**Name: Sai Anish Garapati**

**UIN: 650208577**

# Traffic Sign Image Classification

## Personal views on the field and applications:

The term machine learning is pretty much ubiquitous in the current times though ML research has been going for more than three decades now. It is because of how deeply the applications of ML are playing a role in our daily lives. Though machine learning techniques existed previously, many were not practical at that time. I personally believe this is the golden age of machine learning as it could be applied to almost any field work like medical, autonomous cars, mobiles, computers, stock market, sports predictions, etc. All of these advanced techniques are being implemented practically because of the computational power we have today. With the rate of improvement in computational power each power, the complexity and type of machine learning models especially deep learning learning models and their applications in various fields would be extended even further within this decade. I personally like the idea of implementing machine learning strategies in the field of computer networking, especially Software Defined Networking(SDNs), to enhance network security and better load balancing, which is really important these days, as most of our personal and professional information is on the cloud and it is an area that I wish to learn more about.

## Brief description of learnings in this course:

Though I had taken a couple Machine learning related courses in my undergraduate, they covered only the surface. In this course I learned a lot of mathematics behind every model. I enjoyed the probabilistic view point of looking at ML problems and how they in turn produce the posteriors we already know. I have also learned a lot of unsupervised learning techniques like the Gaussian mixture model, Agglomerative clustering and the extensive math behind SVMs. Rather than just learning about these methods, I was able to apply them practically in the well designed assignment which helped me a lot in understanding the intuition behind the ML models. This course provided me with an idea of what could be possible in this field and has complimented my learnings from my other courses on Neural networks and Information retrieval.

**Project Description:**

There is a lot of focus on Autonomous driving vehicles these days. One of the most important aspects of an autonomous driving vehicle is to identify various kinds of traffic signs in order to provide a safe and secure ride to the people inside the vehicle and also to the people outside. This is a difficult image classification problem in the real world as there is impact from a lot of factors like the weather condition, speed at which a vehicle is moving, condition of the traffic sign, etc.

The objective of this project is to design a machine learning model to classify various Traffic Sign images into their respective categories. This is a Multivariate Classification problem as the image could belong to one of many categories as there are more than two types of traffic signs.

**Project Applications:**

- Traffic sign image classification is already being extensively used in the application of autonomous driving vehicles.
- Can be implemented in already existing automatic braking vehicles to reduce the speed of a vehicle upon detecting an important stop sign.

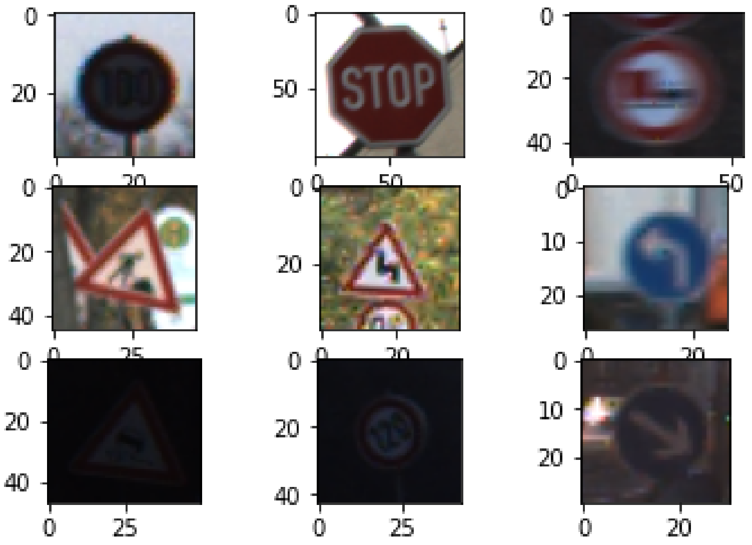
- Can be used for general purpose traffic sign detection in any vehicle to display the type of traffic sign on the car display, in case the driver missed it or is new to the traffic infrastructure of a particular location.

**Dataset:**

The dataset used for this project is the German Traffic Sign Image Recognition Benchmark (GTSRB) from Kaggle. This dataset consists of 39,209 training sample images and 12,630 test sample images. Each image is classified into one of 43 categories.

