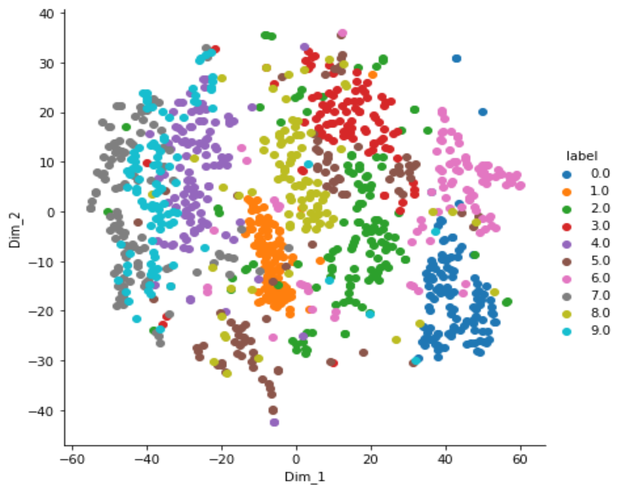
MIDAS@IIITD Internship/RA Task 2021

I chose **Task 2** for application at MIDAS Internship.

Standard MNIST Dataset Statistics

Total Dataset size	**Number of classes**	**Class wise distribution**
60,000	10	~6000

TSNE Plot of standard MNIST dataset

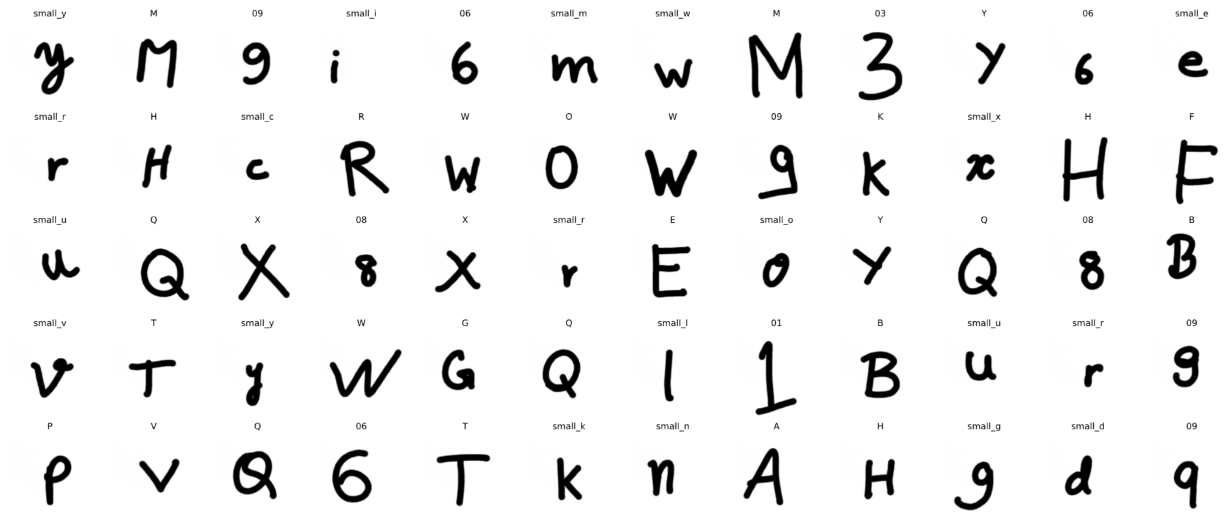


Q1 : Train CNN on shared dataset

Dataset Statistics details

Total Dataset size	**Number of classes**	**Class wise distribution**
2560	62	40

Samples of dataset



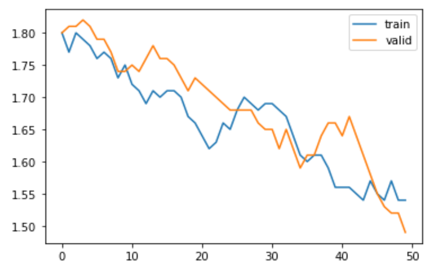
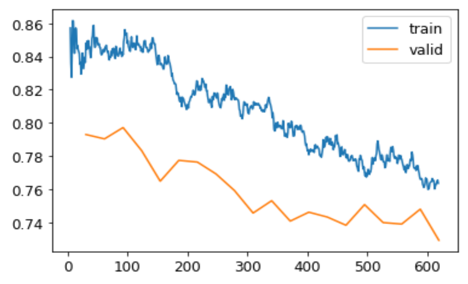
Preprocessing techniques :-

1. Resized image size from 1200*900 to 224*224 for faster model training.
2. Augmented Technique from http://fast.ai/
3. Tried **GANS** but failed to generate good quality samples because of improper training between discrinninator and generator due to less time. https://github.com/charchit7/GANS.

