INTERNSHIP/RA TASK 2021
EXPERIMENT LOG DOCUMENT

This is an experiment log document of **Task 2** submitted by **Manpreet Kukreja**.

PART A

It was observed that the dataset provided consisted of 62 classes each comprising 40 images.

- At first, a CNN consisting of 2 convolutional layers and 2 fully connected layers was trained on the dataset with no further augmentation applied to it. This resulted in achieving a training accuracy of 20.45% after 50 epochs. The batch size used for this training was 32 and the learning rate was 0.01.
- Now, in order to achieve higher accuracy, data augmentation was applied. This was done as the dataset provided consisted of very few images per class, and these wouldn't be enough to achieve an efficient model. A script, augment.py, was generated to randomly choose one of the three augmentation techniques to apply on each image. These were:
    1) Rotation by a random angle between -15° and 15° (both inclusive)
    2) Translation by -10 pixels to 10 pixels in both the x-axis and y-axis.
    3) Both rotation and translation
- Using this script, the images in each class were increased from 40 to 80. On training a model on this new dataset, it was seen that an accuracy of 61% was achieved after 300 epochs.
- Another augmentation step was applied to this new dataset. In this, warping of the images was done which zoomed in the images by 20% along each axis and increased the images per class to 160. This warping function was also added to the augment.py script. Training a model on this dataset led to achieving the accuracy of 60% within 70 epochs, rather than 300 epochs previously.

- It was observed that the dataset provided consisted of black alphanumerics on a white background and the MNIST dataset along with the dataset provided in Part C consisted of white numerals on a black background. This led to the inversion of the augmented dataset and similar results were achieved to the model that had been trained previously.
- To reduce the chances of overfitting, batch normalization and dropout layer techniques were tested. It was observed that using only dropout layers provided better accuracy in comparison to just using batch normalization or both batch normalization and dropout layers.
- To further increase accuracy, another convolutional layer was added to the network. Altering the batch sizes from 32 to 64 to 128 to 256 led to the conclusion that the best training accuracy was achieved when the batch size was 128. The training accuracy achieved when this network was trained was 80% within 100 epochs.
- Addition of another convolutional layer was done and trained, but this led to a decrease in accuracy in comparison to the previous model and this network was discarded.

The final training accuracy achieved after training the network was ~80%.

PART B

- At first, a model was trained from scratch on the 0-9 class dataset provided. It was observed that a training accuracy of **99.5%** was achieved in **60 epochs**.

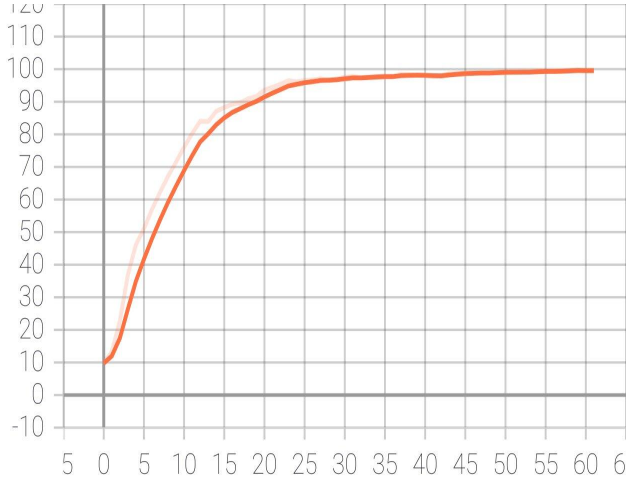

Fig. 1: Training Accuracy Graph

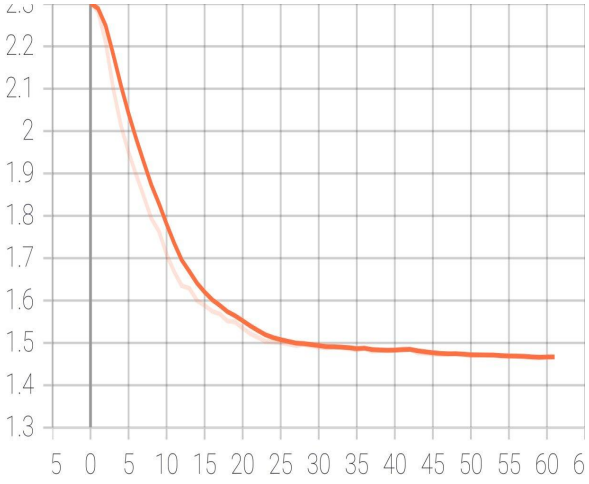

Fig. 2: Training Loss Graph

- This pretrained model was now trained on the MNIST dataset and it was seen to achieve a training accuracy of **98.43% over 20 epochs**. The same is shown below using graphs formed with the help of Tensorboard.

