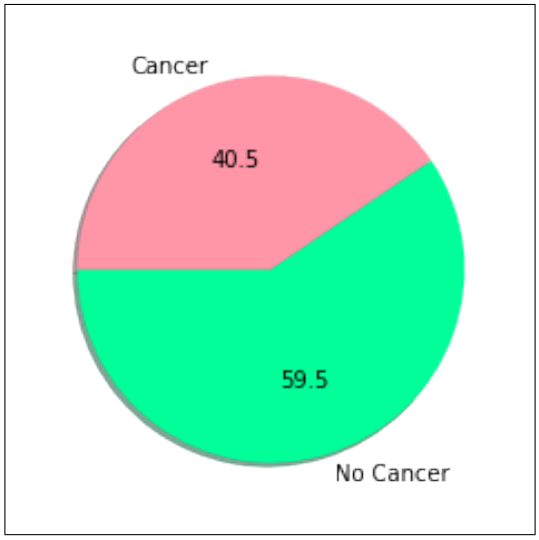

Figure 2: Distribution of training data samples

An image labelled as positive indicates at the least one pixel of tumor tissue in the center 32 x 32 portion. Therefore, the rest of the portion of the image that lies outside the centric region has no influence over the labelling of the samples. In turn, this implies that a negative sample might have metastases outside the centric 32 x 32 region of the image. Each of these images depict slides produced by clinically which trained pathologists use in the detection of metastases.

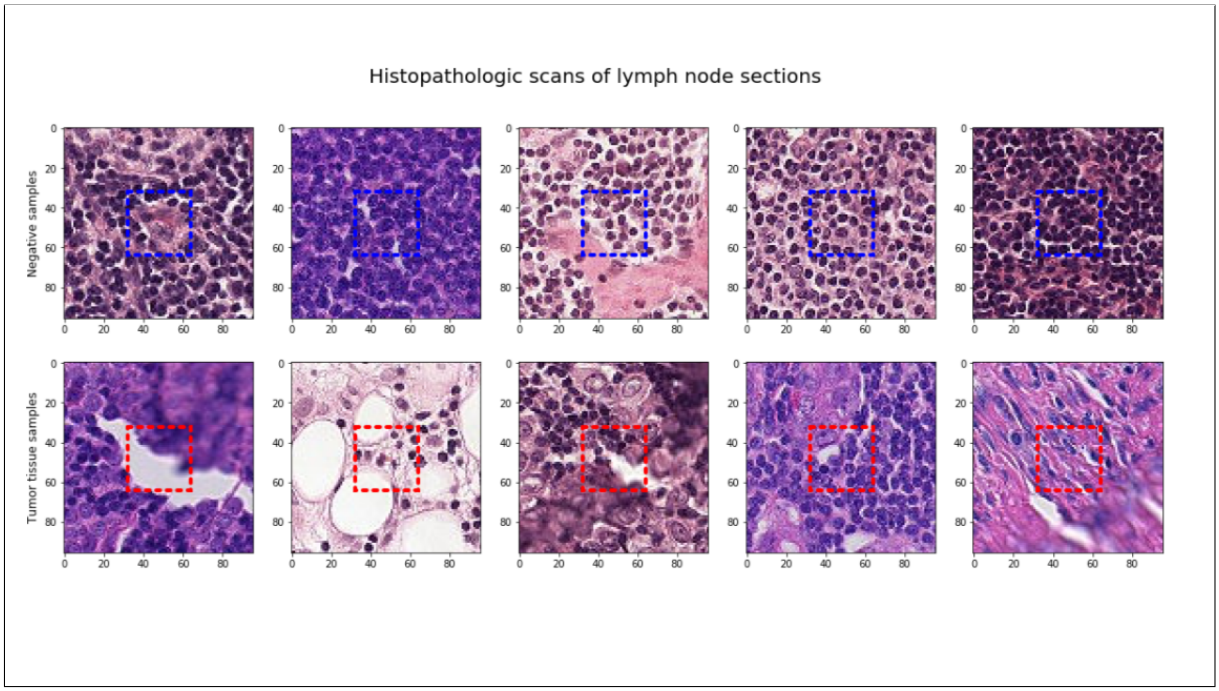

Figure 3: Positive and negative samples

2.2 Problem Statement

Binary classification of images:

Create an algorithm to identify metastatic cancer in small image patches taken from larger 96 x 96px digital pathology scans.

Binary Classification as an Optimization problem:

The aim is to minimize the loss function/cost function using an efficient optimizer.
The cost function of a binary classification problem can be defined as the penalty due to inaccurate predictions made by the designed model.

3 Algorithm

3.1 CNN

Convolutional Neural Networks[1] show state- of-the art performance in image classification. CNN's are known to have the ability to reduce the number of parameters without a loss in the quality. Since images have high dimensions, the above described quality of CNNs make the algorithm a best fit to classify images. The image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers as shown in figure 4, through which the algorithm employs transfer learning to pick up characteristics of the image in a layered fashion to generate the final output. This kind of transfer learning does not happen in other machine learning algorithms such as knn, svm, logistic regression etc. Due to transfer learning the deep learning models learn more which is essential for classifying high dimensional data such as images causing the less occurrence of errors. These multi-layered networks are equipped with the ability to learn complex, high dimensional, non-linear mappings from large collections of examples. Hence making them the obvious candidates for image classification and recognition.

