Assignment 3, Part A
Convolution Neural Networks - Digit Recognition

- Sarthak Vishnoi, 2016CS10336

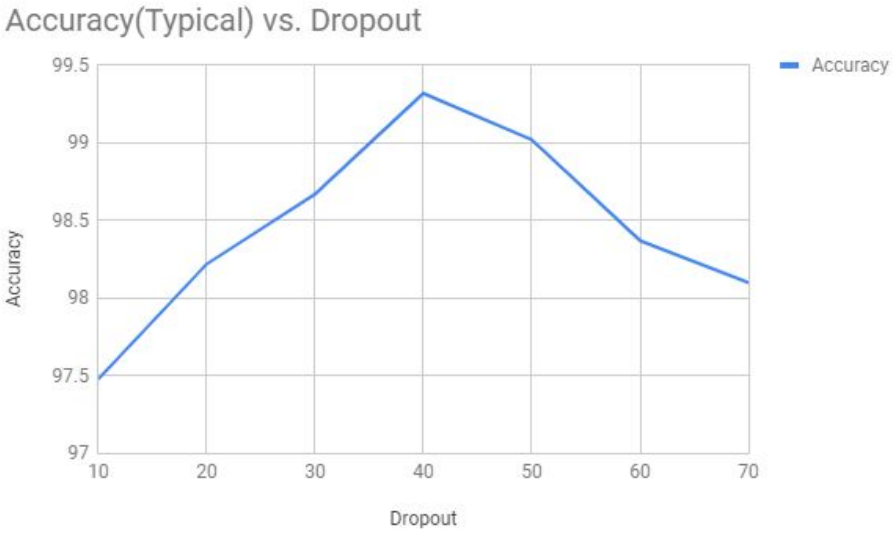
The network which was used was built using the keras library, it contained convolution layers, max-pooling layers, filters and dense layer at the end.

The whole architecture is summarised below:

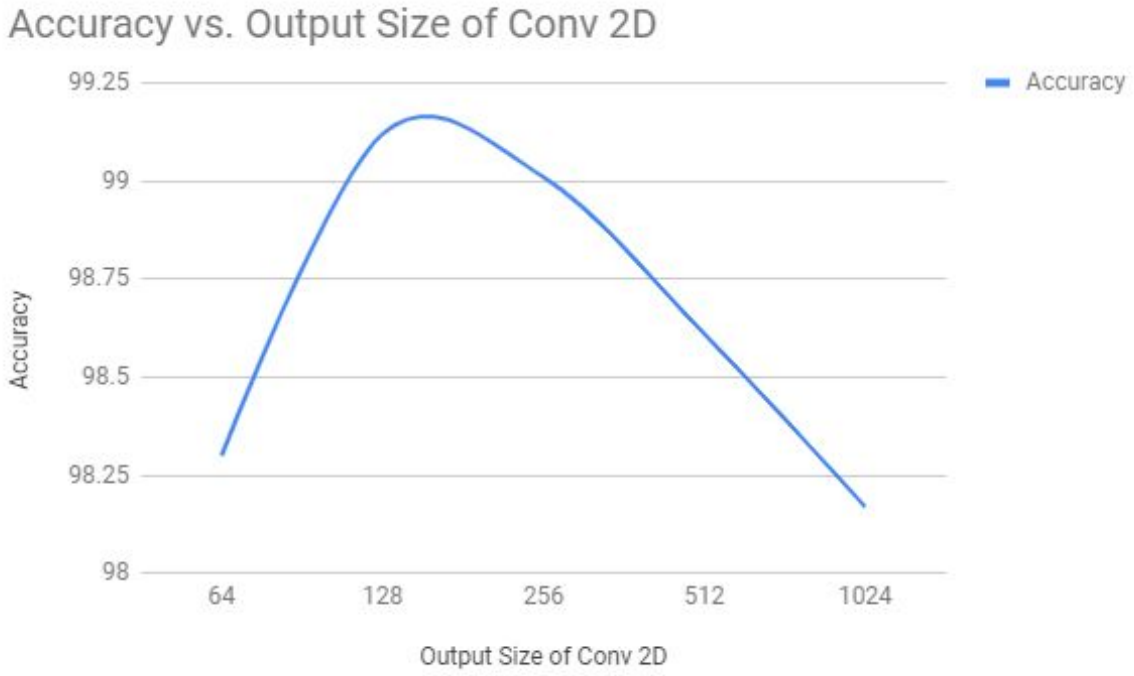
1. 2-D convolution layer of output size 128, and size 3*3 with linear activation function.
2. A leaky ReLU layer with alpha=0.1
3. A 2*2 max-pooling layer, followed by a 40% dropout rate
4. Layers 1-3 repeated again two times
5. A dense layer with tanh activation function followed by a 40% dropout
6. A softmax activation layer with output size equal to number of classes