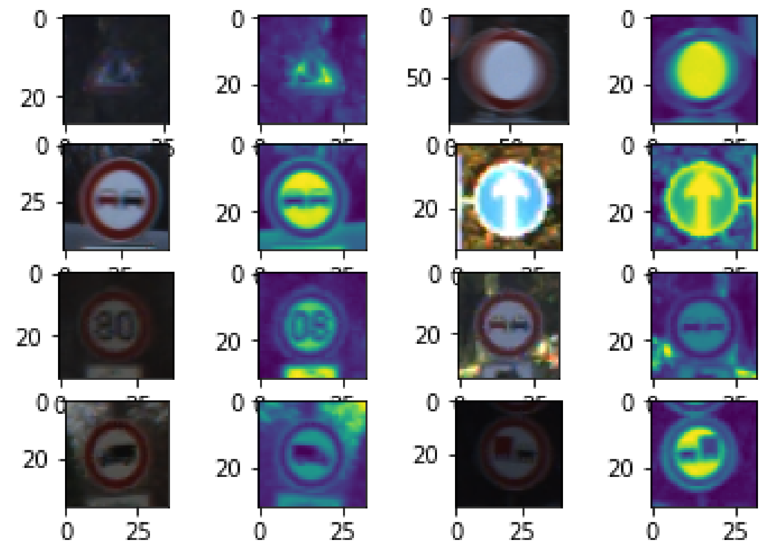
Some of the target classes are Speed Limit (20 km/h), Stop, No Entry, Double curve, Slippery road, Pedestrians, Children crossing, etc.

Some of the sample images from the dataset can be found in the grid below:



**Dataset Preprocessing:**

The GTSRB dataset consists of coloured images of varying resolutions. So, all the images are set to a default resolution of 32 x 32. To make the model adjust well on the testing data, some transformations are tried out on the training data like horizontal flip, grayscale and auto contrast. Some of the processed images can be found in the grid below:



**Methods Implemented:**

The following Machine learning models are implemented on the dataset to compare the accuracy from each of them:

- Support Vector Machine (SVM)
- Random Forest Classifier
- Artificial Neural Network (ANN)
- Convolutional Neural Network (CNN)

**Support Vector Machines (SVM):**

They are a set of supervised machine learning algorithms which can be used for classification. SVM model is applied on the default training data first. The time taken for reading the default data is about 58 seconds. The Support Vector Classification(SVC) classifier from the svm module from the sklearn library is used to build the model on the training data.

The following variations of Support Vector Machines are applied and tried out on the default dataset for this purpose of classification:

| Method | Training Accuracy | Training Time | Test Accuracy | Inference Time |
|---|---|---|---|---|
| **Linear SVM** | 99.89 % | 9.27 minutes | 80.78 % | 5.43 minutes |
| **Polynomial SVM (degree 4)** | 74.34 % | 29.33 minutes | 49.69 % | 9.55 minutes |
| **Linear SVM (5-fold cross validation)** | 94.37 % | 48.3 minutes | 79.83 % | 4.67 minutes |
| **Polynomial SVM (5-fold cross validation)** | 61.51 % | 92 minutes | 46.27 % | 3.85 minutes |

The SVM with Linear Kernel provided very high accuracy on training data but provided less accuracy on the test set. The SVM with Polynomial Kernel with degree 4 performed poorly on both training data and test data. We can clearly see that the Linear Kernel SVM is overfitting on the training data which is why it is failing to generalize on test data, whereas the Polynomial Kernel SVM is underfitting the training data. The main reason for this to be happening could be the size of the training dataset which is 39,309 which could be exposing the data to SVM model too much. To reduce this effect, I applied K-fold cross validation with K value = 5, with both Linear and Polynomial Kernel SVMs. But the results are almost similar with the regular models performing better than the K-fold cross validation models.

In order to reduce the effect of the model on the size of the dataset, I tried augmenting the existing dataset and added a few augmented samples based on the samples from the training data. All the images are also converted to grayscale to reduce the processing time. The augmentations that are applied are RandomAutocontrast(p=0.4), RandomHorizontalFlip(p=1.0). The time taken for reading the augmented data is 53.66 seconds. The size of the training data post augmentation is 58,726 training samples.