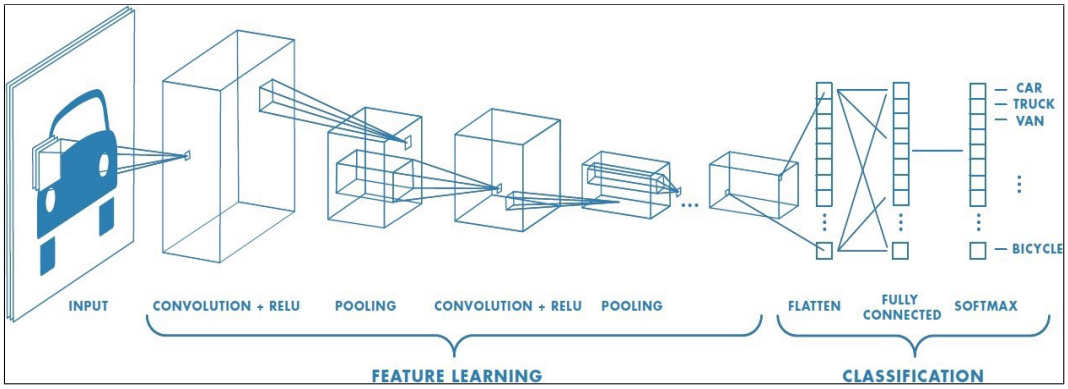

Figure 4: Convolutional Neural Network

3.1.1 Convolutional layer

Convolution layer is the main building block of CNNs. Mathematically, convolution is combining two sets of information. In a CNN architecture the convolutional layer appears first. A feature map is produced by applying convolution to the input data. This feature map is extracted using a convolutional filter. The convolution operation is applied by sliding this filter over the input,and thus operating on two images: the image given as input and the filer on the original image,also called the kernel.The results of element wise matrix multiplication computed at every locations are summed up. This sum goes into the feature map. The convolution of two functions f and g can be defined as follows:

$$(f * g)\,(i) = \sum_{j=1}^{m} g(j).f(i - j + m/2)$$

which is the dot product of the input function and kernel function. Channels depict colors. A colored image has multiple channels, one for each color. The number of channels in a filter must be the same as the number of channels in the input. Performing convolution on image with different filters can

result in operations such as edge detection, blurring and sharpening of the image. Stride denotes the number of pixels shifts of the filter over the input matrix. To prevent the feature maps from shrinking at every layer, it is common to use padding in CNN. Multiple convolutions are performed on the input, each time using a different filter which results in a distinct feature map. Final output of the convolution layer is these feature map stacked together.

3.1.2 Pooling layer

To reduce the dimensionality after convolution, pooling operation is performed thus enabling the reduction in the number of parameters. Pooling ensures that the depth of the feature map is intact whilst down sampling each feature map independently by reducing the height and width.Through pooling, the dominant features are extracted which are positional and rotational invariant. Like in convolution layer, the window size and stride are specified. Max pooling, which is the most used type of pooling takes the max value in the pooling window.

3.1.3 Fully connected layer

Fully connected networks form the last few layers of the network. A fully connected layer expects a 1D vector of numbers. The output from the last pooling layer is first flattened. This flattened output is then fed as input to the fully connected layer. Flattening can be defined as transforming 3D volume of numbers into a 1 dimensional vector. While the convolution layers serve the purpose of feature extraction, classification of the data into various classes is done using fully connected layers.

3.1.4 Activation Function

In a neural network, each neuron in the input layer is fed input, where it gets multiplied with the weight of the neuron which gives the output of that neuron.The activation function serves as an intermediary between the input being fed to the neuron and the output that it gives to the next layer. It performs a transformation that maps the input signal to the output signal. Mostly, the activation functions used are non-linear. Some of these are as follows:

ReLU Activation: Rectified Linear Unit (ReLU) is an activation function that is widely used[6] because it is sparsely activated and converges faster. This function resembles a linear function but acts like a non-linear function. This allows it to learn and understand complex relations in data.
ReLU can be mathematically represented as follows:

$$\phi(x) = max(0, x)$$

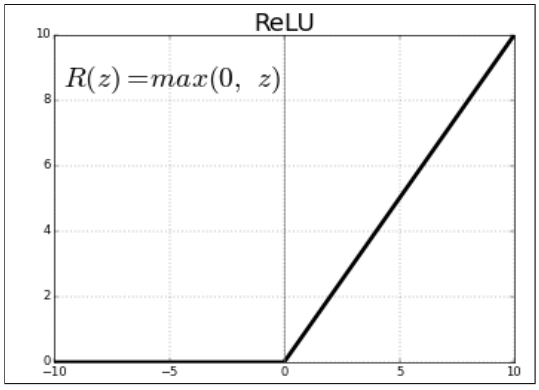

Figure 5: ReLU Activation Function

Sigmoid Activation: Sigmoid functions are non-linear by nature, they are continuously differentiable and their output lies between the range of 0 and 1.
Sigmoid can be mathematically represented as follows:

$$\phi(x) = \frac{1}{1 + e^{-x}}$$

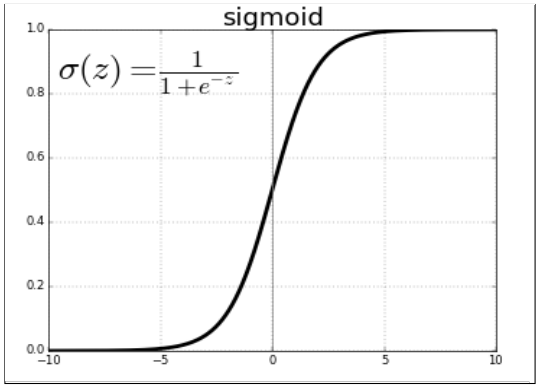

Figure 6: Sigmoid Activation Function

TanH Activation: TanH activation function is non-linear by nature, similar to the Sigmoid function, but its output is in the range between -1 and 1.
TanH can be mathematically represented as follows:

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

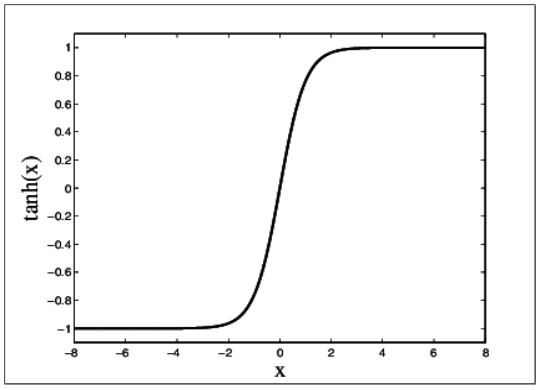

Figure 7: TanH Activation Function

Softmax Activation: Softmax activation function is used in the final layer of a Neural Network,it calculates the probability distribution of each target class over different possible classes. This is used in determining the target class for a given input data point.

3.1.5 Learning Rate

Learning rate is a hyper-parameter in Machine Learning that determines the step size during every iteration while minimizing a loss function. Choosing the learning rate follows a trial and error approach that is, there needs to be a reasonable trade-off between rate of convergence

3.1.6 Objective Function

Objective Function measures the correctness of a neural network by mapping the relationship between the input and the output. It does so by finding the difference between the desired output and the predicted output. Some of them are as follows:
Mean Squared Error: Mean squared error can be mathematically represented as follows:

$$l = \frac{1}{n}\sum_{i=1}^{n}(y_i - p_i)^2$$

where y_i is the desired output and p_i is the predicted output for the i^{th} data point

Binary Cross Entropy: Binary Cross Entropy can be mathematically represented as follows:

$$l = -\frac{1}{n}\sum_{i=1}^{n}(y_i log(p_i) + (1-y_i) log(1-p_i))$$

where y_i is the desired output and p_i is the predicted output for the i^{th} data point. Both $y_i$and p_i have values of either 0 or 1

Squared Hinge Loss: Squared Hinge Loss can be mathematically represented as follows:

$$l = \sum_{i=1}^{n}(max(0, y_i.p_i)^2)$$

where y_i is the desired output and p_i is the predicted output for the i^{th} data point. Both $y_i$and p_i have values of either -1 or 1

3.1.7 Optimizer

An optimizer's main task is to find parameters that can minimize the loss function. To achieve this, we run the algorithm multiple times with different weights and find the weights that minimized the loss function. This is called Gradient Descent. The weight updation is as follows:

$$w_t = w_{t-1} + \Delta w$$

where w is the weight, t is the time step and Δw is the change in weight, which minimizes the loss function. Δw is given by:

$$\Delta w = -\eta \Delta L$$

where eta is the learning rate and L is the loss function.There are several types of optimizers, some of them are as follows:

Adadelta: The mathematical formulation for Adadelta is as follows:

$$[w_t = w_{t-1} + \Delta w$$

$$\Delta w = -\frac{RMS[\Delta w]_{t-1}}{RMS[g_t]}.g_t$$

where g_t is the sum of all he past gradients with respect to w.

ADAM: The mathematical formulation for Adaptive Moment Estimation (ADAM)[3] is as follows:

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

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

where β_1 and β_2 are hyper-parameters that control the exponential decay rates and $m_t$and v_t are estimates of first and second moment respectively

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

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

where $\hat{m_t}$and $\hat{v_t}$ are bias corrected estimates of first and second moment. Therefore,

$$w_{t+1} = w_t - \frac{\eta \hat{m_t}}{\sqrt{\hat{v_t} + \epsilon}}$$

RMSProp: The mathematical formulation for Root Mean Square Propagation (RMSProp) is as follows:

$$w_{t+1} = w_t - \frac{\eta}{\sqrt{(1-\gamma) g_{t-1}^2 + \gamma g_t + \epsilon}}.g_t$$

where γ is the decay rate

3.2 SVM

Support Vector Machine[7] categorized as a supervised Machine Learning algorithm can be used for both classification and regression problems. More commonly it is used in solving classification problems. The data points required to be classified are plotted on an n dimensional feature space, where n is the number of features of every data point. Data points are differentiated and classified into two classes by identifying a separating hyperplane.
Any hyperplane can be written mathematically as follows:

$$\beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + ... + \beta_n * x_n = 0$$

A hyperplane for a n-dimensional space is an (n-1) subspace separating the two classes of data. Given a hyperplane, the smallest perpendicular distance of all perpendicular distances drawn from data points to the hyperplane is the margin of the hyperplane. When dealing with linearly separable data, SVM aims to find a separating hyperplane such that the margin of the hyperplane is maximized.

