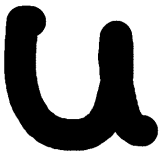
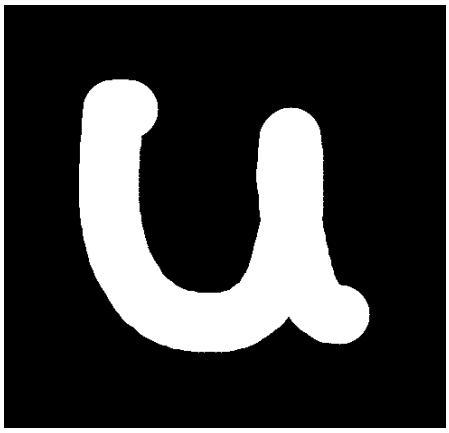
Task 2 IIITD Internship 2021

Part 1

1. Images from the following dataset are analysed and studied so as to enable a better preprocessing.
2. The images are preprocessed by first extracting the alphabets and digits using the opencv contour extraction. The images are cropped so as to minimize the extra pixels not necessary for feature extraction . The objective of this part is to make sure that only the digit and alphabet part of the image are there in the background is suppressed as much as possible. Then the extracted digits and alphabets are provided with a border around them in order to make them center. This helps in making them centerized in the image. The objective of obtaining the digits and alphabets from the original image was to reduce as much as image size by extracting maximum information possible from the image. Then the images with the border are inverted. The inversion of image is done in order to prevent the dying ReLU issue of using a ReLU activation function and make the pixels required for classification to be more prominent than other pixels.



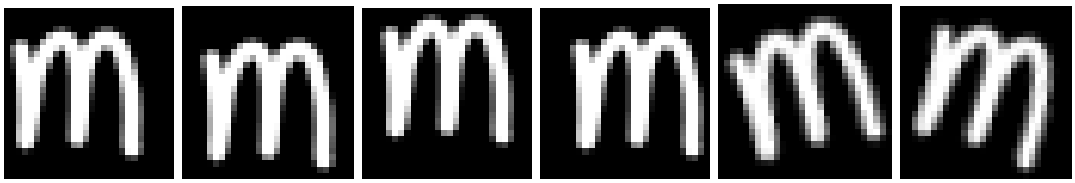
Before



After

3. Then the dataset is augmented. Image augmentation refers to the various transformations applied to the image like rotation, shift, zoom etc. The transformations used in this dataset to augment it are rotation and shift.
   - A shift of 2 pixels in a 28x28 pixel image is applied in all directions. This is done in order to make the model suitable for all types of input. e.g. Different handwritings have different writing sizes and styles.
   - A rotation from -30 to 30 degree is applied with a step of 10. This is done in order to cater to various types of human writing behaviours like cursive fonts.
     The augmentation helps in creating a dataset as close as possible to real life data.It

also helps in reducing overfitting of the model during training. As the training set images are only 2,480 containing 62 classes which is low to train a good algorithm, hence augmentation is performed.



(a) (b) (c) (d) (e) (f)

(a - d) Shift Augmentation (e - f) Rotation Augmentation

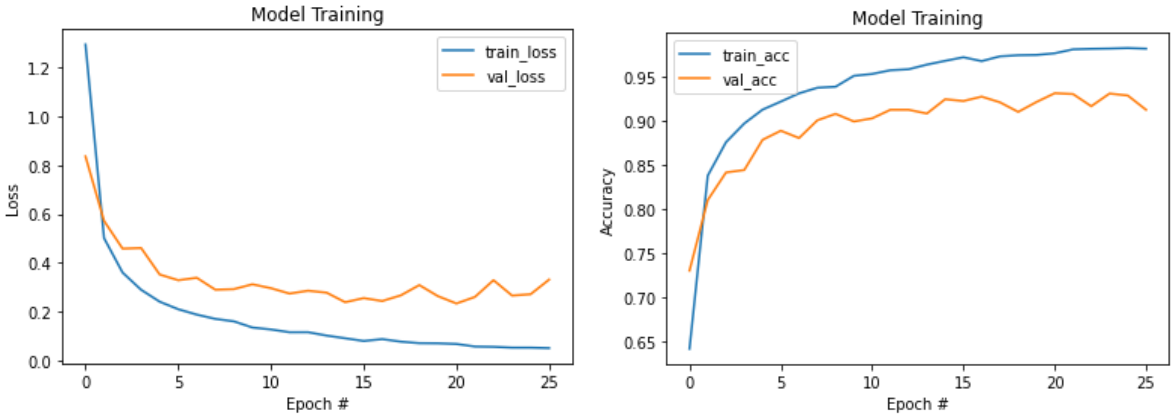
4. Then the model for the following task is prepared. The model for the following task is a pretty simple model with the following architecture prepared through Sequential API :

