Cancer Detection using CNNs

CS736 Project

Siddharth Khandelwal, Prasann Viswanathan

Understanding the Problem…

- Our aim is to train a classifier to identify cancer from small tissue images
- This is an important area of research because it is essential to predict the onset of cancer as well as identifying what stage of cancer has reached
- Also, the diagnostic procedure is a tedious one as it involves checking many scans and small details can be easily missed, as a result

Understanding the Data...

- The data consists of 220,000 images in training and 57,000 in test sets
- They have been derived from the PCam dataset, and have been cleaned of duplicates
- Roughly 60% are negative for cancer and 40% positive
- These images and their labels were obtained from a Kaggle dataset

Understanding the Images...

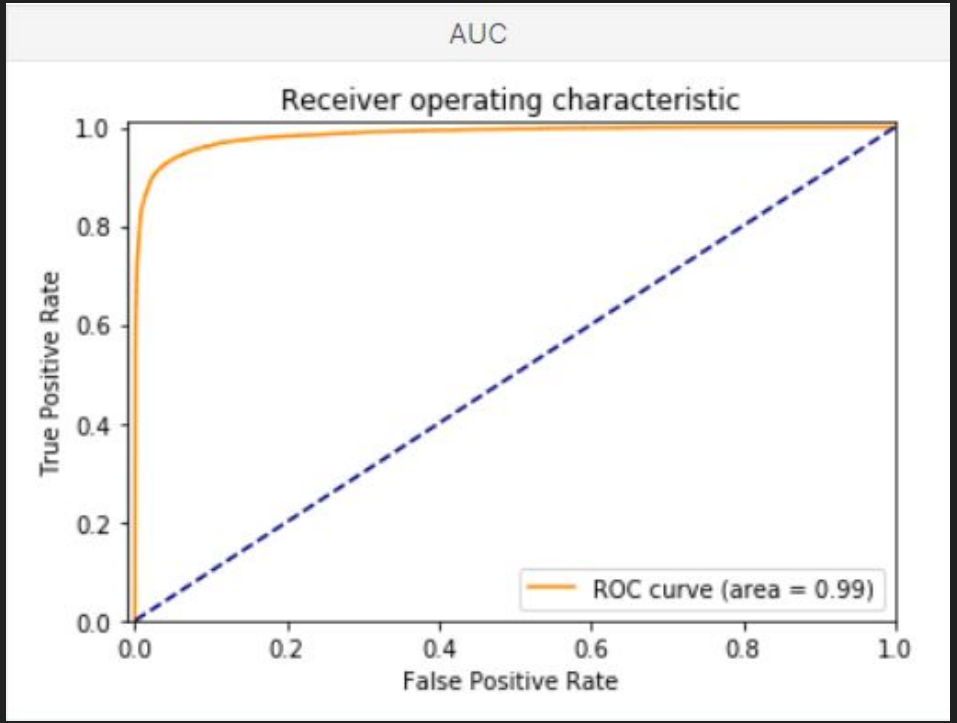
- Our classifier must identify cancer only within the central 32x32 pixels
- Therefore a positive label indicates presence of cancer in this region
- The outer region, does not influence the label
- To allow convolution and padding, the outer region is provided

Understanding the Evaluation Metric...

- The metric is Area Under ROC curve, one of the leading metrics to evaluate the performance of a classification models performance
- ROC is a probability curve and AUC represents degree or measure of separability.
- Also we have defined a accuracy criteria which gives an idea of the accuracy of the model on the validation set.

Looking at AUC/ROC

- Higher the AUC, better the model is at predicting 0s as 0s and 1s as 1s. By analogy, higher the AUC (close to 1)
- The curve is plotted with True Positive Rates Vs the False Positive Rates along the x and y axes respectively.