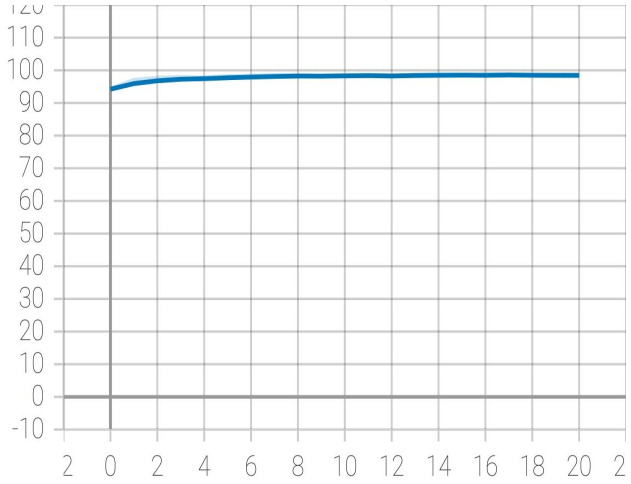

Fig. 3: Training Accuracy Graph of Pretrained and MNIST

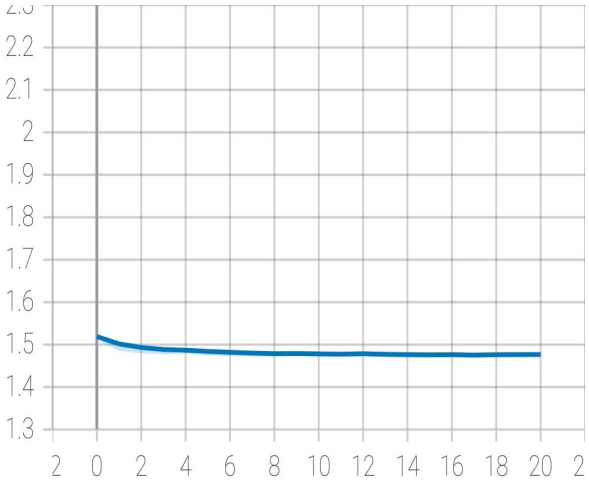

Fig. 4: Training Loss Graph of Pretrained and MNIST

- Next, a randomly initialized model was trained on the MNIST dataset. This network achieved an accuracy of **97.46% over 20 epochs**. It was allowed to train for a few more epochs, but no significant changes in the metrics were observed.

