Visual Recognition

Assignment-3

Object Classification and Object Detection

Team Members

Kanigri Sri Sai Venkata Naveen – IMT2019039

Kartikeya Siva Hitesh Kasturi – IMT2019041

Kopparapu Sai Krishna – IMT2019045

Assignment 3a:

Process Overview:

- Loaded the CIFAR-10 dataset (train, validation, test) from torchvision library.
- Created a neural network with 2 convolution layers and 3 fully connected layers.
    - Layer-1: `nn.Conv2d(3, 32, kernel_size=3, padding=1)`
    - Layer-2: `nn.Conv2d(32, 64, kernel_size=3, stride=1, padding=1)`
    - `MaxPooling: nn.MaxPool2d(2, 2)`
    - Layer-3: nn.Linear(64*64*16, 1024)
    - Layer-4: nn.Linear(1024, 512)
    - Layer-5: nn.Linear(512, 10)
- Wrote forward, training, validation, fit and evaluate functions from the notebook shared by TA in the pytorch tutorial.
- No. of epochs used in fit function is 20.
- We used Adam as the optimization function for adaptive learning rates.

Observations:

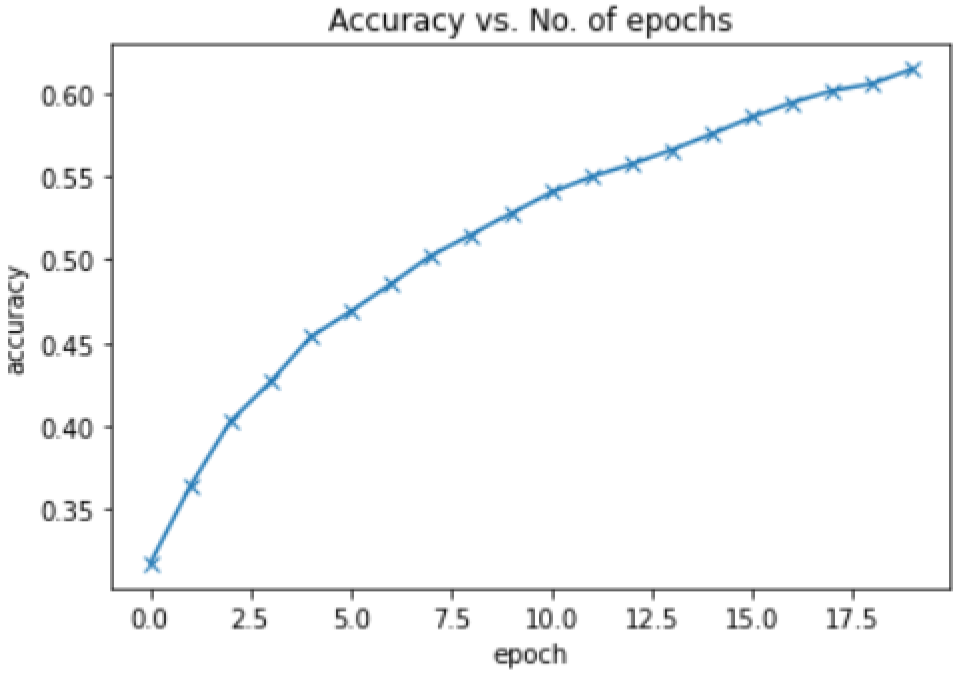
	Activation Function	**Training time**	**Classification performance (Accuracy)**
Adam	**ReLU**	6min 20sec	0.6815664768218994
	Sigmoid	6min 24sec	0.6247033476829529
	Tanh	6min 22sec	0.6414161324501038
SGD	**ReLU**	5min 28sec	0.6200553774833679
	Sigmoid	5min 32sec	0.2772943079471588
	Tanh	5min 33sec	0.5854430198669434
Momentum=0.9, SGD	**ReLU**	5min 55sec	0.7020371556282043
	Sigmoid	6min 0sec	0.5089003443717957
	Tanh	6min 0sec	0.6801819801330566

- The training time is calculated only using the cell which contains `history = fit(num_epochs, lr, model, train_loader, val_loader, opt_func, momentum)` because the time taken to run all the other cells is nearly zero.