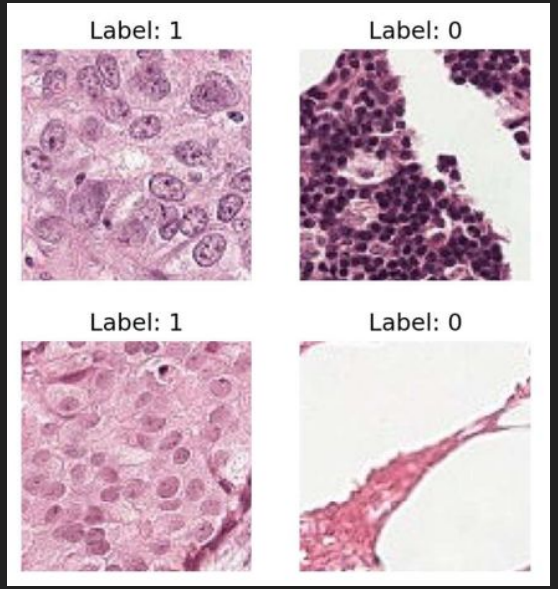


How to Load the Data?

- We ran the training model locally, making use of computation speeds provided by local GPU on our laptops
- This was accomplished by os library of python

Visualising the Data

- Classifying metastases is probably not an easy task for a trained pathologist and extremely difficult for an untrained eye when we take a look at the images.



Sampling and Data Pre-processing

- Since the train dataset contains 220,025 images we can sample out a shuffled part of that, in this case 80000 samples and train on them to make predictions later. (As training on all images will take forever)
- First we turn our data into PyTorch dataset then the data is sampled into train and validation sets. Data Augmentations are added for train data to improve performance.
- In order to avoid overfitting, when any classification parameter needs to be adjusted, it is **necessary** to have a **validation dataset** in addition to the training and test **datasets**.
- We also add random horizontal and vertical flips as well as rotations of our image to better train our model
- All of these functionalities are available in the **PyTorch** library

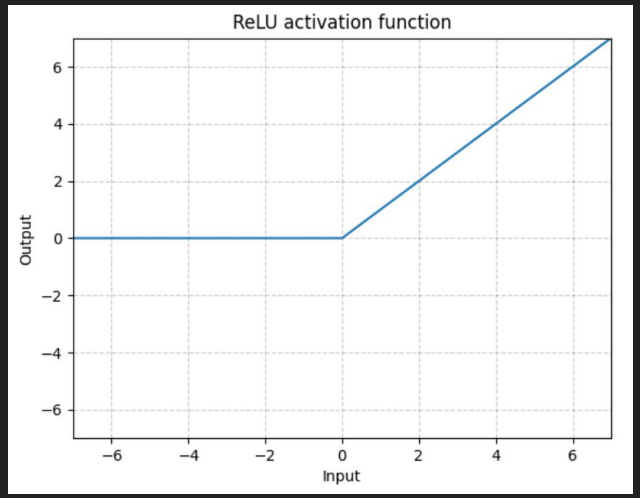
Model Architecture Definition

- Unarguably, the most important step. Defining the Model Architecture of a Deep CNN involves selecting the layers and components that make up the CNN we are training on.
- Since we are using pre-existing functions from the PyTorch library, we shall have a few slides to give brief overviews of the functions and parameters encountered.

- **Sequential:** It is a container through which the arguments of it will be passed in sequence. Within a sequential block, we have :
- Conv2d -
  - This is 2D convolution of the image by a kernel of dimensions specified by kernel size
  - Stride is the step size of the convolution along the axes
  - Padding ensures the output dimensions match our requirement
  - In channels are the number of input image Matrices of form ℝ^(mxn) (For example RGB images are 3 channels wide)
- BatchNorm2d -
  - This works on the principal in this paper
  - The formula applied is
  - γ and β are learnable parameter vectors

$$y = \frac{x - \mathrm{E}[x]}{\sqrt{\mathrm{Var}[x] + \epsilon}} * \gamma + \beta$$

- ReLU -
  - The rectified linear activation function overcomes the vanishing gradient problem, allowing models to learn faster and perform better.
  - The rectified linear activation is the default activation when developing multilayer Perceptron and convolutional neural networks.
- MaxPool2d -
  - **Max pooling** is done to in part to help over-fitting by providing an abstracted form of the representation.
  - As well, it reduces the computational cost by reducing the number of parameters to learn and provides basic translation invariance to the internal representation.