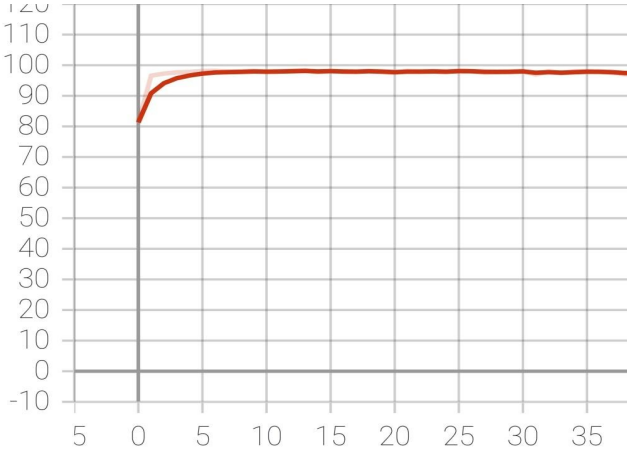

Fig. 5: Training Accuracy Graph of MNIST from Scratch

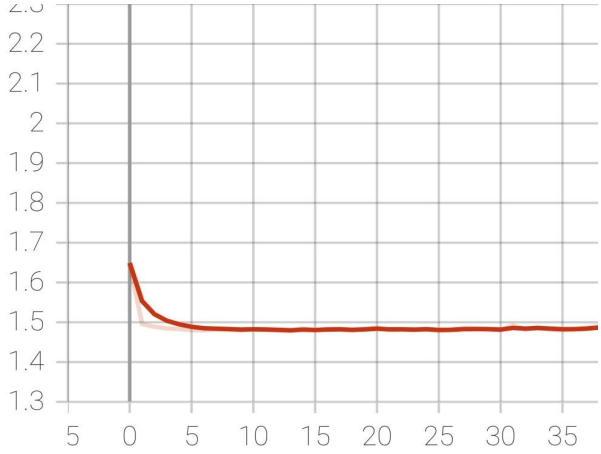

Fig. 6: Training Loss Graph of MNIST from Scratch

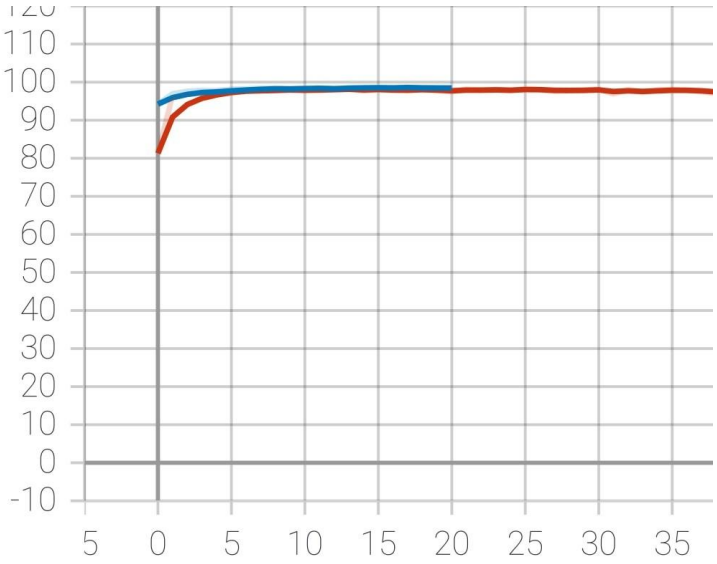

Fig. 7: Side by Side Comparison between Training Accuracy of Pretrained (Blue) and Scratch (Red)

- After the training of both the networks was complete, they were tested on the MNIST test split. Confusion matrix of the results were plotted and testing accuracies were noted. The testing accuracy of the pretrained model was found to be **98.82%**, whereas it was **97.84%** for the network trained from scratch.

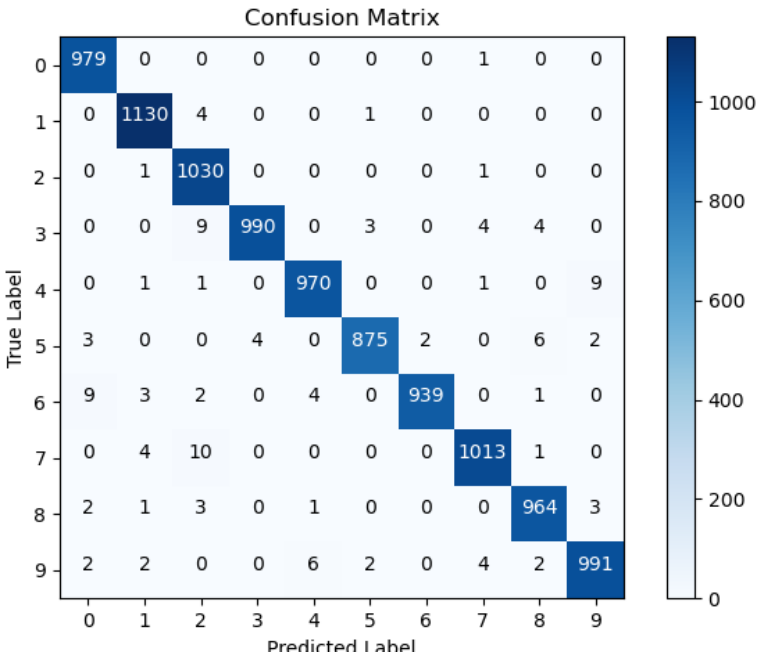

Fig. 8: Confusion Matrix obtained by testing the network trained using the pretrained network on MNIST test split.