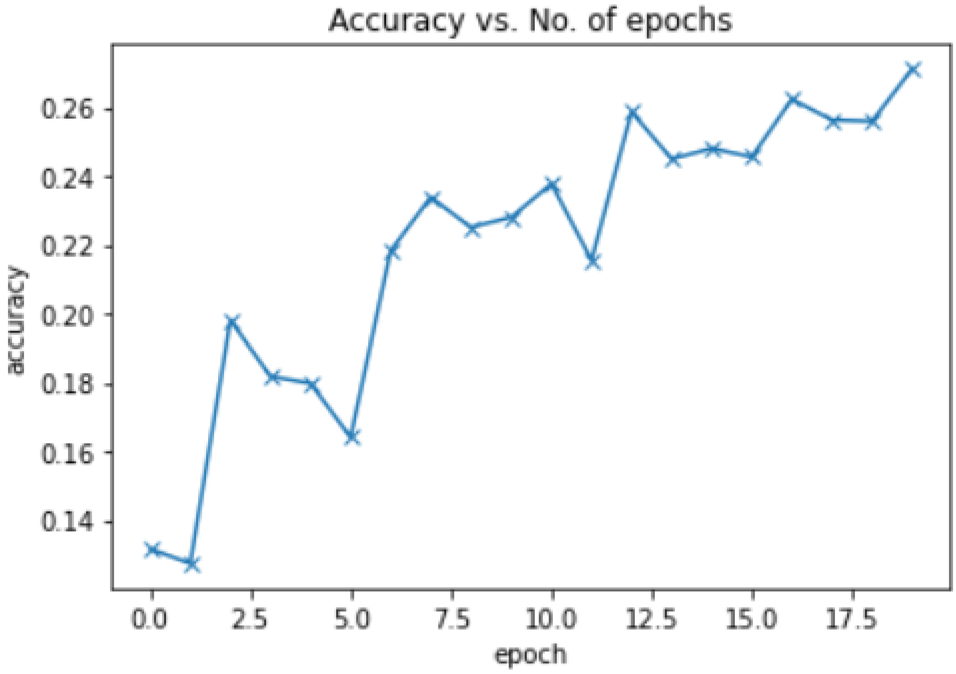
- We recommend the architecture whose optimization function is SGD, momentum = 0.9 with ReLU as its activation function because it has the highest accuracy and relatively low training time.

With SGD:

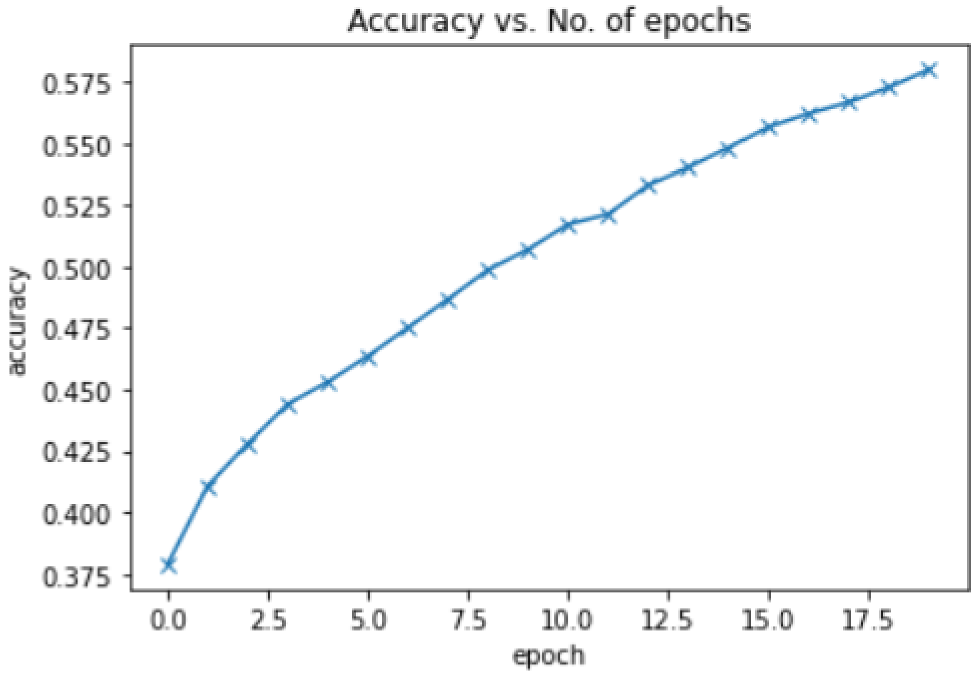
1) Relu as activation function:



2) Sigmoid as activation function:

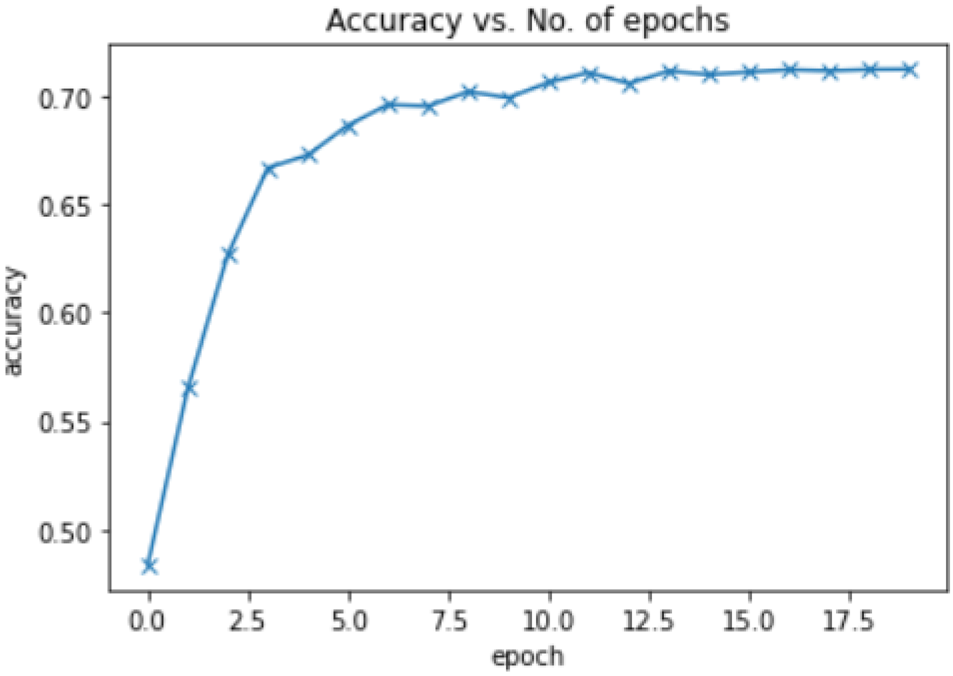


3) Tanh as activation function:

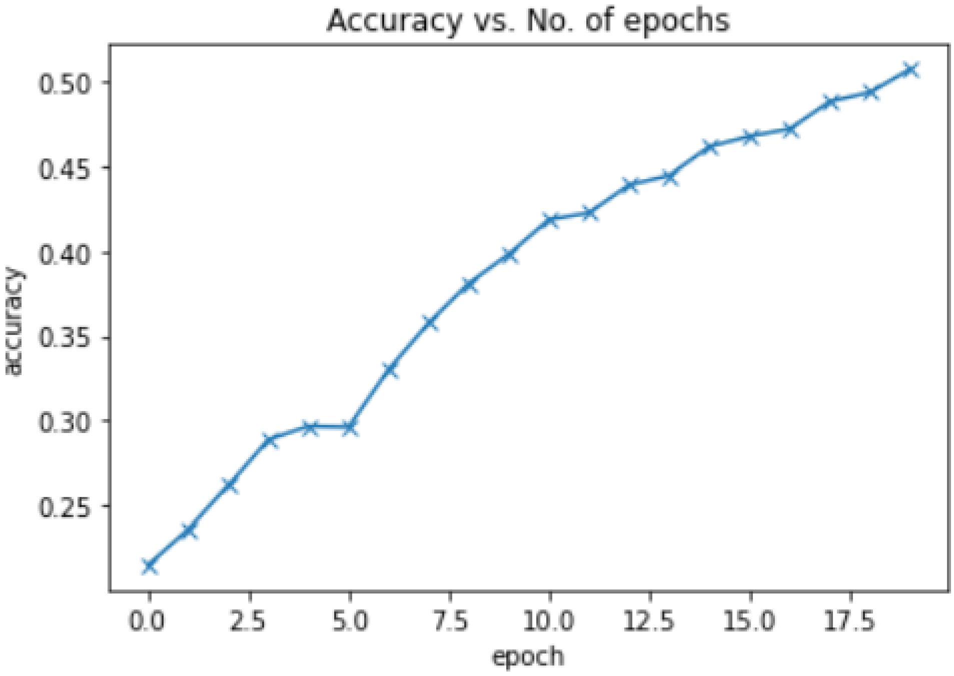


With Momentum, SGD:

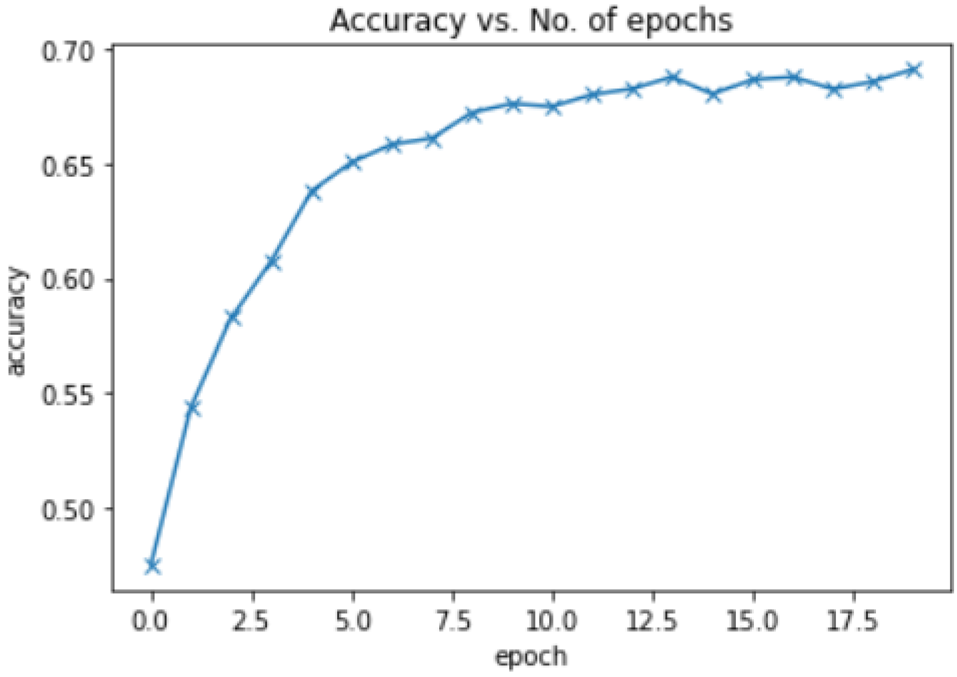
1) Relu as activation function:



2) Sigmoid as activation function:

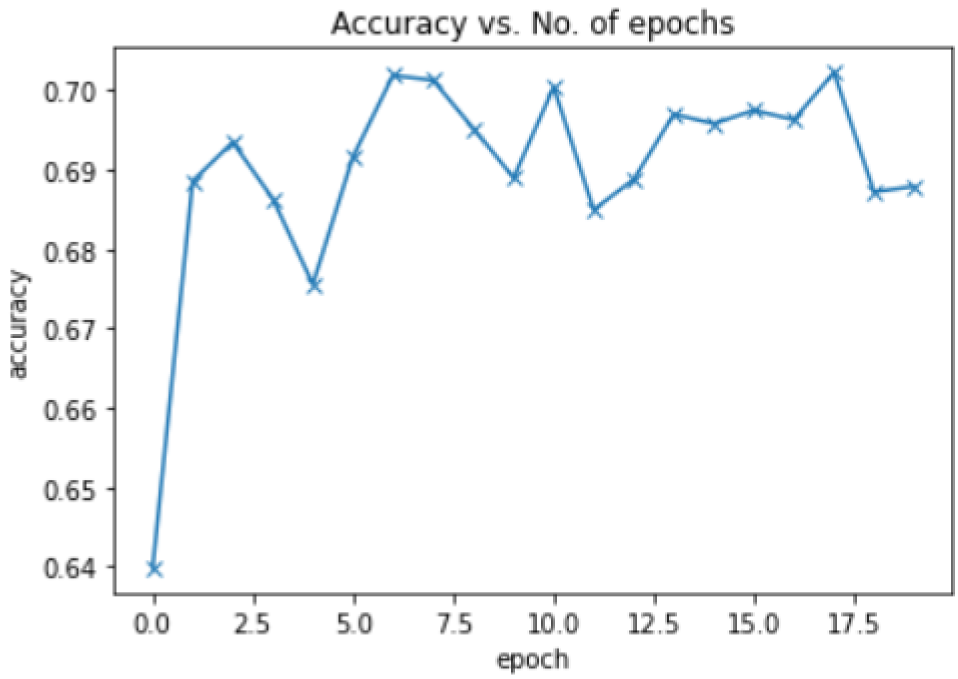


3) Tanh as activation function:

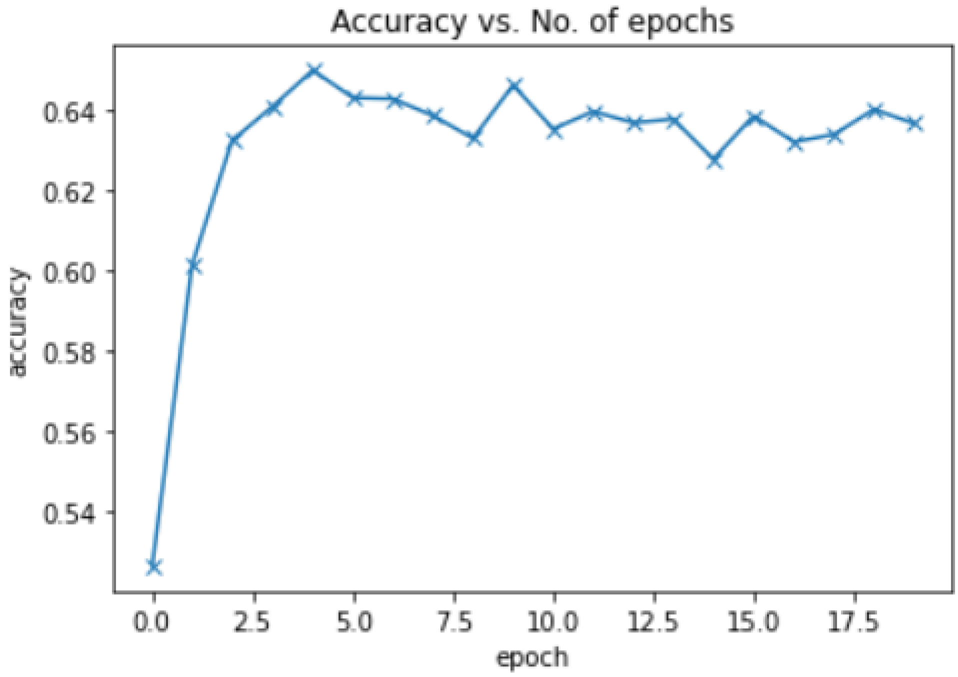


With Adam:

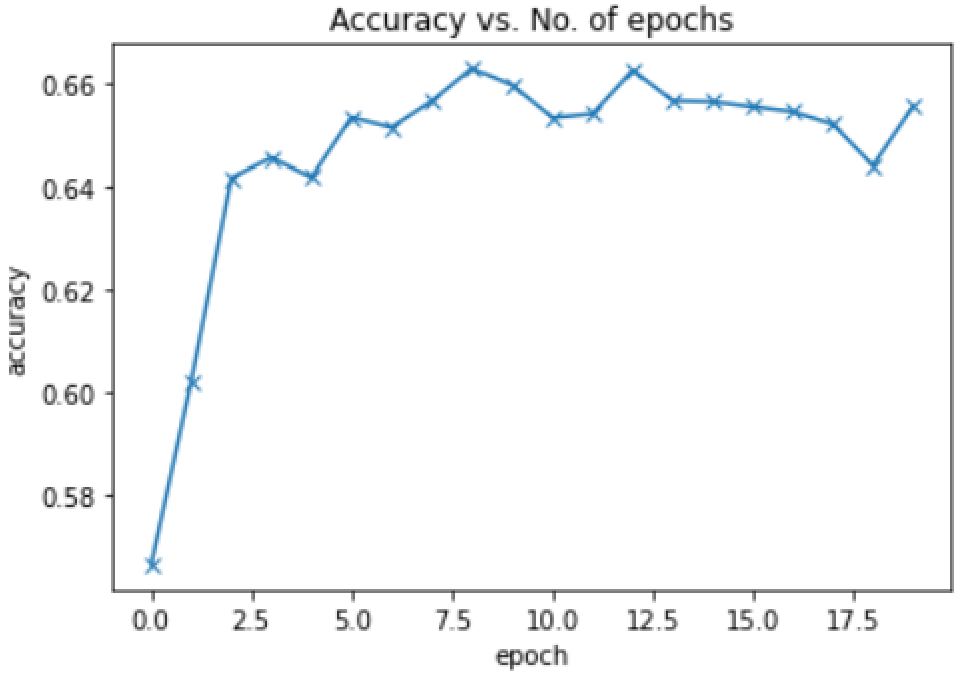
1) Relu as activation function:



2) Sigmoid as activation function:



3) Tanh as activation function:



Assignment 3b:

30 musical instruments is the dataset we chose.