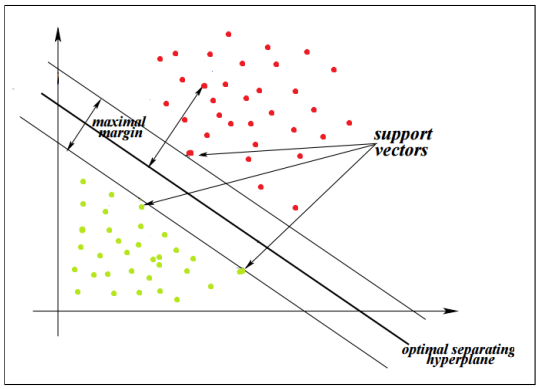

Figure 8: SVM

Nonlinearly separable data can be handled using kernel trick and soft margins. Kernel trick involves utilizing input features to create new features. These new features are then, used in forming a nonlinear decision boundary. By converting low dimensional input to a higher dimensional space, the problem is transformed into a linearly separable form.

Soft Margin introduces tolerance to SVM. SVM is allowed to make a certain number of mistakes and keep the margin as wide possible, making allowance for the points to be classified correctly. We would aim to maximize the following objective:

$L = \frac{1}{2}||w||^2 + C(\text{number of mistakes})$

C is the hyper parameter that designates the trade of between the maximizing the margin and and minimizing the mistakes

4 Methodology

4.1 Convolutional Neural Network-1

4.1.1 Data Preprocessing

After making careful observations about the data,we realised that there was an imbalance in the number of positive and negative samples in the dataset, this might have led to an overfit model. There were around 130,000 positive and 89,000 negative samples which corresponds to 59% and 41% respectively. To avoid this, we modified the dataset to contain 80,000 images belonging to each class. This set was further divided into training set and validation sets, containing 144,000 and 16,000 images respectively. We used the stratify option to evenly divide the training data into training dataset

and validation dataset, each containing equal number of images from both the classes. 7 images were omitted from the training dataset for being entirely black or entirely white.

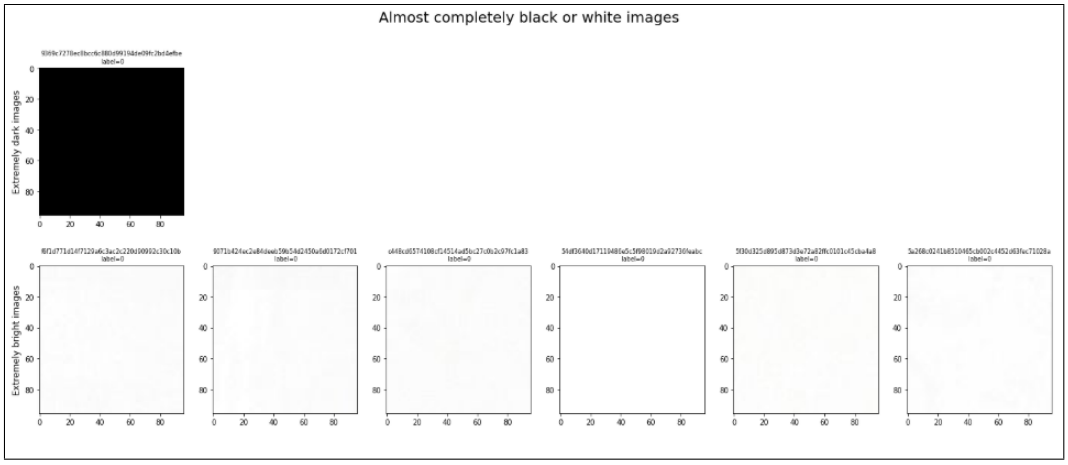

Figure 9: Completely black or white images

4.1.2 Network

The Convolutional Neural Network consisted of 11 layers. The input to the network is an image of size 96x96x3, where 3 is the number of channels and 96x96 is the size of the image in pixels. Te input layer consists of 27,648 neurons. Following which are 3 sets of layers, each set containing 3 convolutional layers.The number of neurons in each of these sets is 32,64 and 128 respectively. Following these layers are 22 fully connected layers. The output layer consisted of 2 neurons, because the number of classes is 2. The activation function for the convolutional layers and the first dense layer is ReLU and the activation function for output layer is Softmax. Each of these convolutional layers have a kernel size of 3x3.

4.1.3 Training

The CNN was trained with a learning rate of 0.0001, The optimizer chosen was ADAM and the objective function chose was Binary Cross Entropy. The data was divided into 10 batches, each batch containing 14400 images. The model as trained for 20 epochs. After each epoch, the best model is saved. This model is chosen based on validation accuracy.
After every three convolution layers, a max pooling layer followed by a dropout[2] layer was used. Max pooling layers were used to reduce the number of parameters. During training, weight of 30% of the neurons was set to 0 due to the dropout layer. This was done to avoid an overfit model.

4.2 Convolutional Neural Network-2

4.2.1 Data Preprocessing

Training dataset contains 220,025 images. The data is turned into a PyTorch dataset. We sample out a shuffled part of the train data (160000 images) in order to split the data into train and validation sets. In order to improve performance data augmentations are added to the train data.

4.2.2 Network

We have designed a neural network with 6 layers. The input to the network is an image of size (3x96x96), where 3 corresponds to the number of channels and (96x96) is the size of each image. The first 5 hidden layers are two dimensional convolutional layers and the last 1 layer is fully connected layer. The activation function for all the convolutional layers and the first dense layer is rectified linear unit (ReLU). The activation function for the output layer is sigmoid.