Experiments done :

1. We first experimented on baseline model i.e, 3 layer CNN network (stride = 1, with padding, kernel size = 3*3) and 1 layer fully connected (FC) layer (out layer = number of classes) to train our dataset.
2. Then, we experimented on deep neural networks i.e, VGG, AlexNet, ResNet to see its impact on the given dataset and found that with hyperparameter tuning ResNet is performing better.
3. We experimented on different hyperparameters like batch size, optimizer like Adadelta, Adagrad, etc and learning rate, etc. Best results and hyperparameters are shown in the observation table below.

Observation Table

Hyperparameters used : **Batch size :** 64 **Optimizer :** Adam **Learning rate :** 0.001		
Qualitative Metrics	**Baseline**	**Using resnet34**
Accuracy	62%	**87%**
Loss curve		

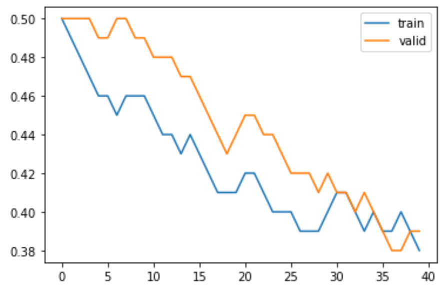
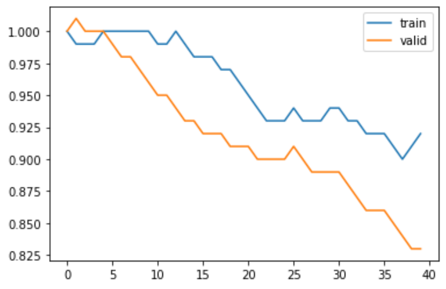
Observations/Analysis

1. Adam optimizer performs really well in comparison to other optimizers.
2. Batch size is given as 64 to take almost all labels in batch. Reducing it affects the model performance.
3. Loss curve shows that our model trains well.
4. We found in predictions that “O” is classified as “0”, “I” is classified as “1” and so on. These misclassifications are done by our best model. For this, I wanted to add more samples for classes using data augmentation but failed because of time constraint.
5. We used adaptive adaptive learning rate to have better optimum lr value.

Q2 : Comparison of pre-trained with randomly initialised weights in network for MNIST (0-9)

Observation Table

Hyperparameters used : **Batch size :** 64 **Optimizer :** Adam **Learning rate :** 0.001		
Qualitative Metrics	**Pre-trained network (Resnet34)**	**Randomly initialised network**

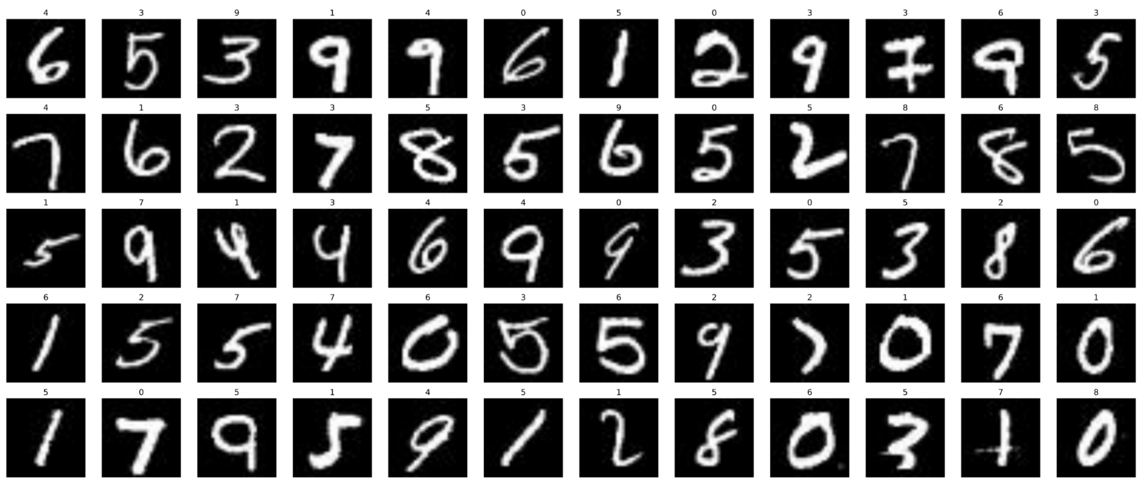
Accuracy	**96.2%**	94.5%
Training time	**128 seconds**	222 seconds
Loss curve		

Observations/Analysis

1. With initialisation with pre-trained weights, our model trained early in comparison to a randomly initialised network. Possible reason could be since it is similar kind of task (and similar dataset) so training the whole model from scratch (with random initialisation) will take time as compared to training from some trained seed (using pre-trained network).
2. Accuracy of pre-trained network is better than randomly initialised network since pre-trained trained upon similar tasks previously and so have better understanding for the similar task. So, fine-tuning over that helps in model training in comparison to random initialization.
3. Since there is pre-learning in the pre-trained model, the loss curve of random initialisation starts from higher loss in comparison to pre-trained.