Pooling operation (2x2 filter)

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

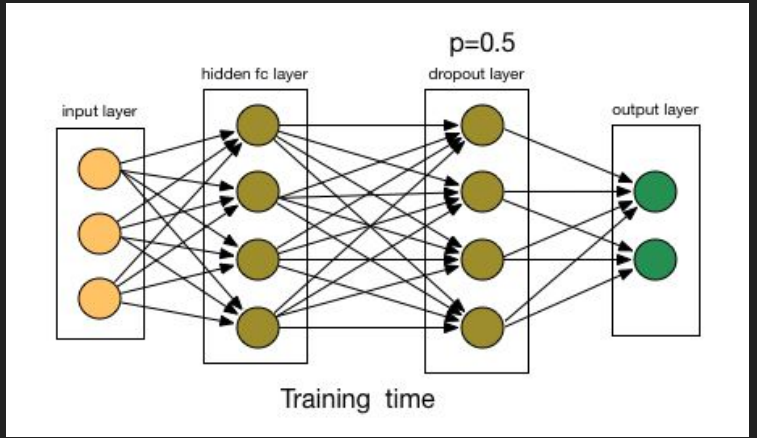
Max pooling

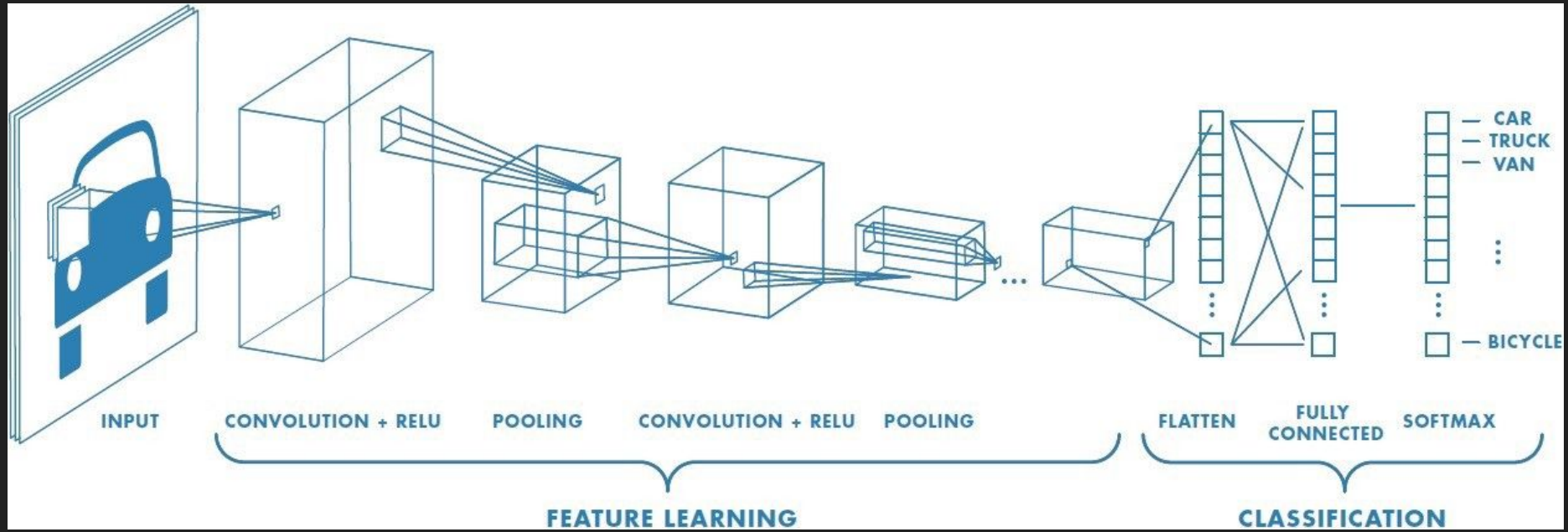
max(v1, v2, v5, v6)	max(v3, v4, v7, v8)
max(v9, v10, v13, v14)	max(v11, v12, v15, v16)

- Linear -
  - Applies a linear transformation to the incoming data
  - It is used to restructure the data into a more workable form by transforming the matrix dimensions

$$y = xA^T + b$$

- Dropout -
  - Dropout is a regularization method that approximates training a large number of neural networks with different architectures in parallel.
  - During training, some number of layer outputs are randomly ignored or “*dropped out*.” This has the effect of making the layer look-like and be treated-like a layer with a different number of nodes and connectivity to the prior layer.