4.2.3 Training

During training, dropout was used. A dropout layer was present after flattening the fifth convolutional layer output. While training, at each iteration, there is a probability of 0.5 at each node whether to keep it in the network or deactivate it out of the network. In other words, the weights corresponding to it were set to zero and got updated only p fraction of times.This is done to avoid overfitting and regularize the network. Dropout is related to the concept of ensemble learning with the unique case that the various models in the ensemble share parameters and these models are "combined" into a single model at test.

5 Results

5.1 Convolutional Neural Network-1

Before performing other experiments on the model, we wanted to study the effect of normalisation on the accuracy. The input was normalised and it was observed that the accuracy was higher. Each of these models had Adam optimiser, Binary Cross Entropy Objective Method, 9 convolutional layers with ReLU activation and 2 fully connected layers with ReLU and Softmax activation respectively. We can observe from Fig. 10 that the change in accuracy between the normalised and the model trained without normalization, is not staggering. This could be due to the fact that all the parameters of the image are on the same scale. However, all the models trained henceforth will be trained with normalised input.

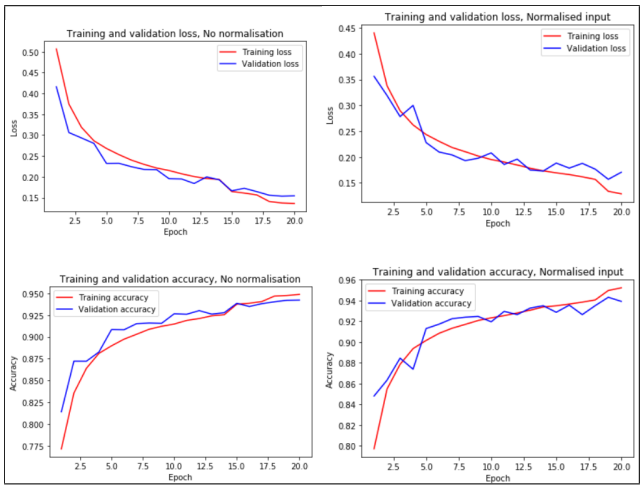

Figure 10: a.Without Normalization b.With Normalization

Table 2: Normalization experiment results

	Validation Accuracy	Validation Loss
Without Normalization	0.93225	0.1545
With Normalization	0.94667	0.1525

To choose the learning rate, we chose to train 5 different models with five different learning rates. Each of these models had Adam optimiser, Binary Cross Entropy Objective Method, 9 convolutional layers with ReLU activation and 2 fully connected layers with ReLU and Softmax activation respectively. Fig 11 shows the results of our experiments. We can observe that a Learning Rate of 0.0001 gave us best results, wit low loss and high accuracy. With high learning rate such as 0.1 and 0.01, the model becomes highly unstable, this is because when the learning rate is high, the model tends to take big steps during gradient descent, thus leading to erroneous output. A high learning rate also yield low

accuracy because it converges too quickly, thus leading to a less than optimal solution as shown in Fig. 11.

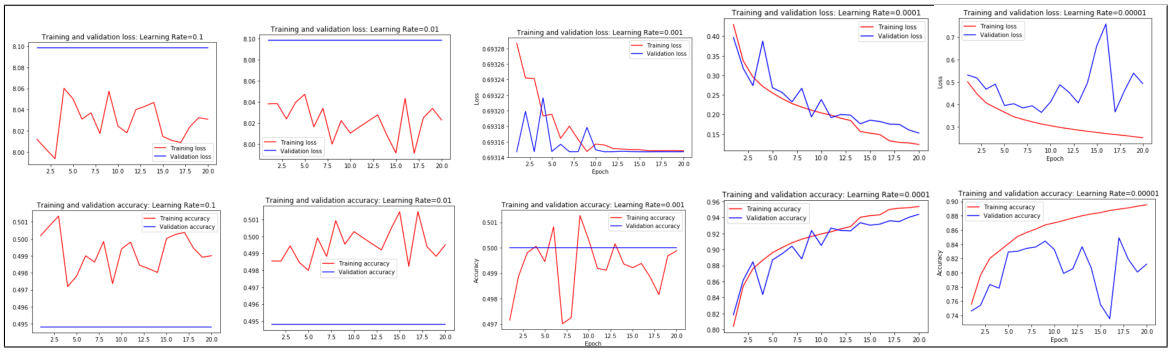

Figure 11: a.0.1 b.0.01 c.0.001 d. 0.0001 e. 0.00001

Table 3: Learning Rate experiment results

Learning Rate	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
0.1	0.501	7.9	0.495	8.10
0.01	0.502	7.9	0.494	8.10
0.001	0.502	0.69315	0.5	0.69315
0.0001	0.95	0.10	0.94	0.17
0.00001	0.90	0.28	0.84	0.4

To choose the activation function, we tested 3 functions namely ReLU, TanH and Sigmoid function. To test the activation functions, the learning rate was set at 0.0001, the optimiser used was Adam, and the binary cross entropy objective function was used. The following results (Fig. 12) were obtained. From these results, it can be observed that ReLU results in the lowest loss and gives highest accuracy. ReLU allows for generalization and also converges faster. This could be due to the nature of ReLU itself, which has very primitive mathematical computations.Therefore, ReLU is set as the Activation function.

