

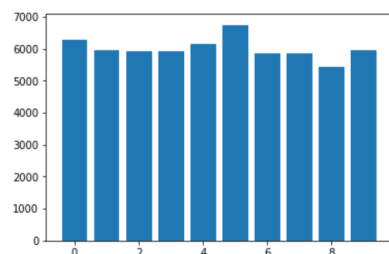
Handwritten character recognition

Dataset:

MNIST(digit):

Train dataset: 6000 images (10 classes)
Test dataset: 1000 images (test_split=0.6)

Frequency distribution of classes:
(Uniform distribution, no skewness in dataset)



Sample: (grayscale images)



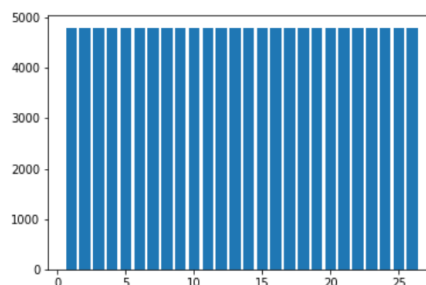
(28,28)

Label: 5

EMNIST(letter):

Train dataset: 124800 images (26 classes)
Test dataset: 20800 images (test_split=0.6)

Frequency distribution of classes:
(Uniform distribution, no skewness in dataset)



Sample: (grayscale images)



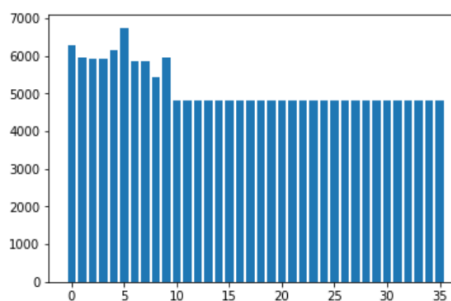
(28,28)

Label: 23

Combined:

Train dataset: 184800 images
(36 classes = 0-9 digit, 10-36 letter)
Test dataset: 30800 images (test_split=0.6)

Frequency distribution of classes:
(skewness towards digit dataset)
**right now ignoring the minor skewed distribution



Methodology:

Random Forest Classification:

Max_depth: 100

N_estimators: 1000

SVM:

C: 0.001

Gamma: 1

CNN:

epoch=15, opt=Adam

batchsize=64

validation_split=0.4

loss=categorical_crossentropy

metrics=Accuracy

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 26, 26, 32)	320
conv2d_1 (Conv2D)	(None, 24, 24, 64)	18496
max_pooling2d (MaxPooling2D)	(None, 12, 12, 64)	0
dropout (Dropout)	(None, 12, 12, 64)	0
flatten (Flatten)	(None, 9216)	0
dense (Dense)	(None, 256)	2359552
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 10)	2570
Total params: 2,380,938		
Trainable params: 2,380,938		
Non-trainable params: 0		

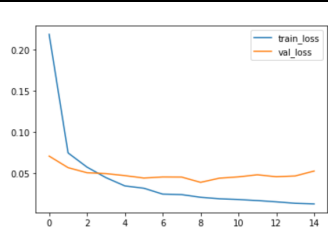
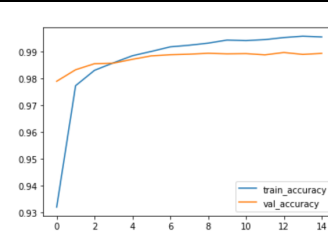
Subtasks:

1. Digit recognition

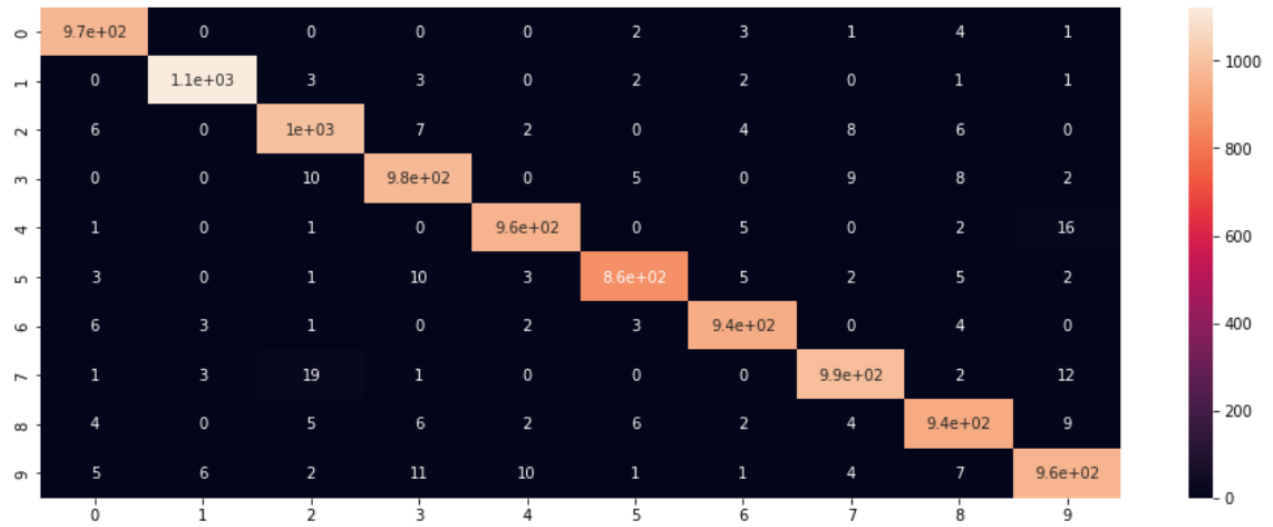
Dataset: MNIST(digit)

Link to collab: [Link](#)

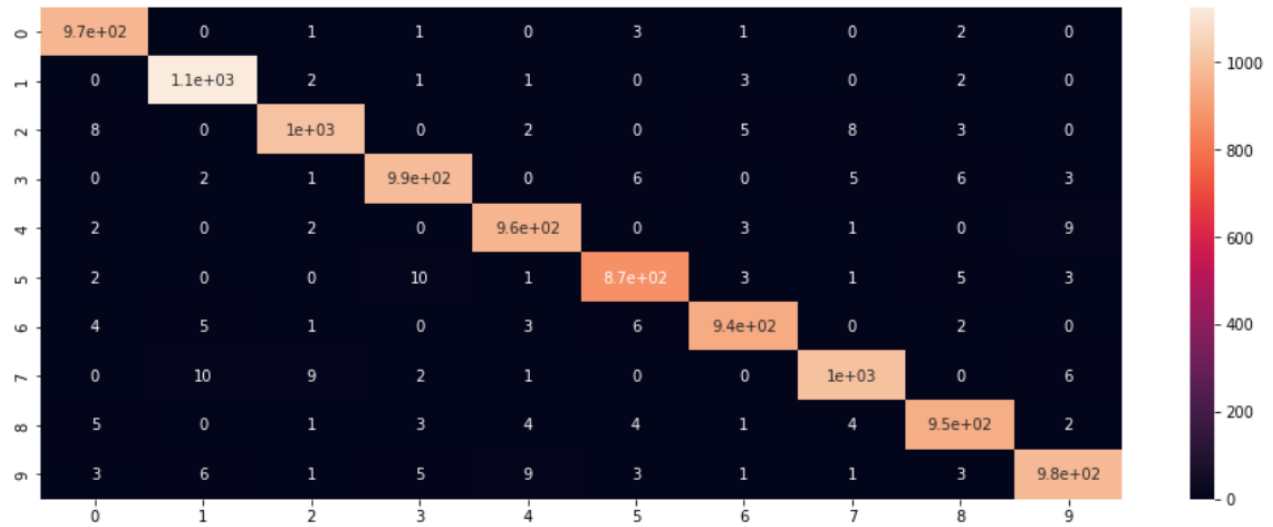
Results:

MODEL	TEST ACCURACY	LOSS	ACCURACY
Random Forest Classifier	97.12		
SVM	97.87		
CNN	98.83 (5 epochs) 99.08 (15 epochs)		