Process Overview:

- Extracted the train and test data from the instruments.csv file.
- Transformed all the images to 224*224 pixels as the AlexNet architecture requires the input to be in that format.
- The code used to import the alexnet architecture is
```
alexnet = models.alexnet(pretrained = True)
alexnet.eval()
```
- Extracted the features from the AlexNet architecture for each image in the dataset.
- The features are then passed through models.
- The models we used are SVM and logistic regression.
- The shape of train features is (4674, 1000) where each feature is a 1000 column vector.

Observations for musical instruments dataset:

- SVM:
  - svm.SVC(C=0.0005,kernel='linear', class_weight='balanced', gamma='scale')
  - Accuracy = 0.9862068965517241
- Logistic Regression:
  - LogisticRegression(max_iter=1000)
  - Accuracy = 0.9793103448275862

Observations for Bikes vs Horses:

- The process for bikes vs horses dataset is similar to the process followed for musical instruments dataset.
- SVM:
  - svm.SVC(C=0.0005,kernel='linear', class_weight='balanced', gamma='scale')
  - Accuracy = 1.0
- Logistic Regression:
  - LogisticRegression(max_iter=1000)
  - Accuracy = 1.0

Assignment 3c:

Overview of the Model Used:

YOLO v5 – You Look Only Once. It is an object detection algorithm that divides images into a grid system. Each cell in the grid is responsible for detecting objects within itself.

It is a single state object detector. It has three parts. Model Backbone, Model Neck, Model Head

The dataset has 152 images which are divided into train (70%), test (10%), validation (20%).

Overview of the process:

1. Annotate the images using Roboflow and split the dataset into test, train, validation. The site provides a link to access the dataset which can be used in the python code.
2. The link to the dataset is placed in the code given in the video
3. The Addresses of the data files (train, test) are stored in the data.yaml.
4. Required dependencies and the yolov5 code is taken from github repo.
5. Yolov5s (small version of yolov5) is the model used.
6. Hyper parameters chosen –
   a. No. of iterations – 150
   b. No. of batches – 16
   c. Image size – 416
7. The trained model is used to predict some images. Detections that have confidence above 0.1 are shown in the output images.

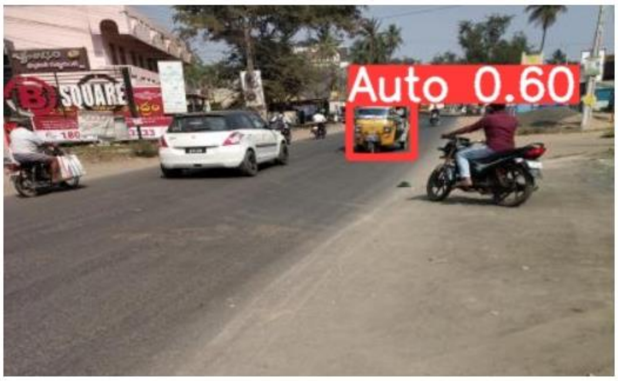
Accuracies:

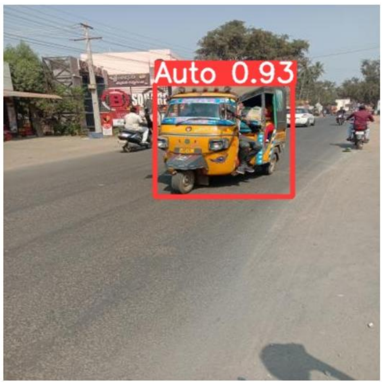
mAP score – mAP stands for mean Average Precision score. A series of Precision and recall curves are plotted with the Intersection over Union threshold values set at various levels.

1. Precision – 0.911
2. Recall – 0.891
3. mAP@0.5 – 0.913
4. mAp@0.5: .95 – 0.603

Results and Observations:

1. Accuracies of the model are mentioned above.
2. Some of the images are detected with very high accuracy. Whereas some auto images are not even detected.
3. Observed Images:





Auto 0.92

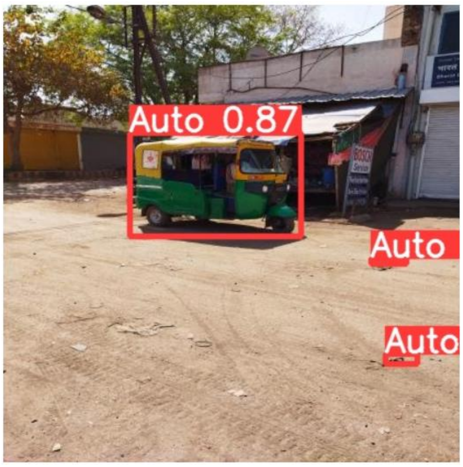
Auto
Auto 0.92

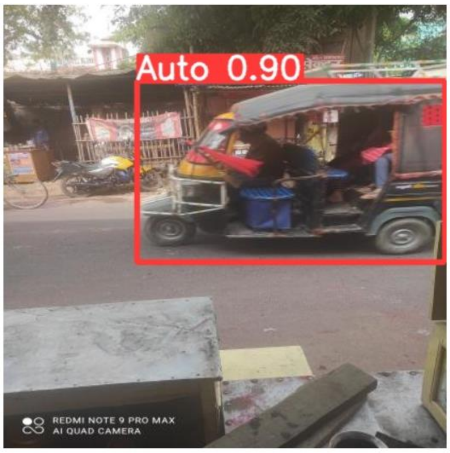
Auto 0.92

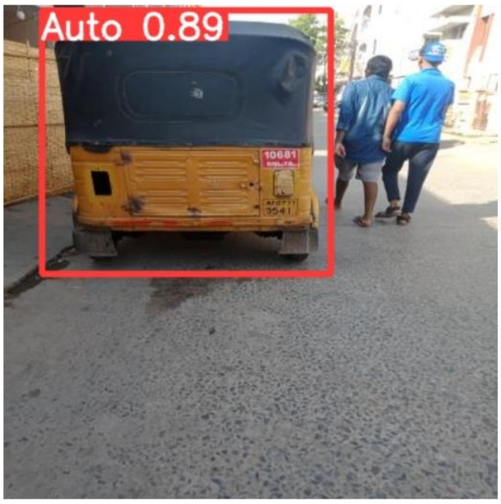
Auto 0.89

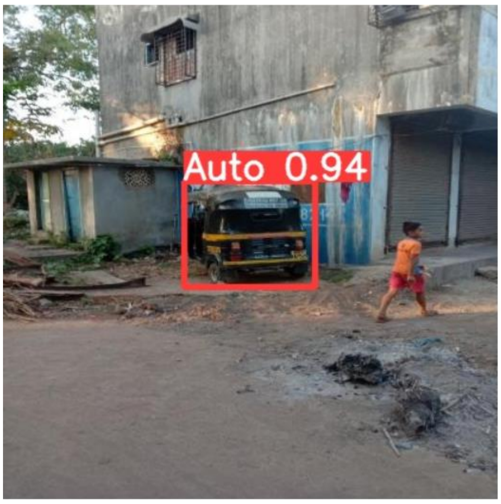
Auto 0.90

Auto 0.92

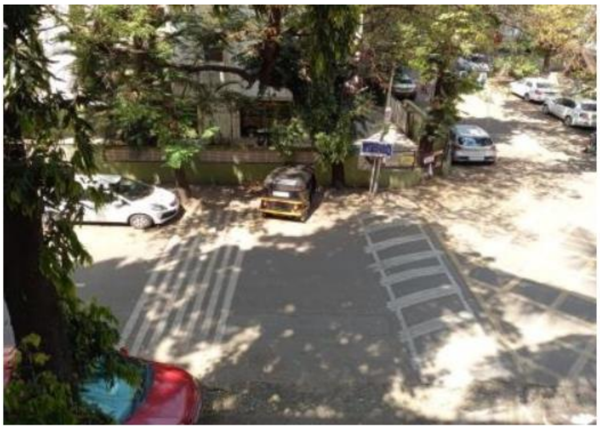


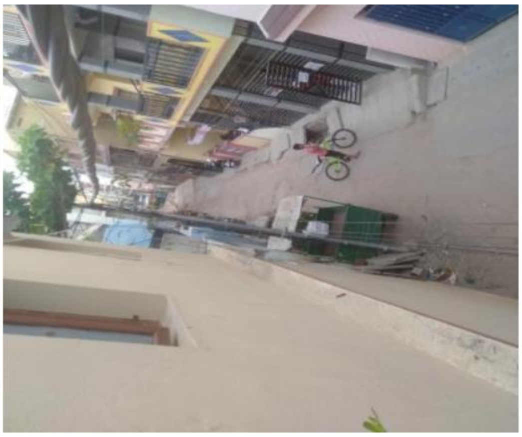






4. Auto images that are not detected:

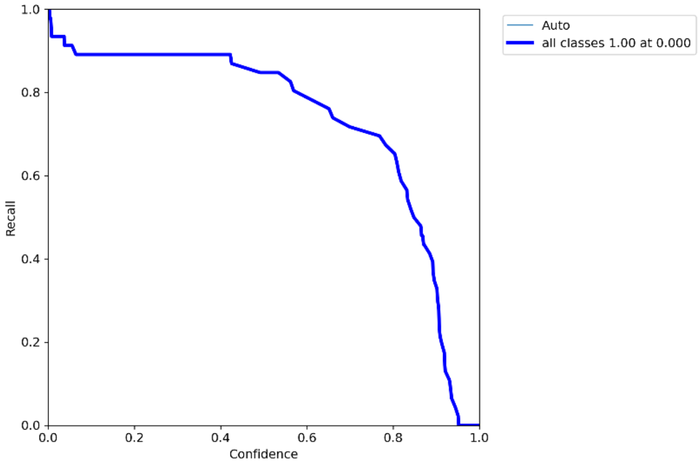




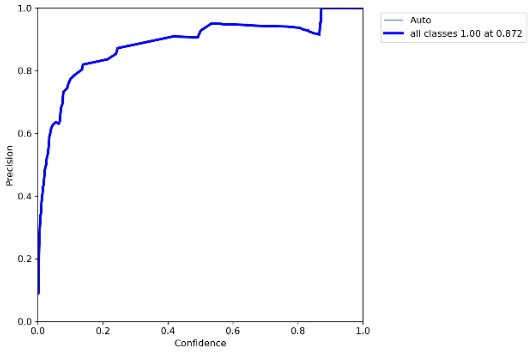