The total trainable parameters in this model are 564,645.

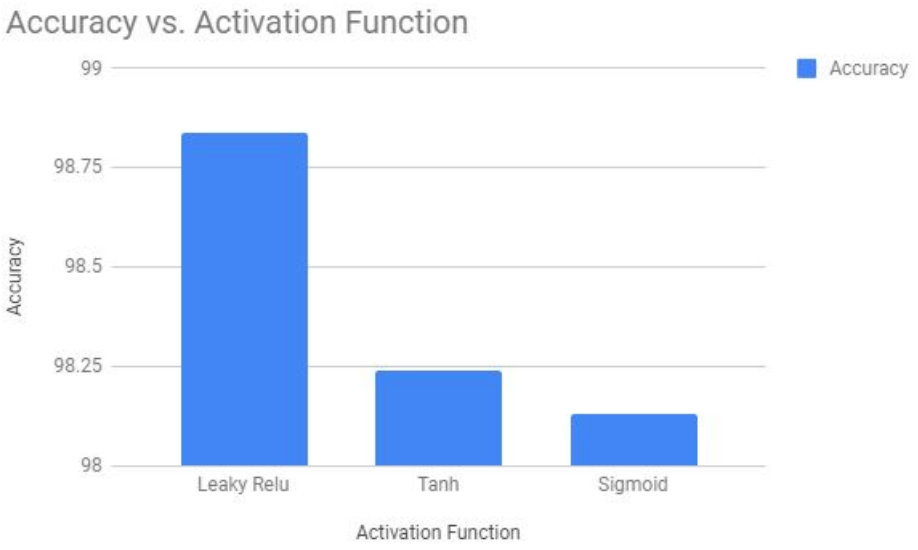
The number of epochs it was run for was 50, and the average time it took to train and predict the data was 30-40 minutes, when using 1 GPU on Google Collab.