The above variations of SVM are tried again and results are displayed as below:

| **Method** | **Training Accuracy** | **Training Time** | **Test Accuracy** | **Inference Time** |
|---|---|---|---|---|
| **Linear SVM** | 95.40 % | 12.26 minutes | 76.73 % | 3.85 minutes |
| **Polynomial SVM (degree 4)** | 73.91 % | 24.58 minutes | 47.67 % | 5.06 minutes |
| **Linear SVM (5-fold cross validation)** | 86.71 % | 49.27 minutes | 75.88 % | 2.9 minutes |

As we can see from the above results, the accuracy has not improved but declined slightly compared to models applied on the training data. This could mean that the performance is not dependent on the size of the training data in this case, or better augmentation of the data is required for better results. Though SVM models produce decent accuracy on the test set, their main downfall is their large training and inference times.

**Random Forest Classifier:**

A random forest fits a number of decision trees on various sub-samples of the dataset and averages on all of them to improve accuracy and to avoid overfitting. RandomForestClassifier from sklearn.ensemble is used for this. Similar to SVM models above, random forest classifiers are first tried on the default dataset.

| **Number of decision trees** | **Training Accuracy** | **Training Time** | **Test Accuracy** | **Inference Time** |
|---|---|---|---|---|
| 100 | 100 % | 3 minutes | 74.92 % | 1.01 seconds |
| 200 | 100 % | 5.36 minutes | 71.25 % | 1.73 seconds |
| 300 | 100 % | 8.33 minutes | 71.63 % | 2.90 seconds |

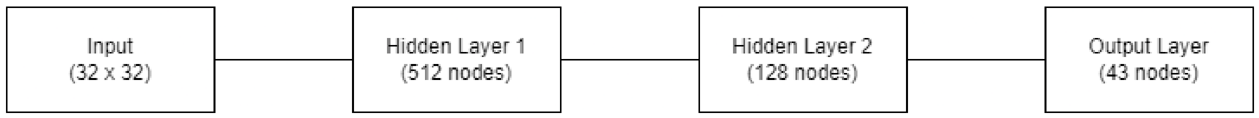
To check if there is a dependency on the size of the training data, similar augmentations are applied on the training data as in SVMs, and the models are applied again.

| **Number of decision trees** | **Training Accuracy** | **Training Time** | **Test Accuracy** | **Inference Time** |
|---|---|---|---|---|
| 100 | 100 % | 2.58 minutes | 70.11 % | 0.9 seconds |
| 200 | 100 % | 5.73 minutes | 71.25 % | 1.9 seconds |
| 300 | 100 % | 8.56 minutes | 71.63 % | 2.75 seconds |

Though the random forest classifier models have accuracy similar to SVM models, random forest models have better training time and very less inference times which are a major advantage for these classifiers.

**Artificial Neural Networks (ANN):**

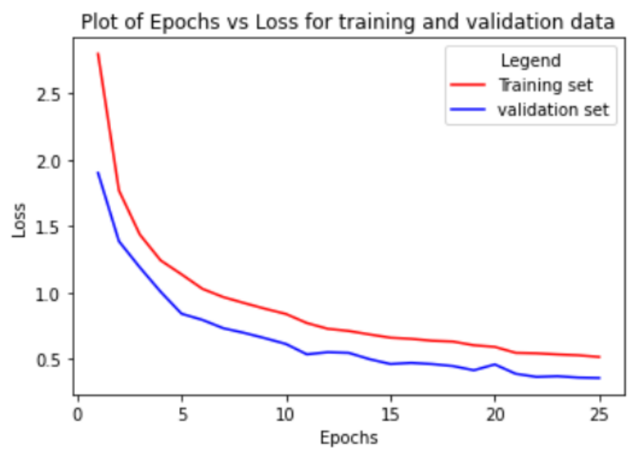
ANN are composed of various layers which have nodes interconnected with the nodes of the layer and the information flows between them. For the goal of this project, I went with a 3-layered fully connected neural network for classification. The image of the architecture chosen is given below:

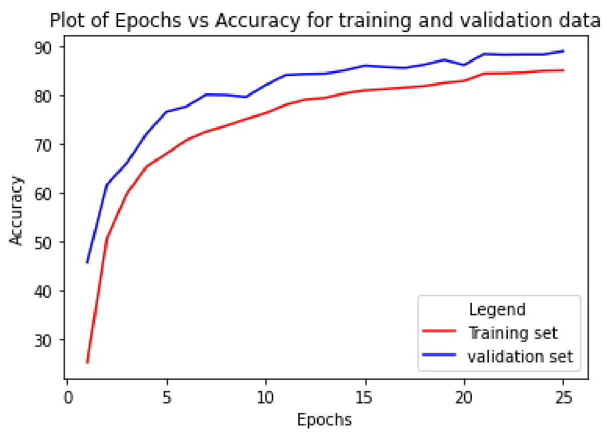


Pytorch library in python is used to code the neural networks part. Dataloader is used to load the data from folders and torchvision library is used to apply transformations on the images for the purpose of training. The batch size and shuffle parameters are mentioned in the Dataloader function.