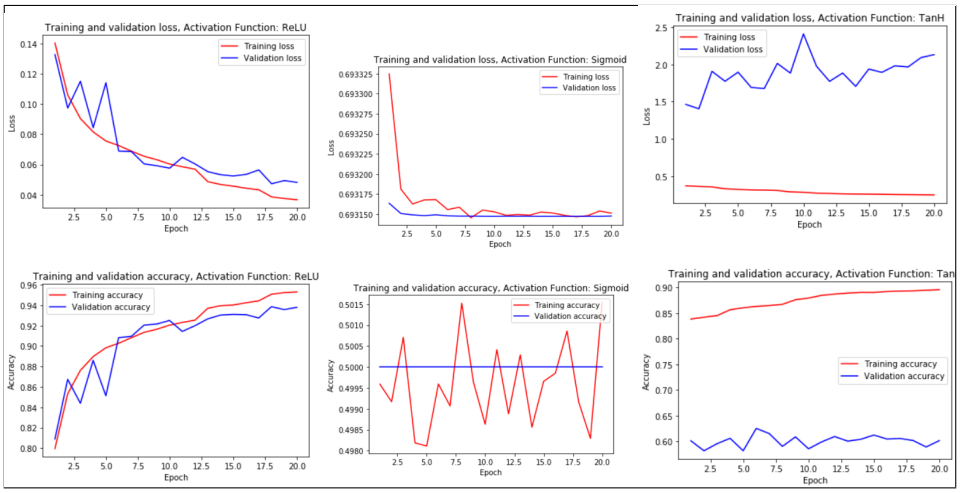

Figure 12: a. ReLU activation b. Sigmoid activation c. TanH activation

Table 4: Activation function experiment results

Activation Function	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
ReLU	0.95	0.15	0.94	0.20
Sigmoid	0.95	0.04	0.93	0.05
TanH	0.96	0.52	0.94	0.52

To choose the the objective function, we tested five different loss functions maintaining the learning rate at 0.0001, using Binary Cross Entropy Objective Method, Adam Optimiser and 9 convolutional layers with ReLU activation and 2 fully connected layers with ReLU and Softmax activation respectively. The results are as shown in table 5 and Fig 13. As we can see, mean squared loss gives a much lower Validation loss when compared to Binary cross entropy, but we choose Binary Cross Entropy because it chooses specific values, either 0 or 1, but Mean Squared error tends to choose a class based on having larger values correspond to higher probabilities. BCE also gives a more stable and quicker learning in fewer passes through data.

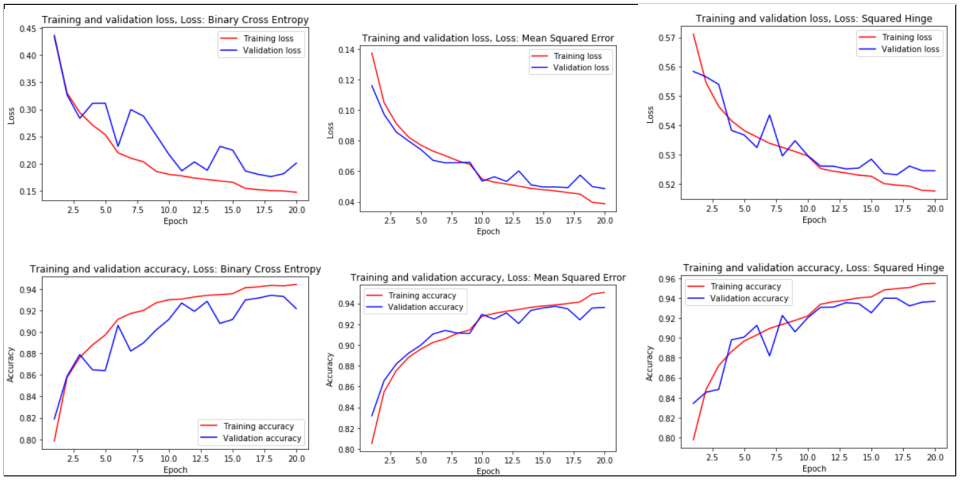

Figure 13: a. Binary Cross Entropy b. Mean Squared Error c. Squared Hinge Loss

Table 5: Objective function experiment results

Loss Function	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
Binary Cross Entropy	0.95	0.15	0.94	0.20
Mean Squared Error	0.95	0.04	0.93	0.05
Squared Hinge Loss	0.96	0.52	0.94	0.52

To choose an optimizer, we performed experiments with 2 optimizers, ADAM optimiser and RMSProp Optimiser. We trained the models while maintaining the learning rate at 0.0001, using Binary Cross Entropy Objective Method and 9 convolutional layers with ReLU activation and 2 fully connected layers with ReLU and Softmax activation respectively. ADAM gave a much better accuracy as well as validation loss as shown below. We believe this is because of the sparse gradient