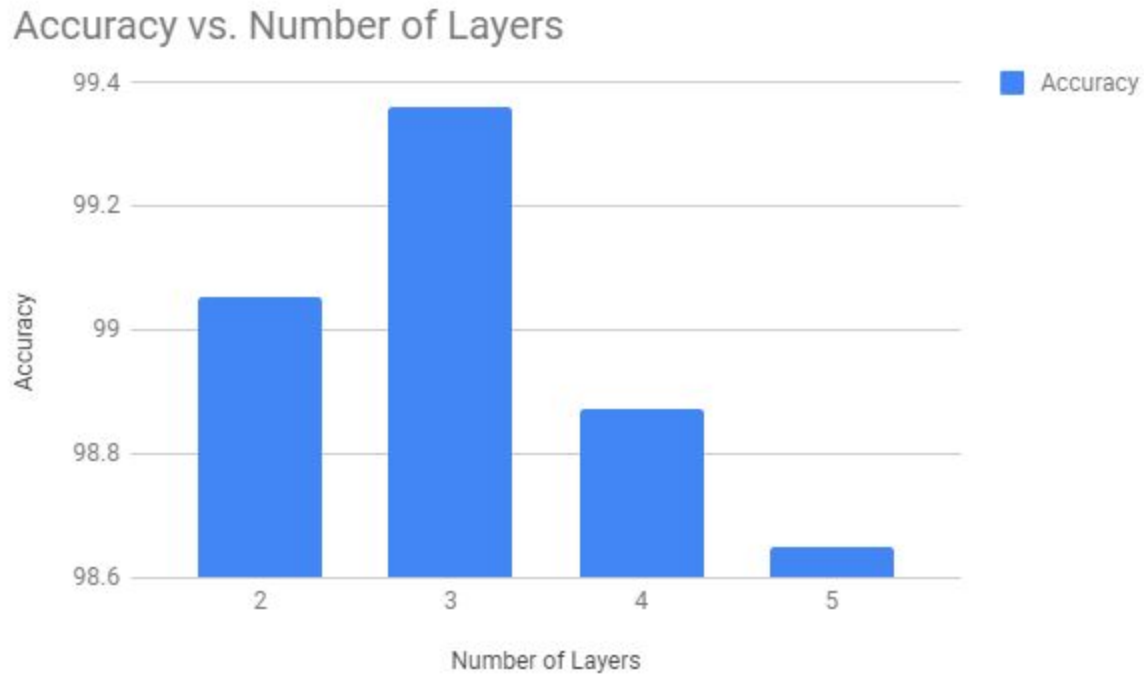
We observe that when we keep the dropout percent as 40, the maximum accuracy is reached. This means out of a total of 564,654 neurons only 338,792 neurons are only activated in a particular epoch. This will help us to avoid overfitting on the training data.



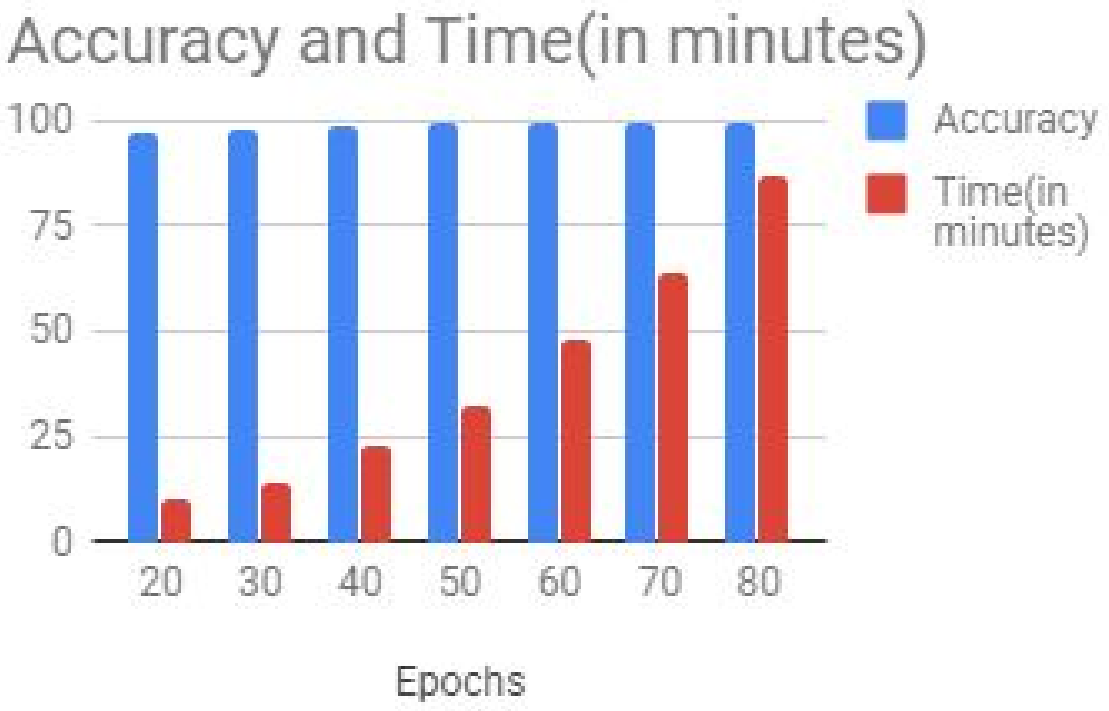
I tried various output sizes and kernel sizes for the convolution layer. The best results (considering the time taken too) were obtained when the output size was 128 and the kernel size was 3*3.