Q3 : Comparison of pre-trained with randomly initialised weights in network for MNIST (0-9) for MNIST-Task3 dataset

Samples of dataset:

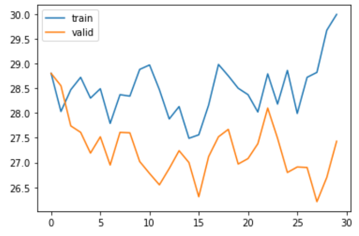


Dataset Statistics details

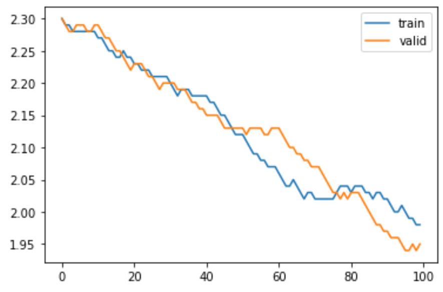
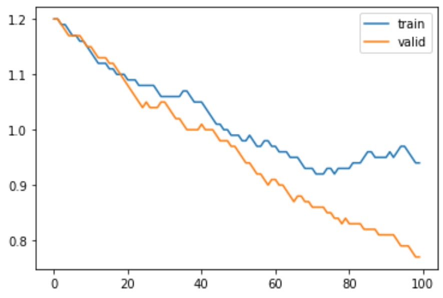
Total Dataset size	Number of classes	Class wise distribution
60000	10	`(5980, 5807, 6009, 6037, 5914, 6139, 6037, 5954, 6129, 5994)`

Observation Table

Hyperparameters used :
Batch size : 64
Optimizer : Adam
Learning rate : 0.001
Note* : Accuracy of model trained on noisy data = 11%



Qualitative Metrics	Pre-trained network	Randomly initialised network

Accuracy	75%	**92%**
Training time	322 seconds	**241 seconds**
Loss curve		

Observations/Analysis

1. Since data given in Q3 is noisy because of which our pre-trained network performed worse in comparison to random initialisation. The reason is training is done on noisy data (using a pre-trained network) which makes models learn noisy features and divert from proper learning.
2. Training time using a pre-trained network takes more time in comparison to using random initialisation due to improper training.

Resources Used

- Used google colab for GPU utilization for deep neural networks.
- Fast AI for writing models and other metrics.

References :

1. Model link :

- **Q1 :** https://drive.google.com/file/d/1-NeyT_BRyjRfDHl4wFT-h3RVYY5dJci0/view?usp=sharing
- **Q1 :** https://drive.google.com/file/d/1-EfsDTiJv3rF2oS8i9ANv9O6POZ9L2q3/view?usp=sharing
- **Q2 :**https://drive.google.com/file/d/1B_XDIh3kTVDhjtc2VoNjC1-XxHHMGBtS/view?usp=sharing
- Q3:https://drive.google.com/file/d/1CGfZIfPkqgErM69iVuHIyG-HedmIuInZ/view?usp=sharing
- Q3:https://drive.google.com/file/d/1COaEMEIippP8basx-qV62sGP5JnHdnqB/view?usp=sharing

2. https://arxiv.org/abs/1411.1792
3. https://www.jeremyjordan.me/nn-learning-rate/
4. https://arxiv.org/pdf/1812.01187.pdf

5. Hyperspherical Variational Auto-Encoders (Davidson, Falorsi, De Cao, Kipf, and Tomczak, 2018): https://www.researchgate.net/figure/Latent-space-visualization-of-the-10-MNIST-digits-in-2-dimensions-of-both-N-VAE-left_fig2_324182043
6. Analyzing and Improving the Image Quality of StyleGAN (Karras et al., 2020): https://arxiv.org/abs/1912.04958
7. Semantic Image Synthesis with Spatially-Adaptive Normalization (Park, Liu, Wang, and Zhu, 2019): https://arxiv.org/abs/1903.07291
8. Few-shot Adversarial Learning of Realistic Neural Talking Head Models (Zakharov, Shysheya, Burkov, and Lempitsky, 2019): https://arxiv.org/abs/1905.08233
9. Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling (Wu, Zhang, Xue, Freeman, and Tenenbaum, 2017): https://arxiv.org/abs/1610.07584
10. These Cats Do Not Exist (Glover and Mott, 2019): http://thesecatsdonotexist.com/
11. Large Scale GAN Training for High Fidelity Natural Image Synthesis (Brock, Donahue, and Simonyan, 2019): https://arxiv.org/abs/1809.11096
12. PyTorch Documentation: https://pytorch.org/docs/stable/index.html#pytorch-documentation
13. MNIST Database: http://yann.lecun.com/exdb/mnist/