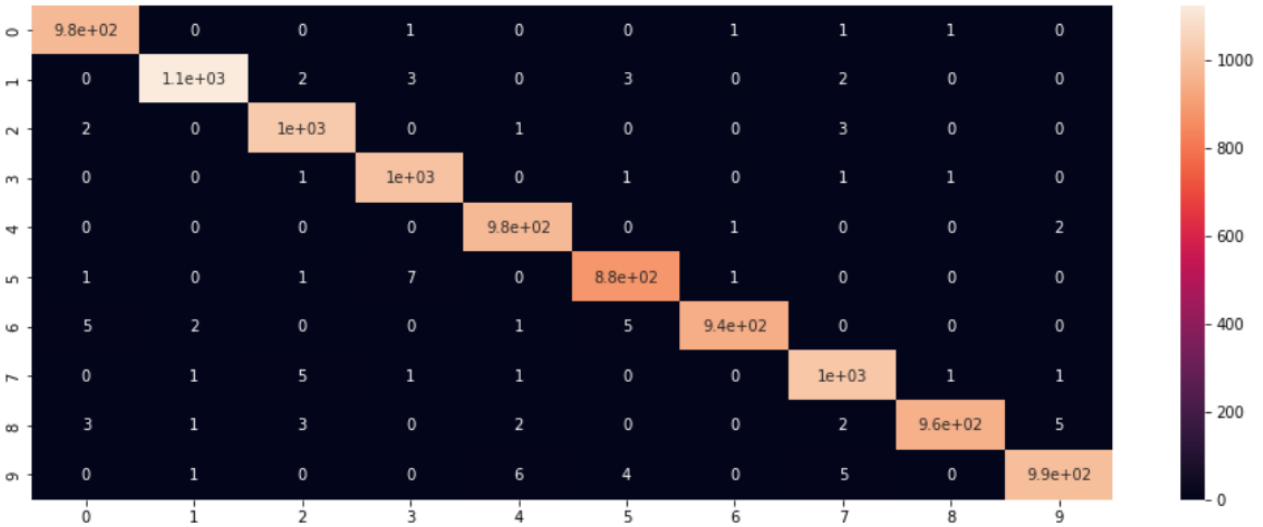
Random Forest Classifier:



SVM:



CNN:



2. Alphabet recognition

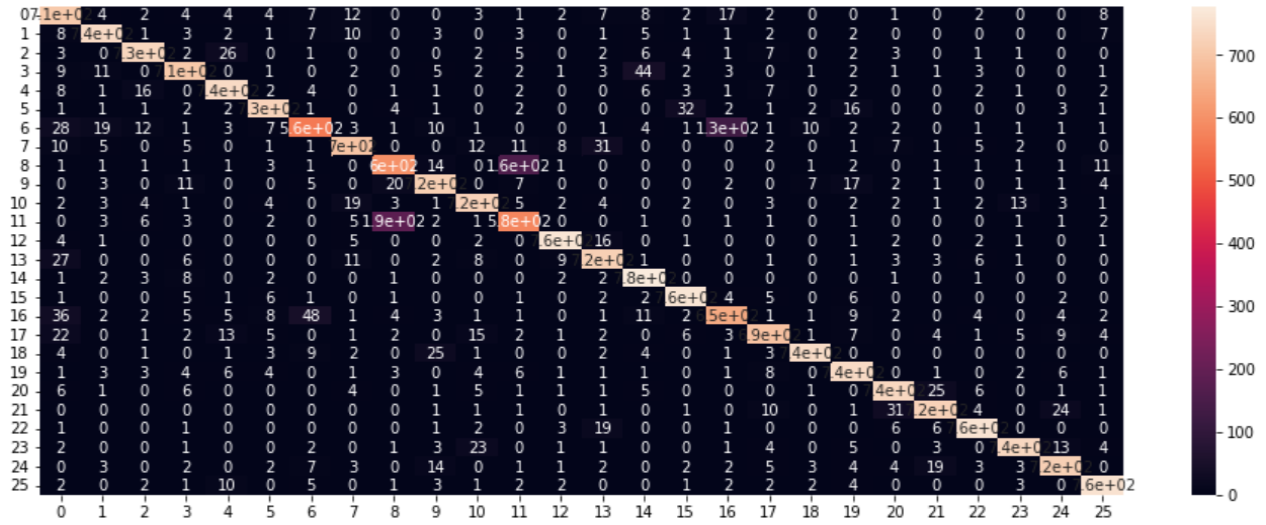
Dataset: EMNIST(letter)

Link to collab: [Link](#)