Rectified Linear Unit(ReLU) activation function is used, as it is well known for easy training and achieving high performance compared to other activation functions. The Categorical cross entropy loss function is used as the problem is a multiclass classification problem. This loss function applies softmax on the output to bring it on probability distribution and then calculates the loss by applying logarithm.

Adam optimizer is used and stepLR scheduler is used with step_size 10 and gamma value 0.8. The results of loss and accuracy for neural network by tweaking the hyperparameters of the network can be found below:



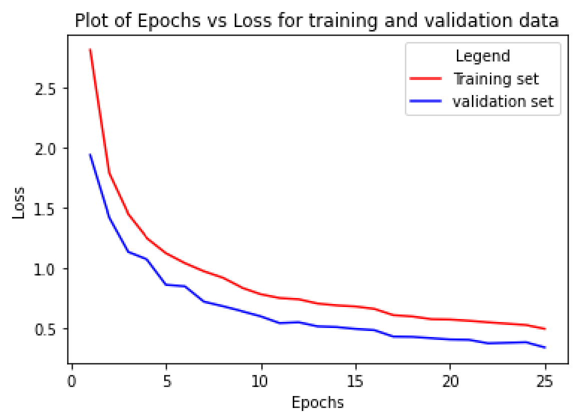


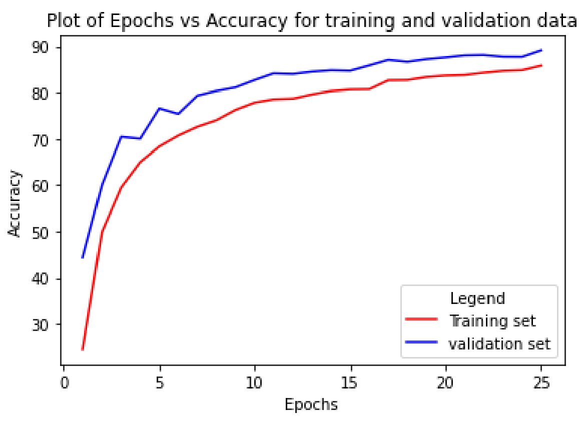
Learning rate: 0.001, Batch size: 128

Training Accuracy: 85%, Validation Accuracy: 88.92%

Test Accuracy: 81.77%

We can see that the convergence is a little slow. So for faster convergence, I tweaked the hyperparameters and set learning rate to 0.01 and batch size for training to 256.





Learning Rate: 0.01, Batch size: 256

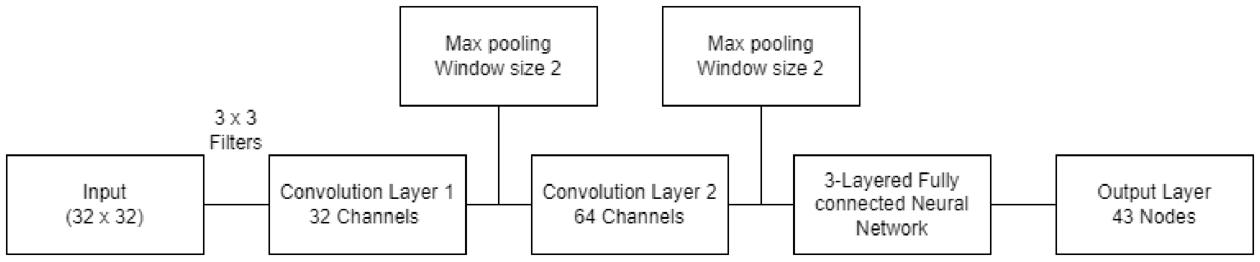
Training Accuracy: 85.82%, Validation Accuracy: 89.12%

Test Accuracy: 82.51%

We can see that the algorithm might perform better if run for some more iterations, as it could catch up with the validation accuracy. But we will see that for the given iterations, performance is less when compared to a more sophisticated network like CNN.