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 9, 9, 64)          51264
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 2, 2, 128)         73856
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 1, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 256)               33024
_________________________________________________________________
batch_normalization (BatchNo (None, 256)               1024
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 62)                15934
=================================================================
Total params: 175,998
Trainable params: 175,486
Non-trainable params: 512
_________________________________________________________________
```

5. The model is compiled with a “adam” optimizer which is a much better optimizer compared to RMSProp, ADAGrad and many more for the specific task. The model is fitted onto the

training and validation data for 100 epochs with a callback of EarlyStopping. The EarlyStopping callback monitors the validation accuracy with a patience of 5

Train Loss : 0.0500 - **Train Accuracy:** 0.9816 - **Val_Loss:** 0.3311 - **Val_Accuracy**: 0.9120 **Epochs count :** 26

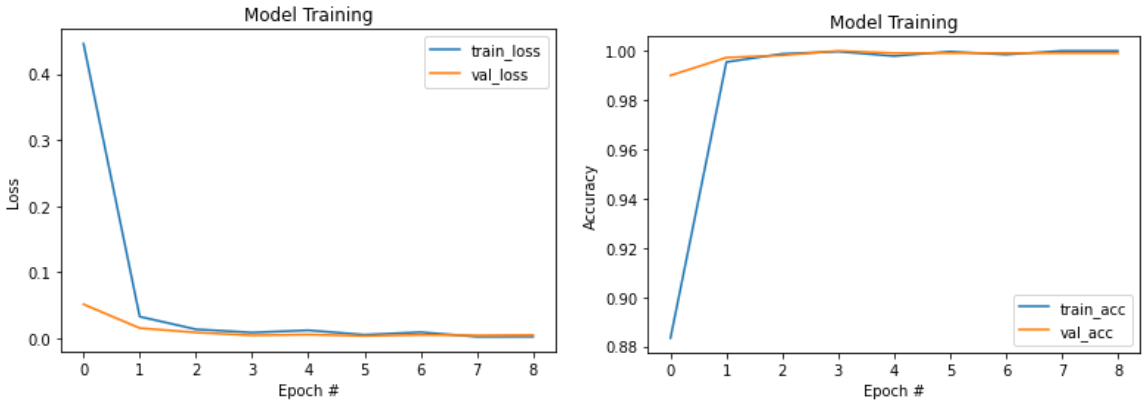


Part 2 - (0 - 9 Digits MNIST Based Classifications)

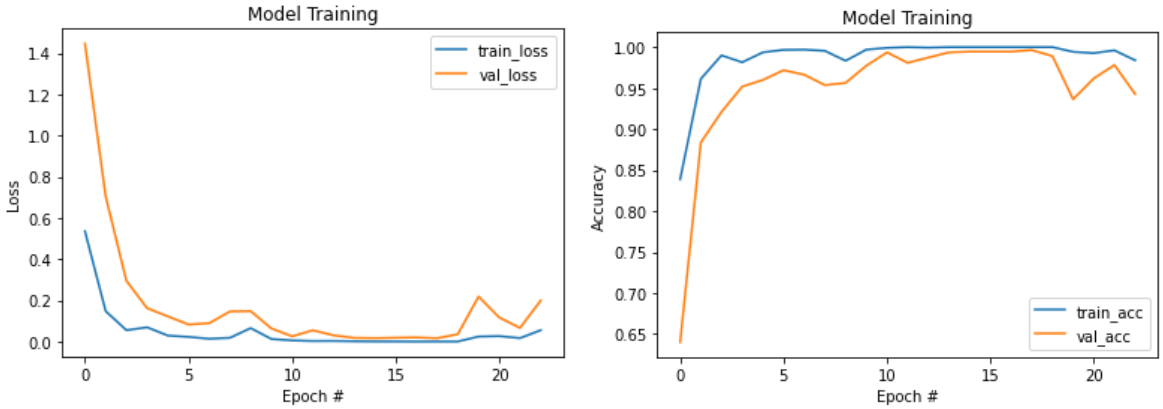
a) Data taken from Task 1

The dataset provided in the first part was extracted and images from 0-9 were obtained from the following dataset for the second part. These images were the one that were preprocessed the similar way as the images in the first and also augmented in order to increase the dataset size as it would prevent overfitting and help in making a better model. Then the model from the first part is loaded with the same weights and the dataset of 0-9 images is trained on the model. Various metrics like time taken accuracy loss validation accuracy and validation loss are observed for the following model. Then a model of the same architecture is taken from scratch in order to make weights random and the following dataset is trained on that. Similar as earlier various metrics like time taken, accuracy, loss, validation accuracy and validation loss are observed for the same model.