5. These images are not detected maybe because the training set may not have images of these views or orientations. Autos in these images maybe detected better if there are more number of images with higher variations.

Accuracy Curves:

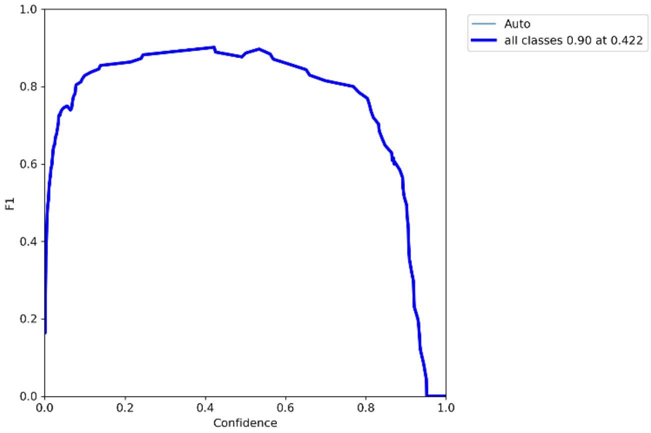
1. R-curve:



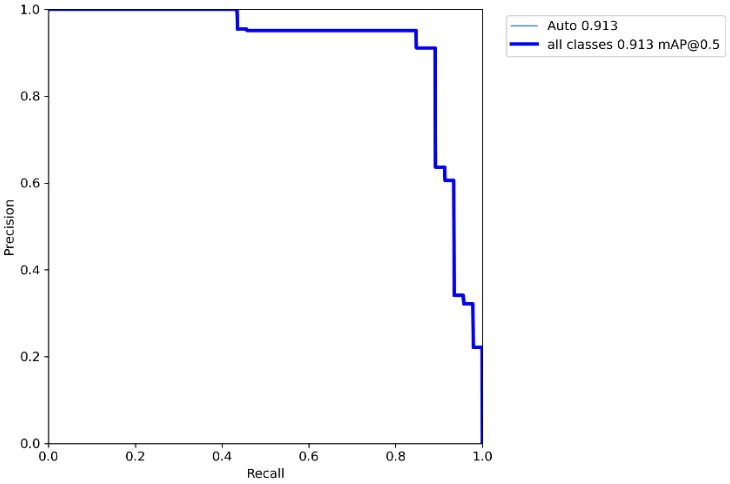
2. P-curve:



3. F1-curve:



4. PR curve:



References Used:

Pytorch tutorial MNIST code.

https://colab.research.google.com/drive/1gDZ2xcTOgR39tGGs-EZ6i3RTs16wmzZQ

https://www.youtube.com/watch?v=MdF6x6ZmLAY