Task 2

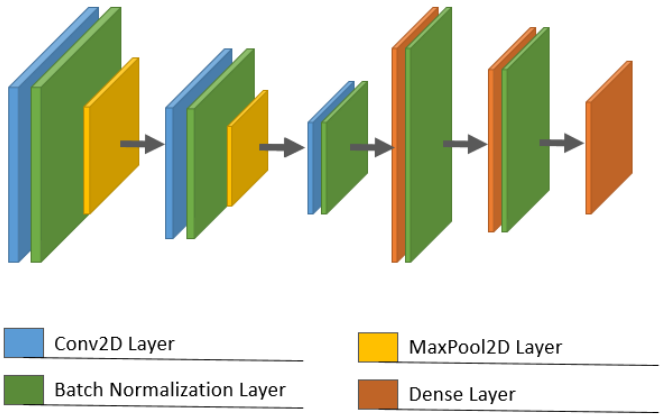
(CNN)

Task 2.1

Dataset Description: The actual shape of the images is (900, 1200, 3). Even being a black and white image, an RGB format was used to save the images. Hence to reduce the size of the image, it was converted from RGB to Grey. Also, the image was rescaled to shape (45, 60), which is sufficient to save the necessary data in the image.

The number of samples in the dataset is very low (2,480). Hence to increase the size of the dataset, data augmentation technique was used. The original images were randomly translated and rotated in order to create new samples of images. After data augmentation, the total number of input samples is increased to 62,000.

Model Architecture: A very basic architecture was implemented for the given problem statement as shown below.



Optimizer = rmsprop

Loss = Sparse Categorical Crossentropy

Training: The train dataset was split into train and validation dataset. The model was trained for 500 epochs with a batch size of 256. The shuffle parameter was also set to true. To get the best out of the training, a model checkpoint was created which saves the best weights of the model while monitoring the validation accuracy.

```
Train Accuracy: 0.9790524244308472 Test Accuracy: 0.9265322685241699
```

Task 2.2

The model architecture used for this task is same as described above, with only difference in the number of output tensors. In the previous task, there were total 62 class labels whereas in this task only 10 class labels are present.

Procedure:

1. Train the model using the dataset used for Task 2.1 having labels 0-9 only. Train the model for 150 epochs. This model will be used as a pre-trained network for future training.

   `Train Accuracy: 0.9931250214576721 Test Accuracy: 0.9779999852180481`
2. Load the train and test data from the MNIST dataset. Load the model with best weights from the previous training and start training the model for 5 epochs.

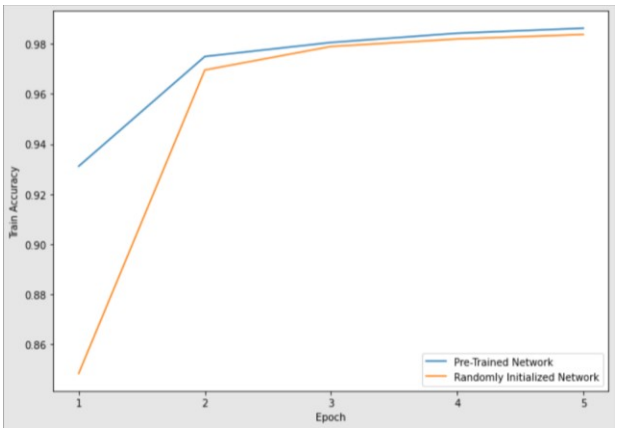
   `Test Accuracy: 0.9882000088691711`
3. Create a new model and start training this model for 5 epochs on the same dataset as used in step 2.

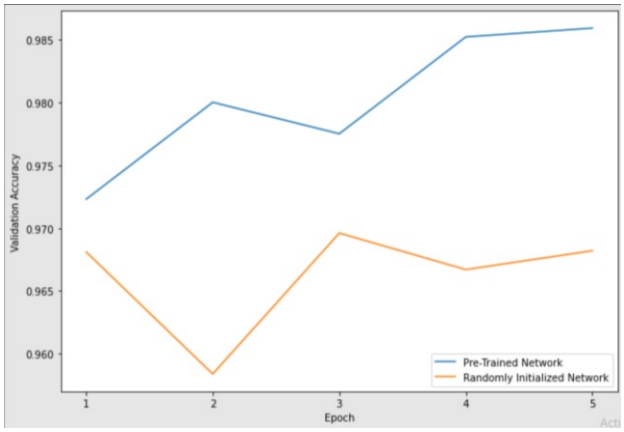
   `Test Accuracy: 0.9702000021934509`

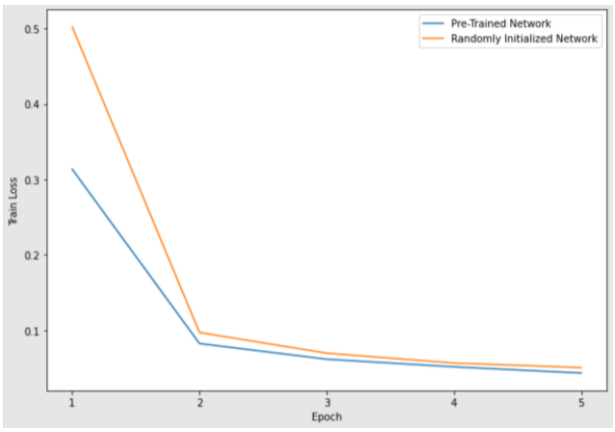
Observations: When same dataset was used to train both the models for same number of epochs, following observations were made:

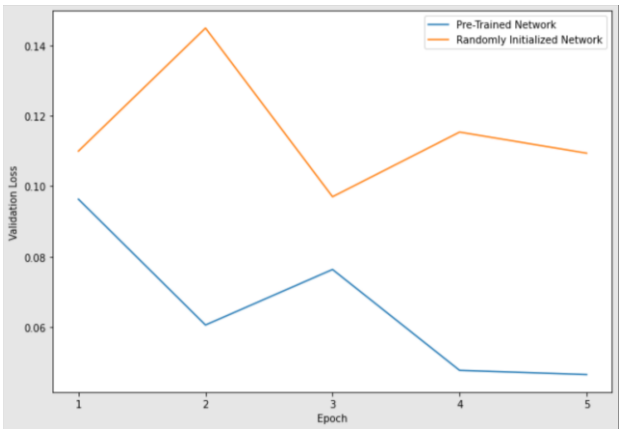
1. In both of the scenarios, the **convergence time** was significantly low when compared to the model trained in step 1 (dataset from Task 2.1). But if we compare the models trained on MNIST dataset, the model trained on a pre-trained network converged slightly faster than the network trained on randomly initialized network.
2. The pre-trained model gained a **train accuracy** > 90% in the first epoch itself while the randomly initialized network achieved that in the second epoch.
3. After each epoch, the train as well as validation **accuracy** is slightly greater than that of a randomly initialized network.
4. There is a significant difference in the validation **loss** of both the models. The train loss of the pre-trained network in each epoch is slightly lesser than that of the other network but the validation loss is very much lesser in case of pre-trained network. After 5 epochs, the validation loss of pre-trained network was `0.0464` while the validation loss of randomly initialized network was `0.1094`.

[Refer to the below graphs for quantitative analysis of the accuracy and loss]


Train Accuracy
Epoch
Pre-Trained Network
Randomly Initialized Network


Validation Accuracy
Epoch
Pre-Trained Network
Randomly Initialized Network


Train Loss
Epoch
Pre-Trained Network
Randomly Initialized Network

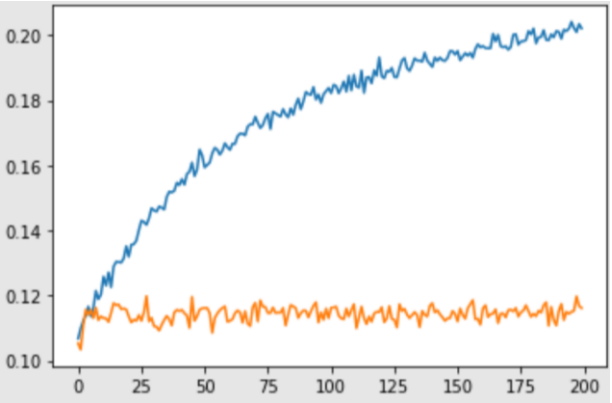

Validation Loss
Epoch
Pre-Trained Network
Randomly Initialized Network

Task 2.3

The same model architecture as used in previous tasks is used again. Only the dataset is changed.

Training on Randomly Initialized Network: The model is trained for 200 epochs on the given dataset. The train and validation accuracy achieved using this dataset is only ~20% and ~11.6% respectively. When the model is evaluated on the MNIST dataset, the test accuracy is 0.05% only.

Training on Pre-Trained Network: The pre-trained network used in Task 2.2 is used for this task as well. When the model is trained for 200 epochs, the train and validation achieved in this case is around ~22% and ~11.8% respectively. When the model is evaluated on the MNIST dataset, the test accuracy is 0.19% only.



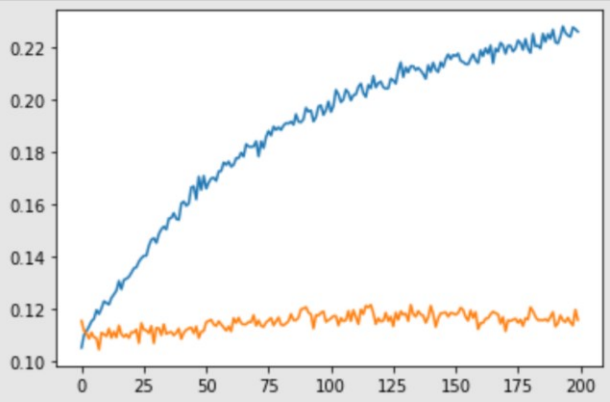


Fig 1 (left) represents the Train and Validation accuracy of randomly initialized network while Fig 2 (right) represnts the Train and Validation accuracy of pre-trained network.

The convergence time for pre-trained network is again lesser as compared to randomly initialized network, but the accuracy is not good. All the observations stated in Task 2.2 holds in this task as well.

Reason for bad accuracy (How this dataset is different from others):

1. The dataset given in this task is wronly labeled, i.e, directory named 0 contains images of digits other than 0 as well, hence the training was not done on the correct data.
2. The images given in this dataset have inverted colors. By inverted I mean that the dataset in previous task had digits written with black color on white background while in this dataset, the digits are written with white color on black background. (This does not affects the accuracy of the model but affects the convergence time)