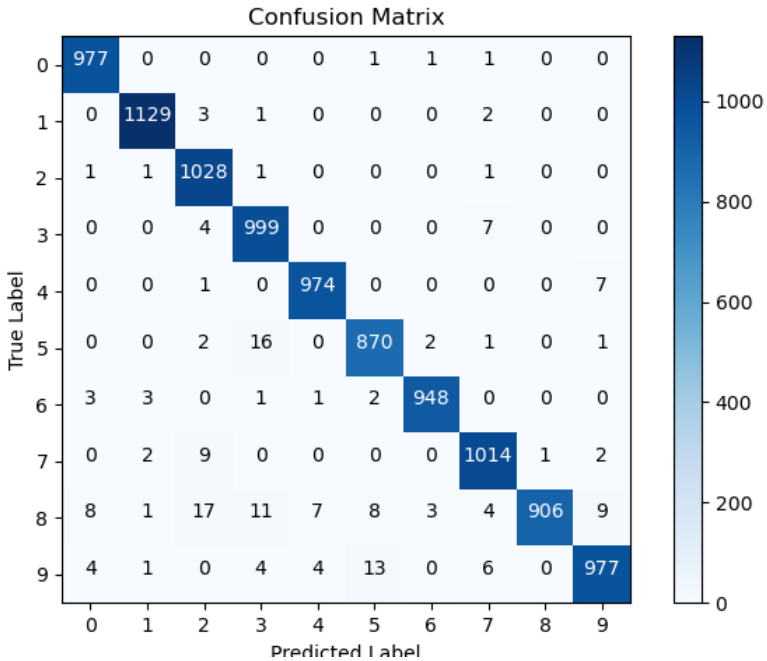

Fig. 9: Confusion Matrix obtained by testing the network trained from scratch on MNIST test split.

Conclusion for Part B

It was observed that the final training accuracy of the model trained using the pretrained network was higher than for the one trained from scratch. Also, the convergence time for this network is also faster as can be seen from the training loss and training accuracy graphs. Observing the confusion matrix made for both the networks, it is noted that the network trained using the pretrained model achieves better results altogether and has lesser misclassifications. Altogether, the model trained using the pretrained network achieved better results than that for the model trained from scratch.

PART C

The dataset provided consisted of 60184 images divided amongst 10 folders labelled 0-9. It was observed that a specific folder consisted of images of all numbers except the name of the folder. For example, the folder 0 consisted of images of number 1-9 and no images of the number 0. The first task was to unjumble this dataset and place the right images in the right folders. This was done using the pretrained MNIST network from Part B. A script, sortData.py, was written which would save an image in the folder which had the same name as the prediction done by the network. An example of this would be an image which was predicted as the number 8 would be placed in a folder named 8. The model didn't show 100% accuracy, and the observation stated earlier was used to lower the number of misclassifications.

- A network was trained from scratch on the unjumbled dataset. It was observed to achieve a training accuracy of **97.20% after 25 epochs**. Further training didn't result in any significant changes.