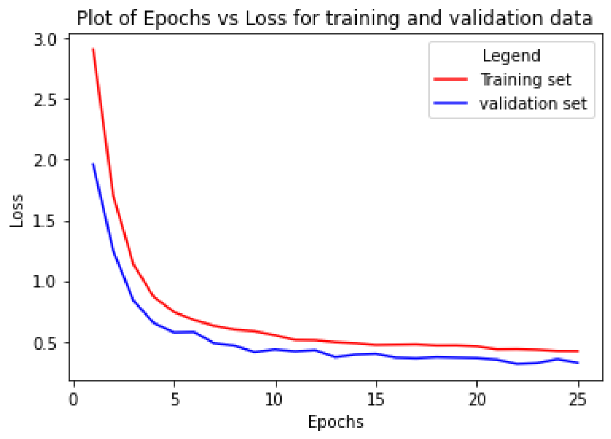
## <u>Convolutional Neural Networks (CNN):</u>

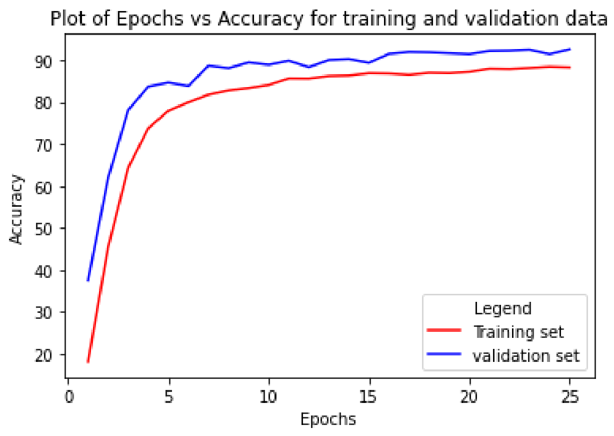
CNN is a type of artificial neural network especially used for image classification. They are used to build more complex models by extracting primitive features and in turn advanced features. They also use a principle of weight sharing to reduce the number of trainable parameters. The CNN network used for this project can be found below.



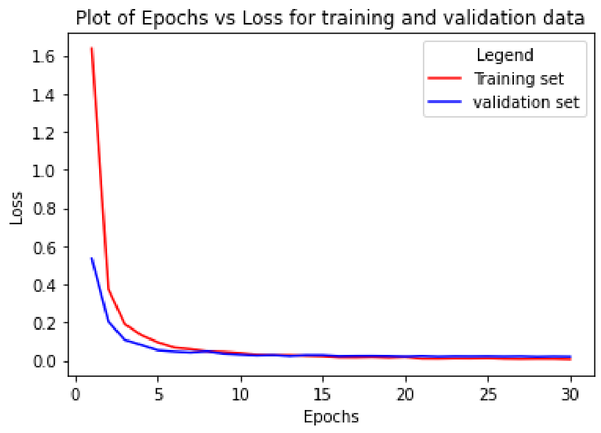
Two convolution layers with the above setup are used. Similar to ANN, pytorch library is used to code the network. The same Rectified Linear Unit(ReLU) activation function and Categorical cross entropy loss function are used.

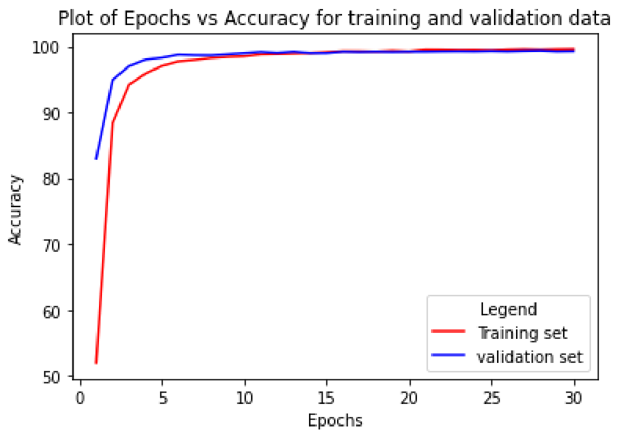
3 x 3 filters are used to extract feature maps from the images. Two convolution layers are used with 32 channels and 64 channels respectively. And after each convolution layer, a max pooling layer is used with size 2 x 2 and a stride parameter equal to 1. I have used two dropout layers which drop neurons at random with probabilities 0.25. After each convolution layer The results of loss and accuracy for neural network by tweaking the hyperparameters of the network can be found below:





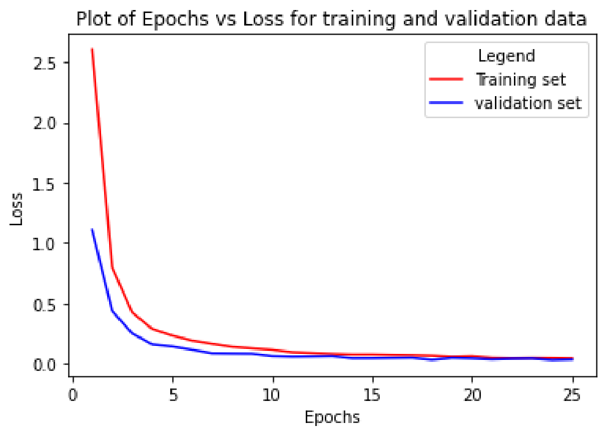
Learning Rate: 0.001, Batch size: 128, weight decay: 0.01
Training Accuracy: 88.30%, Validation Accuracy: 92.59%
Test Accuracy: 85.26%

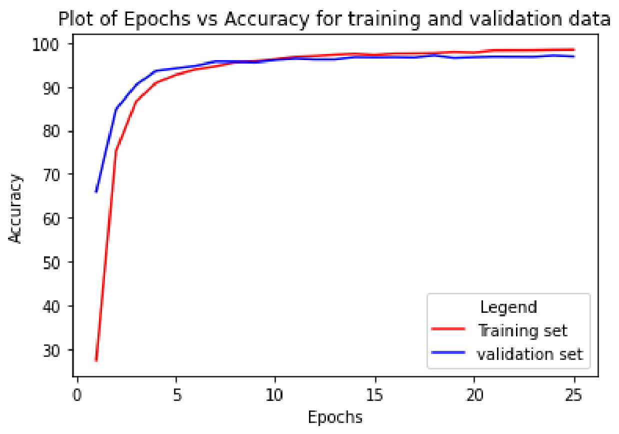




Learning Rate: 0.001, Batch size: 100, weight decay: 0
Training Accuracy: 99.60%, Validation Accuracy: 99.29%
Test Accuracy: 96.025%

Having a weight decay could produce similar or slightly better results. But they would have to be run for longer iterations.





Learning Rate: 0.001, Batch size: 128, weight decay: 0, 3 convolution layers
Training Accuracy: 98.46%, Validation Accuracy: 96.90%
Test Accuracy: 94.56%

To see if there are any performance gains in having more convolution layers, I added a third convolution layer. The graph for this case can be seen above. The performance is slightly similar to what we got from using two layers. The model is fitting slightly less on the training data than the model with two layers. This could provide a more generalized model but could be required to run for much longer iterations.

**Comparison between all the methods used:**

| Method | Test Accuracy | Training Time | Inference time |
|---|---|---|---|
| Support Vector Machines | 80.78 % | 9.3 mins | 5.4 mins |
| Random Forest Classifier | 74.92 % | 3 mins | 1 seconds |
| Artificial Neural Network | 82.51 % | 25 mins | 36.5 seconds |

| | | | |
|---|---|---|---|
| Convolutional Neural Network | 96.025 % | 35 mins | 32.3 seconds |

From the above comparison we can see that CNN is the best model as it should be for image classification problems. But the non-deep learning models also did a decent job. Though SVM has better accuracy than Random Forest Classifier, its downfall is its long training and inference times especially the polynomial SVM and Cross-validation SVM. Random forest classifier has the least training time out of all and has very fast inference time, which could make it a decent practical option, given that some feature extraction is applied to improve the accuracy slights.

**Future work and improvements:**

Some of the improvements that could be done to improve the existing implementation of the project are:

- Study each individual image, especially the ones that are misclassified and do better Data augmentation to add artificial training samples. Study the distribution of images and augment the dataset so as to make the count of all the samples belonging to different classes same.
- For Support Vector Machines and Random Forest classifiers we can do feature extraction using CNNs and then pass the extracted features to the classifiers. This may increase the training time due to additional layers, but it would reduce the inference time.
- Try out better regularization and also apply state of the art CNN models like ResNet to see how they perform on the same dataset.

I would have liked to learn briefly about the applications ML is used in, like Computer vision and NLP which would have helped me in choosing an area of interest to pursue or choosing future courses.