I tried three activation functions for the layers in between. These were Leaky Relu, Tanh and SIgmoid activation functions. The best results were obtained with choosing Leaky ReLU activation function.



The above graph shows the relation between the number of layers (in the final architecture those are 3) versus the accuracy which I got.



The best tradeoff between the time and accuracy with respect to number of epochs run was found at 50 epochs.

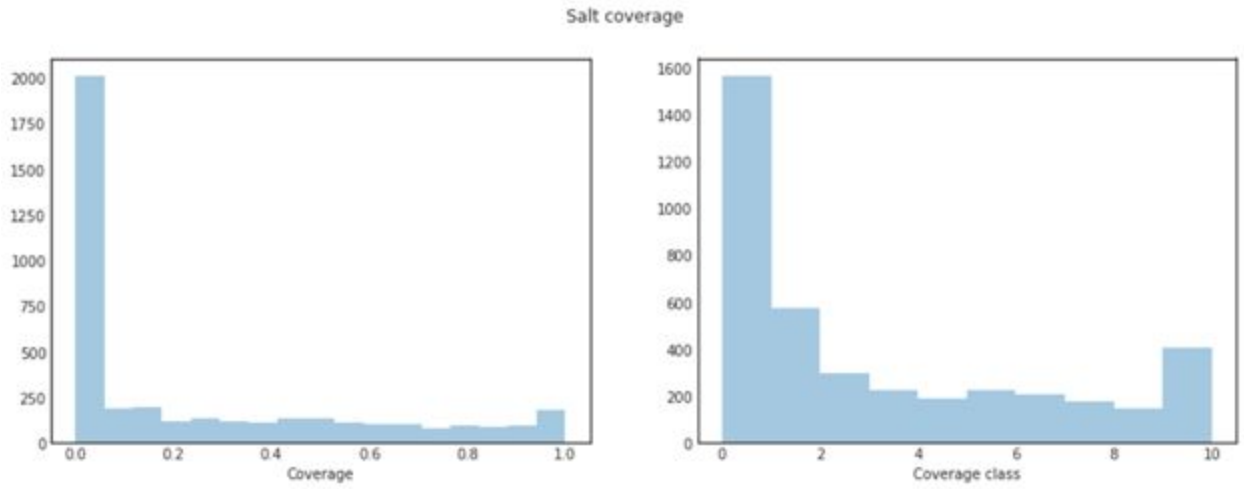
The final accuracy which I achieved on the hidden public dataset is 99.362%.

Assignment 3, Part B
TGS Salt Identification Challenge

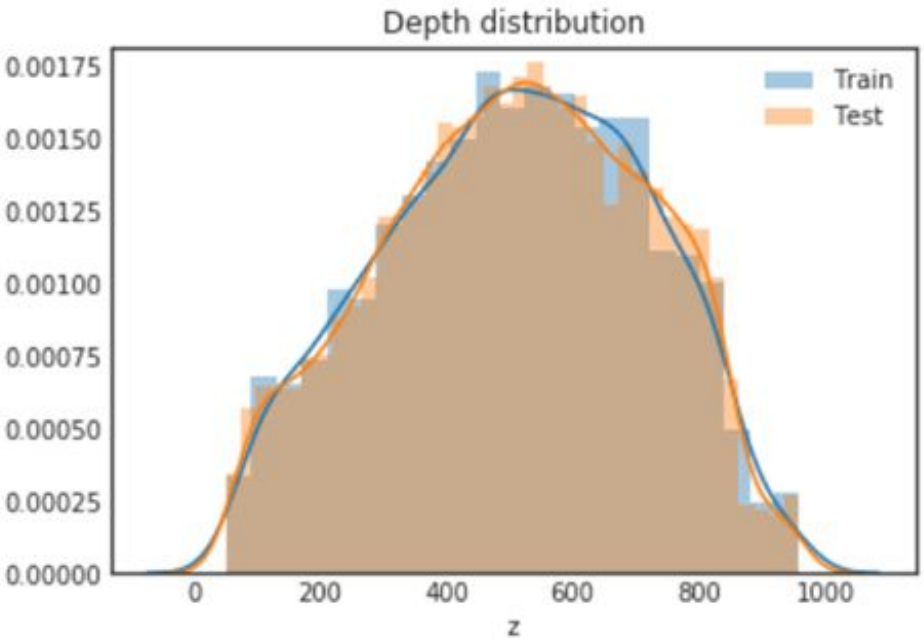
1. Used the method of UNet with simple ResNet blocks for solving the problem statement. The IoU (Intersection over Union) metric was used along with **Lovasz loss** function for evaluation hence for accuracy calculation and early stopping decision taking.
2. Each run of the whole code took anywhere between 6 and 8 hours, so not many trials could be done. The final accuracy that we achieved on the public test dataset was 82.3% which is a lot of improvement from the first run we did in which we could only get an accuracy of 62.7%