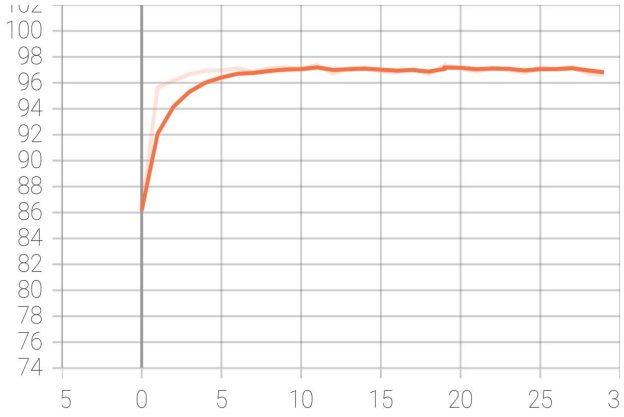

Fig. 10: Training Accuracy Graph of Network from Scratch

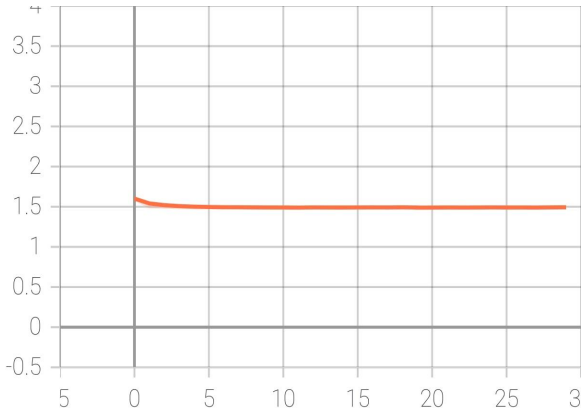

Fig. 11: Training Loss Graph of Network from Scratch

- A checkpoint from the pretrained model from Part A was loaded for training on this dataset. It was observed that **after 25 epochs**, it was able to reach a training accuracy of **88.34%**.

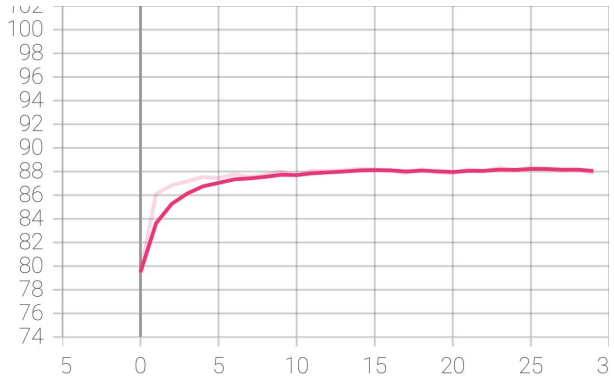

Fig. 12: Training Accuracy of Pretrained Network

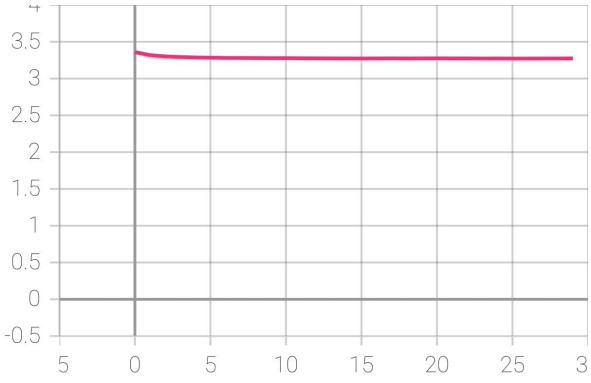

Fig. 13: Training Loss of Pretrained Network

On testing both these models on the MNIST test split, the model trained from scratch was able to achieve a testing accuracy of **98.24%**, whereas the model which was trained using the pretrained network from Task A was able to achieve a training accuracy of **88.88%**.