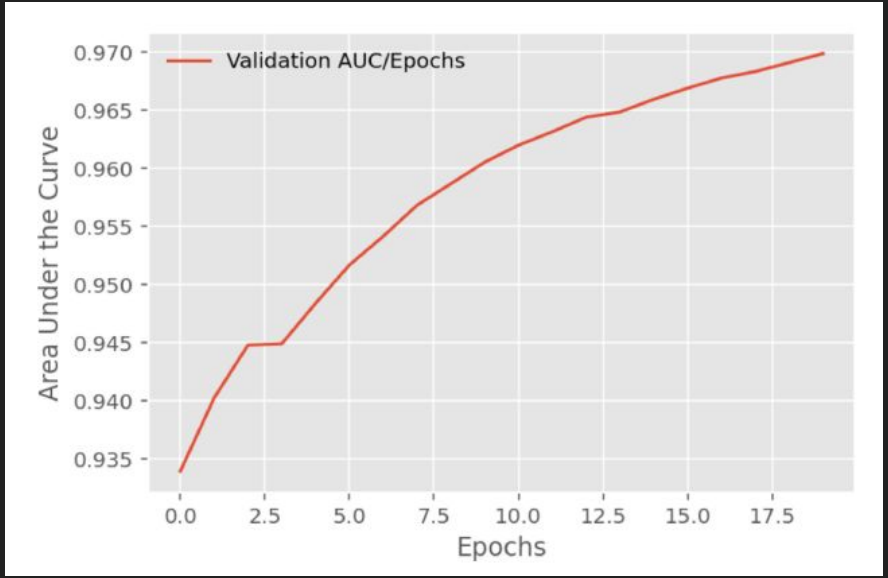


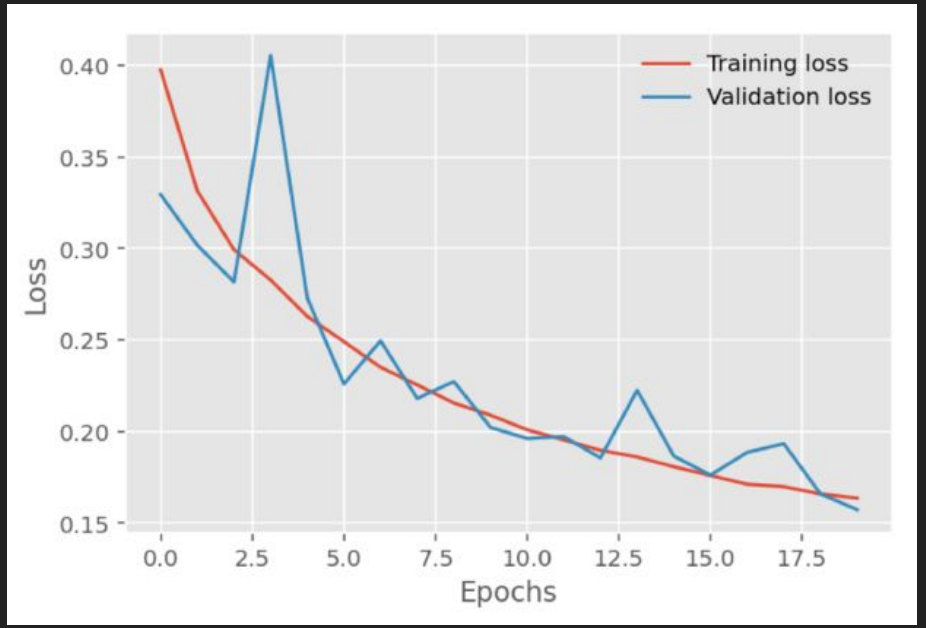
The above schematic is the visual representation of how layers in a CNN work

Our results after training...

- After running the training model for 20 epochs, we observed the following results :

```
Accuracy of model = 93.9375
```





References and Acknowledgments

- https://www.sciencedirect.com/science/article/pii/S2352914819301133
- https://pytorch.org/docs/stable/index.html
- https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html
- https://towardsdatascience.com/histopathological-cancer-detection-with-deep-neural-networks-3399be879671

We would like to thank Prof. Suyash Awate for giving us the opportunity to do this project and in turn get to learn about CNN architecture and delve into Deep Learning

Thank you for your time