Model	**Time Taken**	**Val acc**	**Train acc**	**Val loss**	**Train Loss**
Random	93s	0.9782	0.9961	0.0666	0.0172
Pretrained	37s	0.9991	1.0000	0.0048	0.0024



Pretrained Model



Random Model

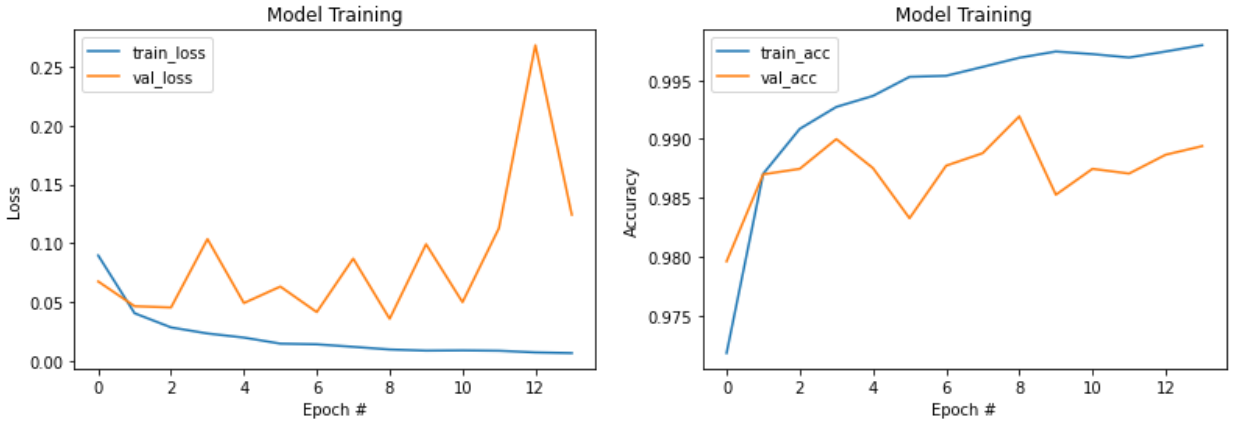
Observations made from above experiments :

1. The time taken for the pretrained network is higher than randomly initialized networks. The reason for this is that the weights in the pretrained network are familiar with detection of the digits and alphabets and just have to tune the weights in order to only detect the digits.
2. The epochs taken to converge are higher for randomly initialized networks than for the pretrained network . The reason for this observation is also similar to the one given in the first part as the weights just have to tune then to recognize digits.
3. Accuracy of the pretrained network both in training accuracy and validation accuracy is higher than the one randomly initialized.
4. Loss of the pretrained network in both training and validation is less than the one in the randomly initialized model.

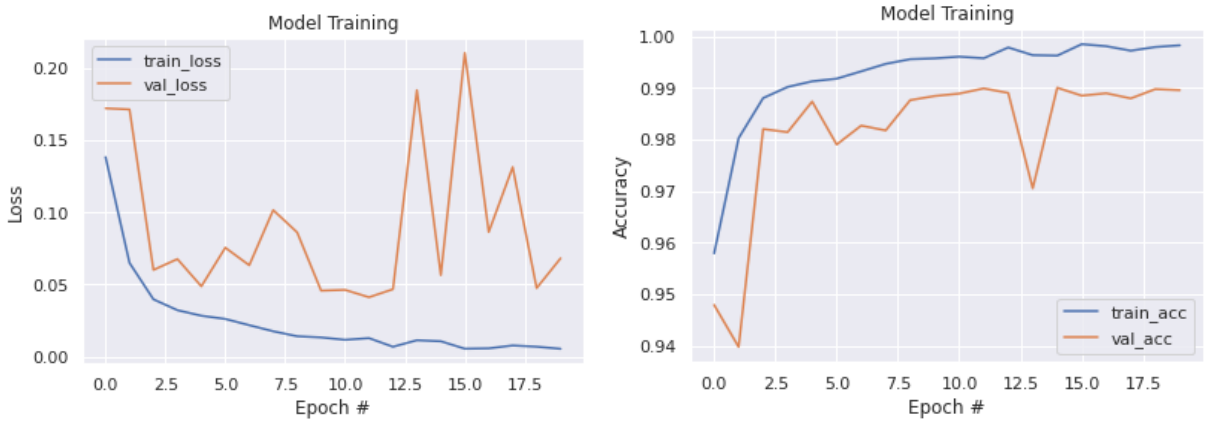
b) Standard MNIST data

Then the standard MNIST dataset with standard train and test splits is loaded. First a model with the same architecture and weights as the one in the first is taken and trained on the dataset and then the same architecture model with random initialization is trained on the dataset. Various metrics like time taken, accuracy, loss, validation accuracy and validation loss are observed .

Model	Time Taken	Val acc	Train acc	Val loss	Train Loss	Test Acc
Random	1074s	0.9900	0.9949	0.0565	0.0143	0.9926
Pretrained	537s	0.9875	0.9974	0.0498	0.0095	0.9922



Pretrained Model



Random Model

Observations made from the above experiment :

1. The time taken for the pretrained network is higher than random initialized networks. The reason for this is that the weights in the pretrained network are familiar with detection of the digits and alphabets and just have to tune the weights in order to only detect the digits.