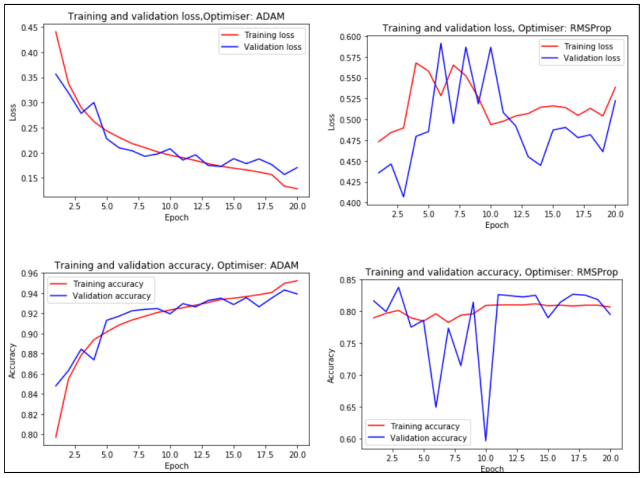

Figure 14: a. ADAM optimiser b. RMSProp Optimiser

Table 6: Optimizer experiment results

Optimizer	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
ADAM	0.95	0.135	0.9431875	0.15678
RMSProp	0.80	0.410	0.8375	0.40695

5.2 Convolutional Neural Network-2

We train 3 different models with different learning rates and the results are tabulated below. Each of the models utilize RELU activation function, Binary Cross Entropy loss function with Adam optimizer. A learning rate of 0.0001 is observed to give the best results. With high learning rate such as 0.1 and 0.01, the model becomes highly unstable, this is because when the learning rate is high, the model tends to take big steps during gradient descent, thus leading to erroneous output. A high learning rate also yield low accuracy because it converges too quickly, thus leading to a less than optimal solution.

Table 7: Learning Rate experiment results

Learning Rate	Training Loss	Validation Accuracy	Validation Loss
0.001	0.105	0.1292	0.98
0.0001	0.128	0.124	0.980
0.00001	0.24	0.237	0.948

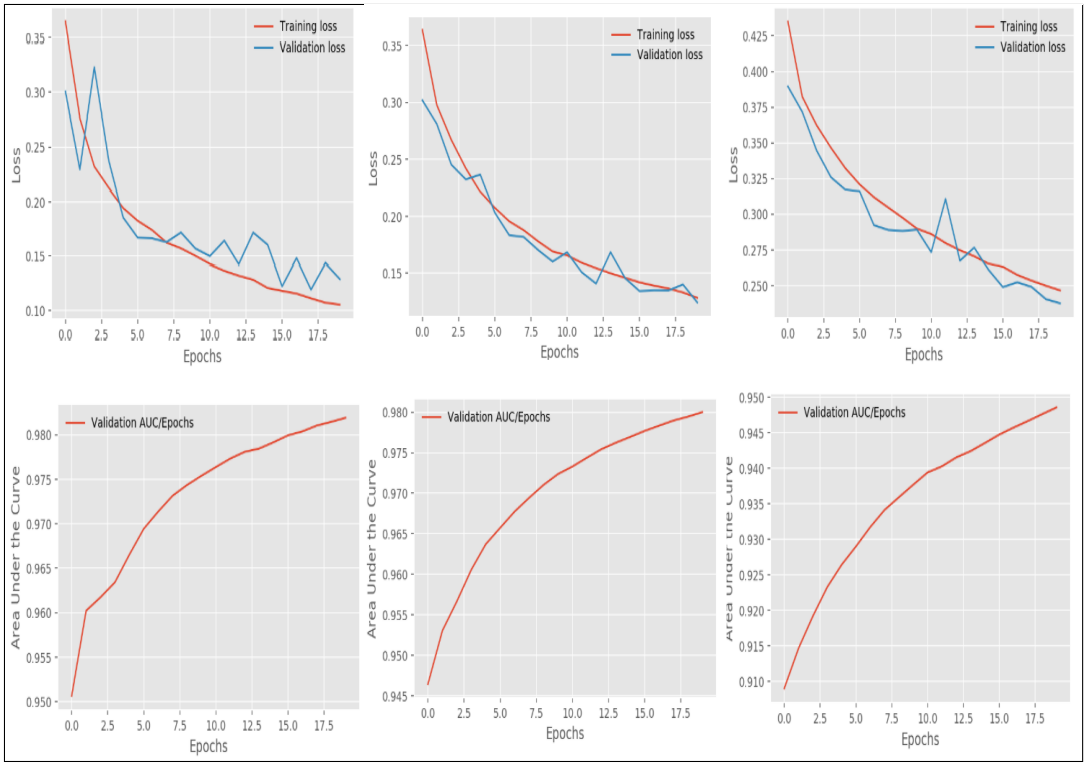

Figure 15: a. lr=0.001 b. lr=0.0001 c. lr=0.00001

We train 2 different models with different loss functions while the learning rate is set to 0.0001. The results are tabulated below. Each of the models utilize RELU activation function with Adam optimizer. BCELoss function is observed to give the best results. We choose Binary Cross Entropy because it gives a more stable and quicker learning in fewer passes through data.

Table 8: Loss Function experiment results

Loss Function	Training Loss	Validation Accuracy	Validation Loss
BCELoss	0.559	0.554	0.933
BCELogitsLoss	0.128	0.124	0.980

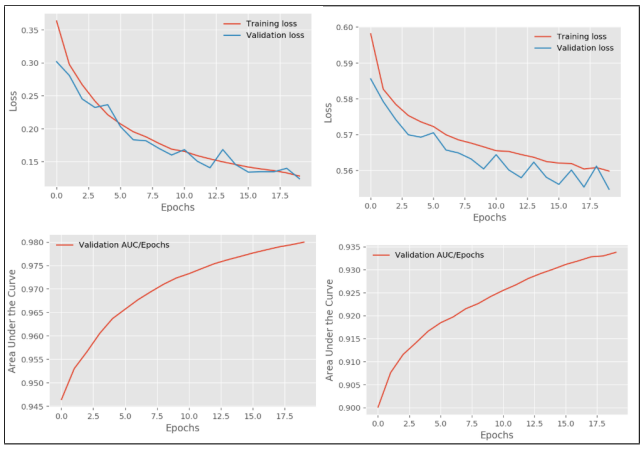

Figure 16: a. BCELoss b. BCELogitsLoss

We train 3 different models with different activation functions with the BCELoss function with a learning rate is set to 0.0001. The results are tabulated below. Each of the models utilize Adam optimizer. RELU activation function is observed to give the best results