Basic Pointers of the code

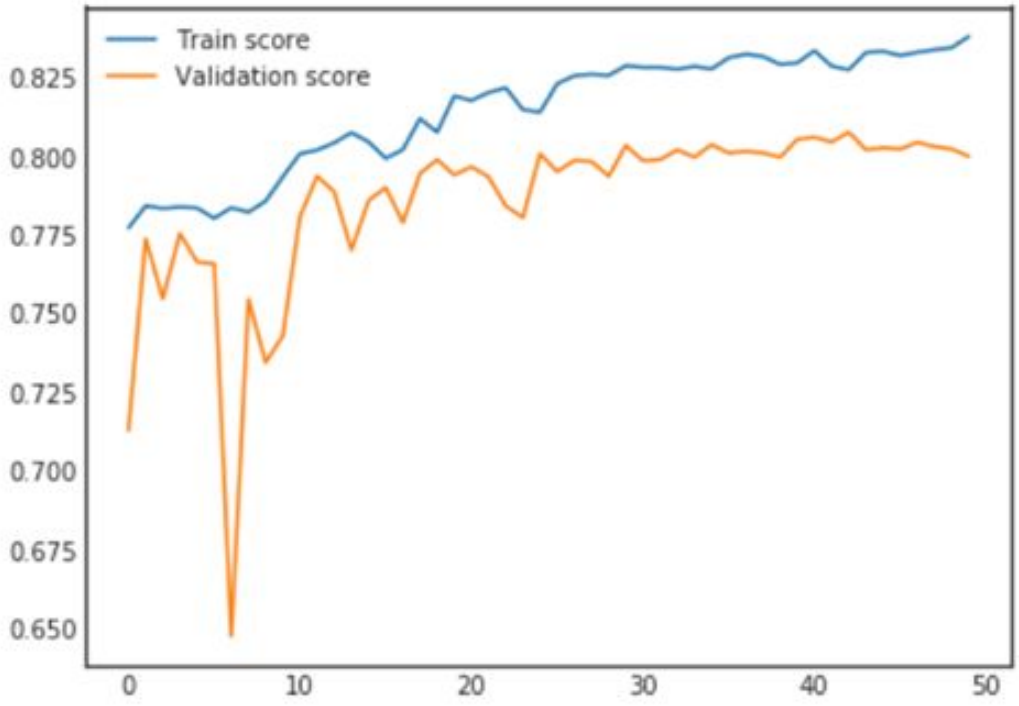
1. Firstly, we divided the images into various classes according to the amount of salt present in each of them. The plot corresponding to this is shown below.