2.The epochs taken to converge are higher for randomly initialized networks than for the pretrained network . The reason for this observation is also similar to the one given in the first part as the weights just have to tune then to recognize digits.

3. Validation accuracy and test accuracy of the pretrained network is lesser than that of randomly initialised model by 0.0025 and 0.0004 respectively which is negligible in comparison to the time both the models take to converge .Also the training accuracy of pretrained is higher than that of randomly initialised model here as well.

4. Loss of the pretrained network in both training and validation is less than the one in randomly initialized.

Conclusion : It was observed that the pretrained network performed much better than the randomly initialized network and the following worked out the way it was supposed to be and there was no strange behaviour observed.

Part 3 - (0 - 9 Digits Jumbled Data)

The new dataset provided in the following task was extracted and observed. The following dataset is just a MNIST dataset with jumbled up instances in order to check the model. The instances of different classes have been shuffled in order to provide a jumbled up MNIST dataset. The following dataset is loaded up with the pretrained model weights in the first part and trained. Then the model is trained with a random initialization. The observations like time taken, training accuracy, validation accuracy, training loss, validation loss of the following models were taken into account and studied.

Model	**Time Taken**	**Val acc**	**Train acc**	**Val loss**	**Train Loss**	**Test Acc**
Random	505s	0.1115	0.1246	2.3105	2.2570	0.1068
Pretrained	358s	0.1172	0.1163	2.2765	2.2861	0.0082

Conclusion : Apart from the fact that pretrained model works better here too in terms of time taken, losses, etc. the result in this part of the task reflects that the model won't perform well when the data itself is labelled in a wrong manner.

References :

1. Sharma, Arnab Sen, et al. "A Deep CNN Model for Student Learning Pedagogy Detection Data Collection Using OCR." *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*. IEEE, 2018.