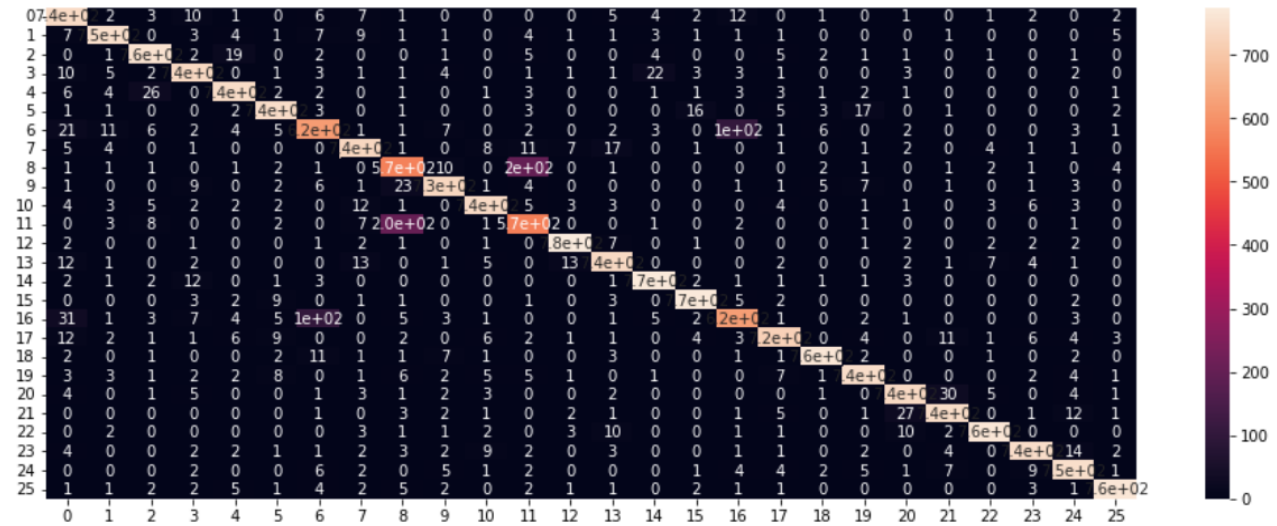
Results:

MODEL	TEST ACCURACY	LOSS	ACCURACY
Random Forest Classifier	89.07		
SVM	90.56		
CNN	93.02 (5 epochs) 93.69 (15 epochs)		

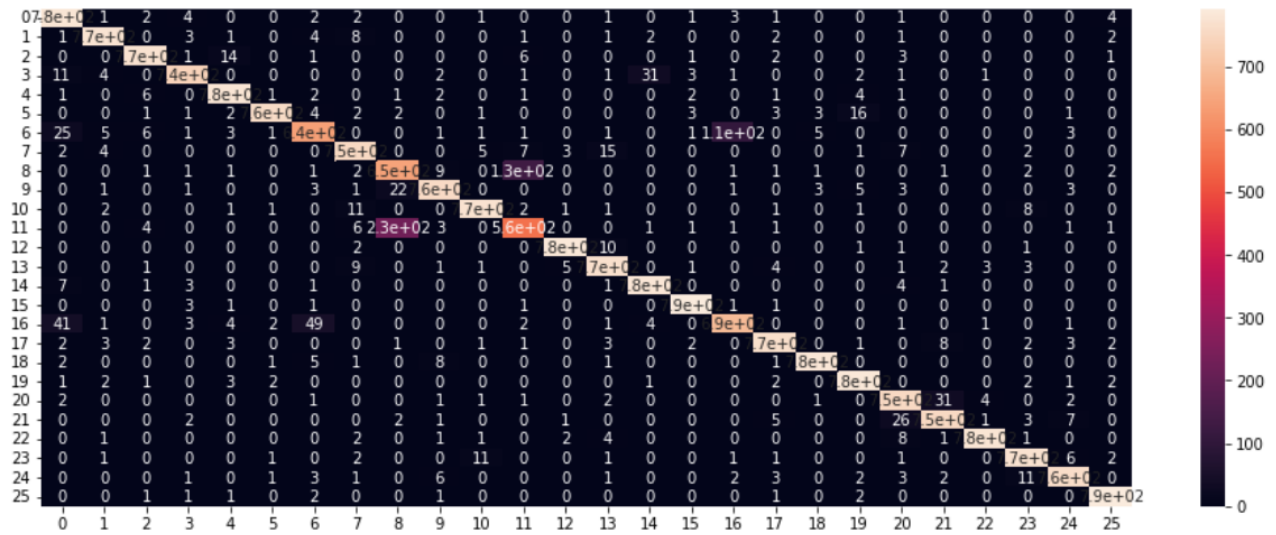
Random Forest Classifier:



SVM:



CNN:



3. Character recognition

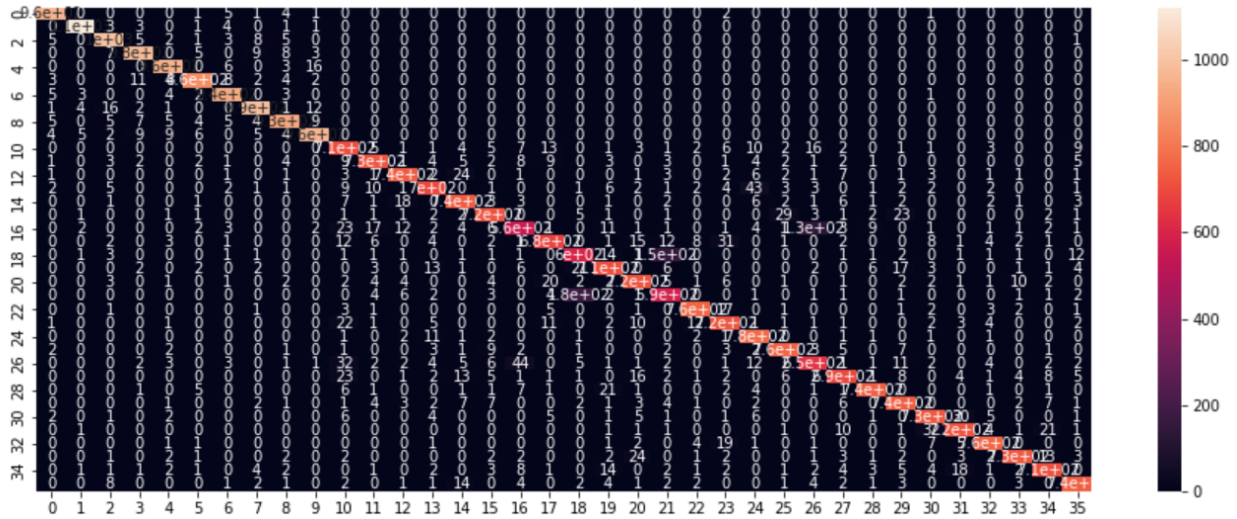
Dataset: MNIST(digit)+EMNIST(letter)

Link to collab: [Link](#)

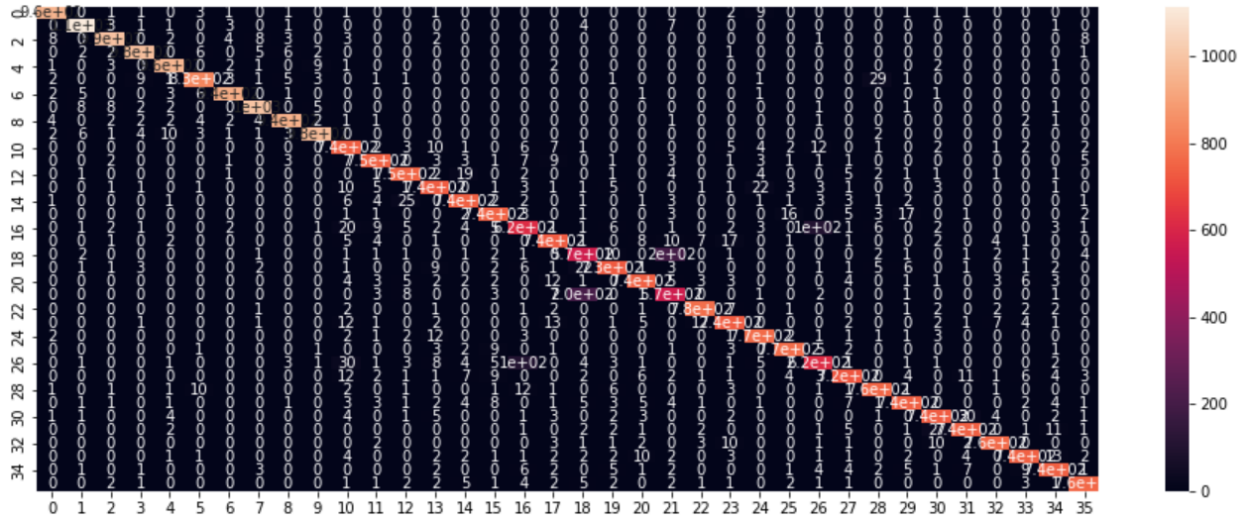
Results:

MODEL	TEST ACCURACY	LOSS	ACCURACY
Random Forest Classifier	91.37		
SVM	92.54		
CNN	94.89 (5 epochs) 95.27 (15 epochs)		

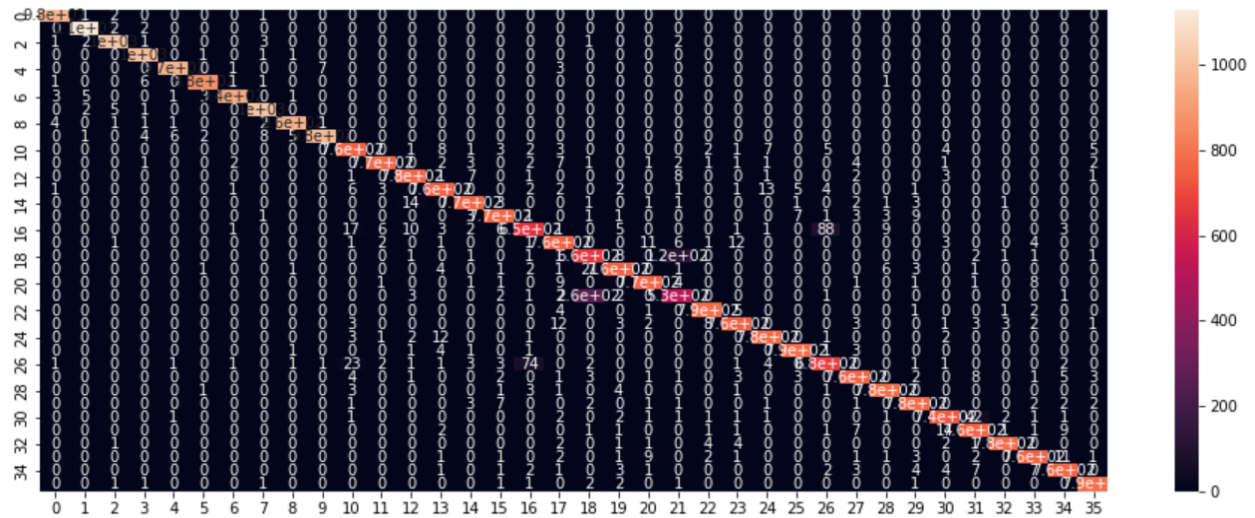
Random Forest Classifier:



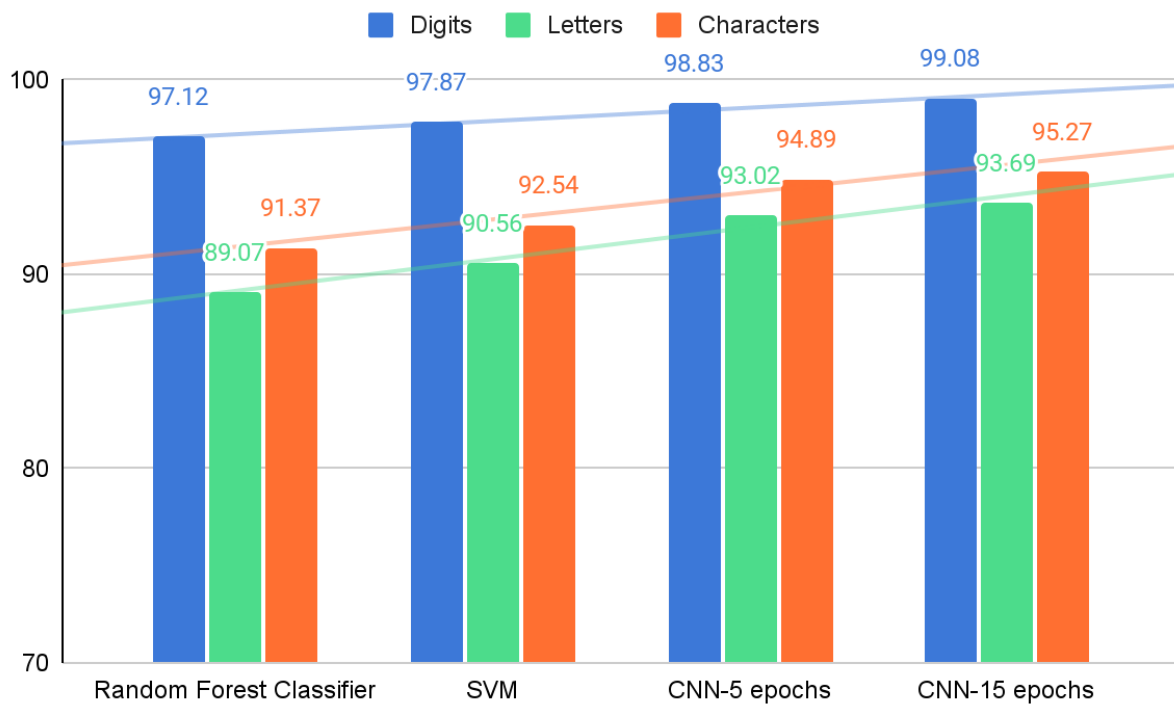
SVM:



CNN:



Note:-



It has been observed that the models can classify digits much easily than letter classification this may be inferred by the respective individual model performances along with the comparative performance in the combined dataset as seen in the confusion matrix. As the number of classes of letters is more, and that the letters are misclassified highly, the combined performance tends to perform lower averaged result skewed towards the letter's low outcome.

This can be due to the following reasons:

- The number of classes in digit classification is less than the letter, which regards the increased performance in individual runs. However, that statement does not account for why the model in the combined dataset could as well predict digits with better accuracy than the letters.
- The dataset for digits classification is slightly more, providing the model to train well on.

By specifically examining the predicted outcome we can observe a trend in all the models for misclassification of digit 5 with various other digits, however feebly. On the other hand, digit 1 is classified very well, due to its low complexity.

In the case of the letter dataset, the models are misclassifying letters ('g'-p'), and ('i'-l'). This trend has even been observed in the combined dataset performances. The very fact of the letters are much similar in appearance causes the model to wrongly predict them.

The same trend of misclassification is combined found when the models are trained in the combined dataset consisting of both the digits and the letters dataset.

CNN:

The learning curves of the CNN model depict the gradual training of the model neither overfitting nor underfitting the dataset. This can be observed by the increase in accuracy and decrease in loss over each epoch for both the training and validation dataset.

All the models have been run on collab GPU. Each epoch took 20 sec, with a total of 15 training epochs, the model for the combined and letter dataset took 300 sec. On the other hand for the digit dataset, each epoch took 8s, with a total of 15 training epochs, the model took 120s.

The CNN model outperformed the other models of SVM and Random Forest Classifier. According to the expected trend in training that deep neural networks have proven to perform better than other machine learning models, a similar result has been obtained.

However, much experimentation on the model architecture has been lacking given the increase in performance. The results of previous works have exhibited a bit more improved outcome which has been approximately achieved by this intuitive CNN model architecture which stands to be very simple in terms of complexity.

Due to the low complexity of the dataset the deep learning computationally heavy model architectures like VGG, Resnet, UNet, Inception Networks have not been deployed.

Further improvements:

Variation in CNN architecture by adding batch normalization or hyper tuning of parameters like learning rate, optimizer, batch size, can be varied and experimented upon.

The model could have been run for more epochs to reach the stable convergence on training which has not been incorporated. It has been observed in the smaller scale that increasing the epochs from 5 to 15 improved the results regardless of the dataset it was applied on. However, with the increasing epochs the chances of overfitting the training dataset increase. The point to note here is that the learning curves did start converging around 10 epochs, thus much necessity of increasing the epochs for such a trivial dataset was not mandatory.

Just a thought: that as the image classification is to be only handled on digits and letters, one can extract image boundaries by the edge detection filter which is based on the difference in the color intensity, and then carry the classification. However, this is something we can inspect in the CNN model that it may train its kernel weights such that it acts as an edge detecting filter. Thus the idea develops into providing the CNN with the initialization of initial kernel weights to edge detection filter rather than default zeros as it will help in faster training convergence.