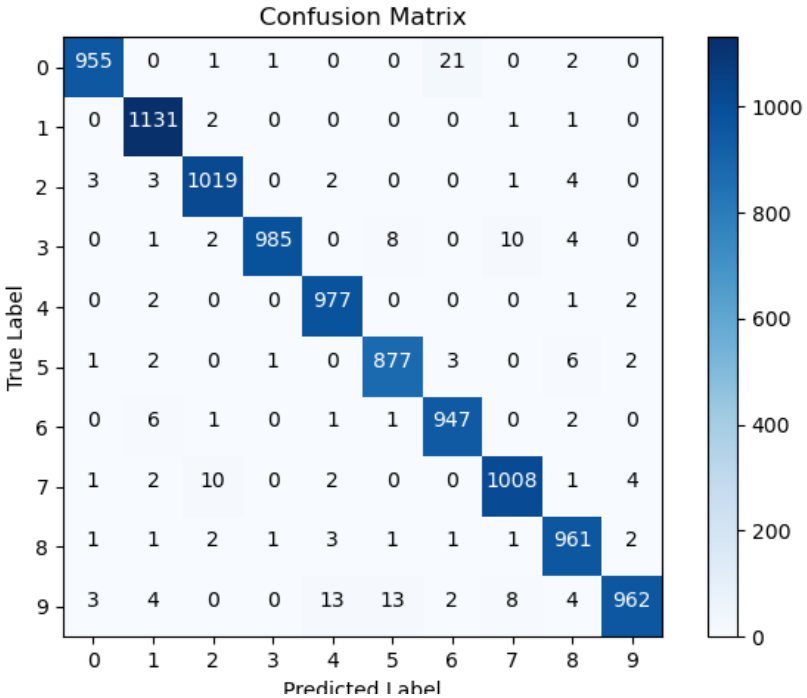

Fig. 14: Confusion Matrix of network trained from scratch

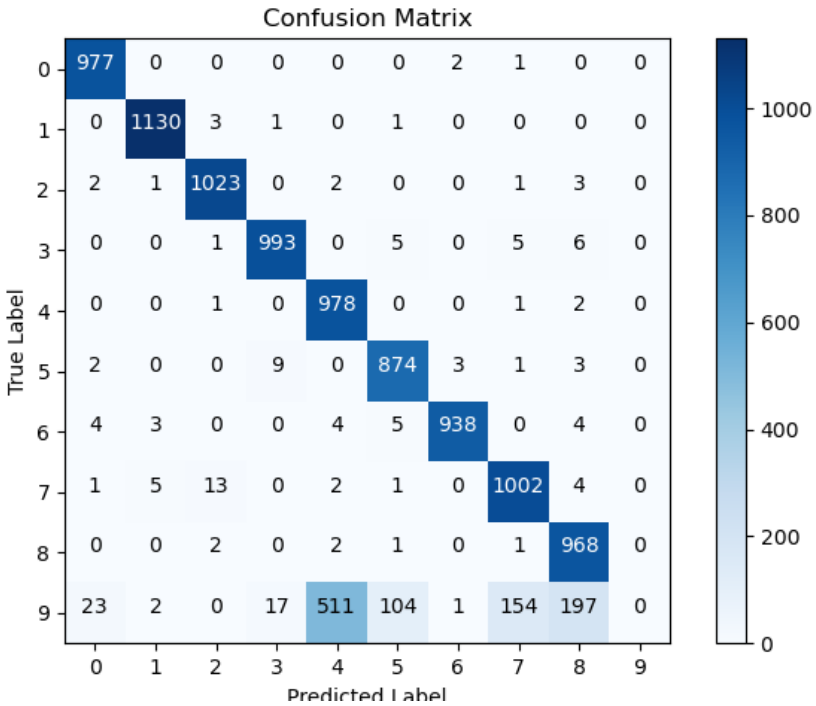

Fig.15: Confusion Matrix of network trained using pretrained network

It can be observed from the confusion matrix shown in Fig. 15 that the network failed in correcting classifying images of the number 9. It was thought that this was due to less data provided for learning. An attempt was made to improve this result by further increasing the data by applying random rotation to the dataset provided for Part A. After applying random rotation of -10° to 10°, the number of images per class were increased from 160 to 320. Using this new dataset, a model was trained for Task A again which reached a training accuracy of 85% in 160 epochs. Using this pretrained model, a new model was trained for Task C, but that model seemed to converge at ~79% training accuracy and no further increase was seen. After changing hyperparameters such as batch size and learning rate, no further improvement was seen in this latest model and it was discarded. The best testing accuracy achieved for the pretrained network for Task C was concluded to be 88.88%.

Conclusion for Part C

Comparing the two networks on convergence time, it can be seen that the network trained from scratch achieved a faster convergence time in comparison to the one which was trained using a pretrained network. Further, the final accuracy of the network trained from scratch was also higher. The model trained using the pretrained network was unable to classify the number 9 in the MNIST test dataset, which led to a huge difference in accuracy between the two networks.

Coming to the dataset, it had classes labelled from 0-9 which combined, consisted of roughly 60000 images. An observation was made that a folder didn't consist of images of that same folder name. Example being that a folder named 0 had no images of 0's in it. This was used in better placing the images in their correctly labelled folders and helped in lowering misclassifications.

The final test accuracies achieved in this subpart were **98.24%** and **88.88%** for the network trained from scratch and the network trained using the pretrained model respectively.

References

1. Image Augmentation: https://www.analyticsvidhya.com/blog/2019/12/image-augmentation-deep-learning-pytorch/
2. Dropout vs Batch Normalization: https://link.springer.com/article/10.1007/s11042-019-08453-9
3. Tensorboard: https://www.youtube.com/watch?v=VJW9wU-1n18
4. Confusion Matrix: https://deeplizard.com/learn/video/0LhiS6yu2qQ