2. Then we divided the images on the basis of their depth

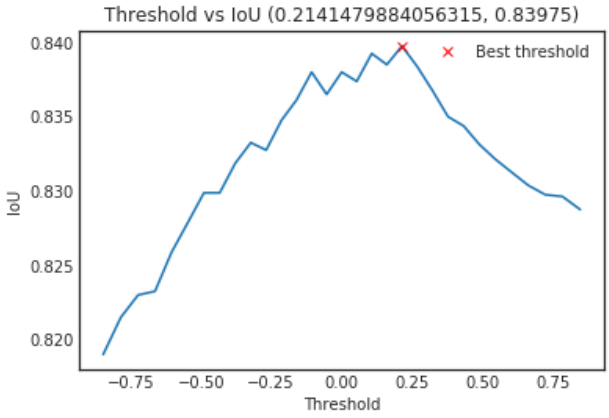


3. Used the IoU (intersection over union) metric for evaluation and hence for early stopping with the validation and training set. Validation set was produced based on coverage classes. The following graph shows the score on training data and validation data v/s number of epochs ran



.

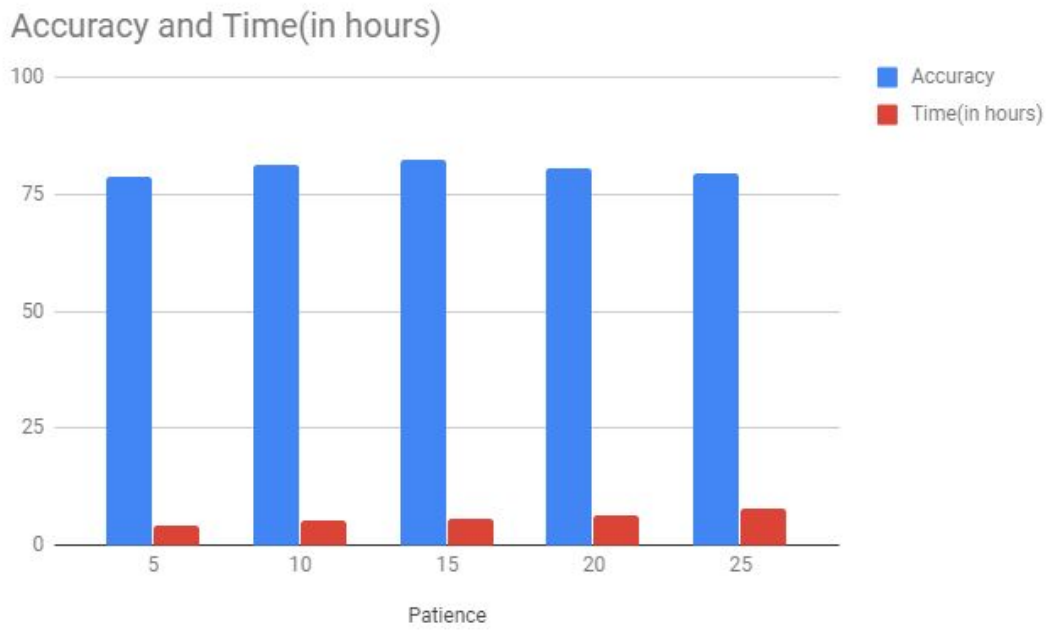
4. We have tried different thresholds for IoU metric. Like -0.75, ..., 0, …., 0.75 chosen the one which gives the highest accuracy on the validation set.



5. Early stopping criteria was set when the IoU metric didn't improve for a continuation of 30 runs. The number of epochs was kept maximum at 100 epochs. The total time taken to run the whole kernel was 5 hours and 50 minutes.

5. Changing the batch size results in the following trend. We chose the best batch size possible which was 128.

6. We kept the patience at 30, ie. if after 30 epochs the IoU metric doesn't improve it'll lead to early stopping. The trade-off in this was between accuracy and time taken to run the code. Smaller patience will lead to poor accuracy and less time, while larger accuracy will lead to higher accuracy or smaller accuracy due to overfitting and large time. The graph for the same is given below.



7. After fine tuning all the parameters mentioned above we saved the output in RLE(run-length encoding) format and saved it as the output CSV file.

Thus, the observations which we got are listed above and hence we came up with the best architecture which gave us an accuracy of 82.3% on the public leaderboard.