Table 9: Activation Function experiment results

Loss Function	Training Loss	Validation Accuracy	Validation Loss
Sigmoid	0.250	0.721	0.933
RELU	0.128	0.124	0.980
Tanh	0.166	0.158	0.965

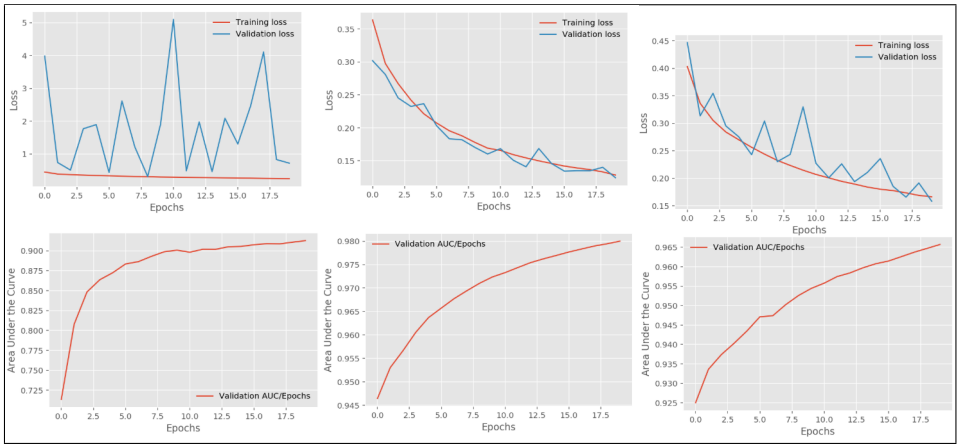

Figure 17: a. Sigmoid b. RELU c. Tanh

5.3 Support Vector Machine

The SVM gave a validation accuracy of 0.5000 and validation loss of 0.5000. The model was trained using a L2 regularizer with a value of 0.01. The results are as follows

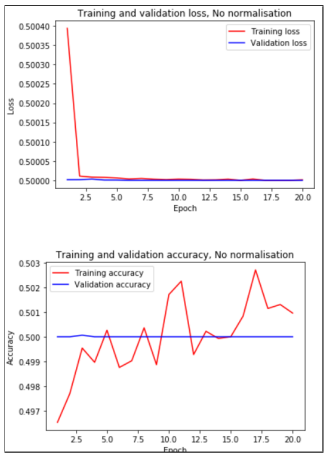

Figure 18: SVM results

6 Conclusion

Our final solution is a model with 9 convolutional Layers, where the number of neurons in the first three is 32, followed by 64 in the next 3 and 128 in the last 3. The kernel size for each of these layers

in 3x3. Each of these convolutional Layers has the ReLU Activation Function. After every three convolutional layers, there is a max pooling layer with a pool size of 2x2, and a dropout layer, with dropout=0.3. The last two layers are fully connected. The first fully connected layer has 256 neurons with ReLU activation and the last layer has 2 neurons with Softmax activation. We train this model with a learning rate of 0.0001, and use ADAM for optimizing the parameters. We use Binary Cross Entropy as the Objective Function. We train this model on normalised data to achieve a validation loss of and validation accuracy of.The results are as follows:

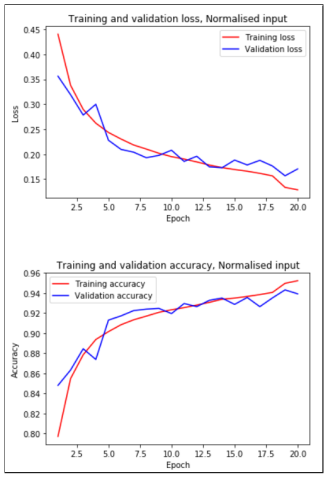

Figure 19: Final Result

References

[1] Yann LeCun, Leon Bottou, Yoshua Bengio, Patrick Haffner. Gradient-based learning applied to document recognition. Published 1998.

[2] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever,Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15 (2014) 1929-1958.

[3] Kingma, Diederik P and Ba, Jimmy Lei. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.

[4] Y. Liang, J. Yang, X. Quan and H. Zhang, "Metastatic Breast Cancer Recognition in Histopathology Images Using Convolutional Neural Network with Attention Mechanism," 2019 Chinese Automation Congress (CAC), Hangzhou, China, 2019, pp. 2922-2926.

[5] Bishop, Christopher M. Pattern Recognition and Machine Learning. New York :Springer, 2006.

[6] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML'10). Omnipress, Madison, WI, USA, 807–814.

[7] Cortes, Corinna; Vapnik, Vladimir N. (1995). "Support-vector networks" (PDF). Machine Learning. 20 (3): 273–297.CiteSeerX 10.1.1.15.9362. doi:10.1007/BF00994018.