MSDS 458

Assignment #1

Setu Madhavi Namburu

1. **Abstract:** The MNIST database consists of handwritten digit images that are used to build classifier systems to help USPS scan the zipcodes and classify mails automatically. Accurate classification of these mails is very important to deliver right mails to the right customers on time. A data scientist is hired to build an automatic classifier system using Deep Neural Networks (DNN) using Keras Library in python. This report presents the research results from various models and set of recommendations along with executive summary on final selection of the model.

2. **Introduction:** The purpose of this research is to build a legible Deep Neural Network (DNN) model(s) for classifying handwritten digits from MNIST database and make model recommendations to the management for practical deployment purpose. The MNIST database consists of a set of 70,000 small (28x28 pixel) grayscale images of digits (0-9) handwritten by high school students and employees of the US Census Bureau. Each image is labeled with the digit it represents. The data scientist is tasked with analyzing the data and building various DNN models using Keras Library in Python and come up with best model/set of models meeting two requirements: 1. High test accuracy, 2. Reasonable computational performance with stability. Heavy visualization will be used to present and discuss the results throughout this report.

2.1 Data Description: The dataset consists of 60000 train images and 10000 test images that comes shipped in python's Keras library. The X-variables are numpy arrays of size 28*28 (pixels of the images representing grey-white scale, values range from 0-254) and the target variables are integers from 0-9 (class labels). Figure 1. shows the counts of train and test images in each of the classes, the proportions seem to be approximately balanced. Set of sample greyscale images are shown in Figure 2 to give a glimpse of ease ness/difficulty in classifying. The 28*28 numpy arrays from each image are reshaped into 60000*784 train, 10000*784 test 2-D arrays & the values are divided by 255 (as the scale ranges from 0-254) to normalize the data. The target variable is 1-hot-encoded as it is a classification problem.

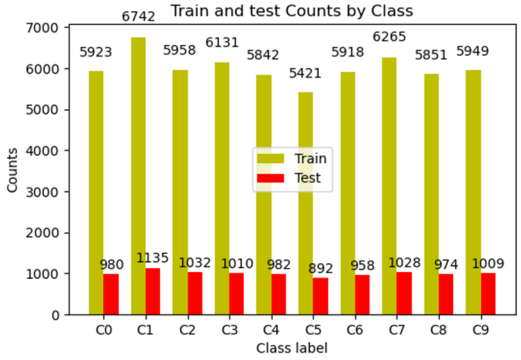

Figure 1. Class counts in train and test images

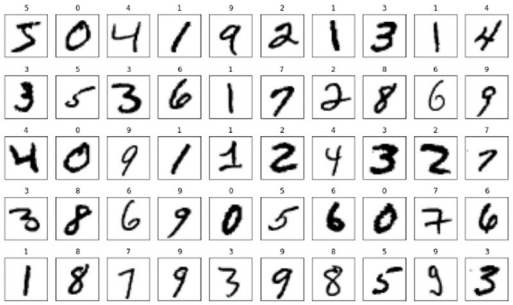

Figure 2. Set of sample images

2.2 Exploratory Data Analysis (EDA): To understand the segregation of the classes, the pixel data is visualized using 2-D scatter plots using 3 different methods. The data has very high dimension of 784 features, so just picking two pixels is not very useful in seeing a pattern. PCA tries to capture as much variation in the data using first few components, in this case just 2 PCs were not able to capture enough variation to show the segregation in classes, while it is better than random pixels. T-SNE is a powerful visualization technique to project any number of dimensions to 2D by maximizing the distance between classes. As seen in the figure, most of the classes seem to have fair segregation while the digits 4 & 9 seem to be heavily overlapped. This is very encouraging to continue further into building classifiers.

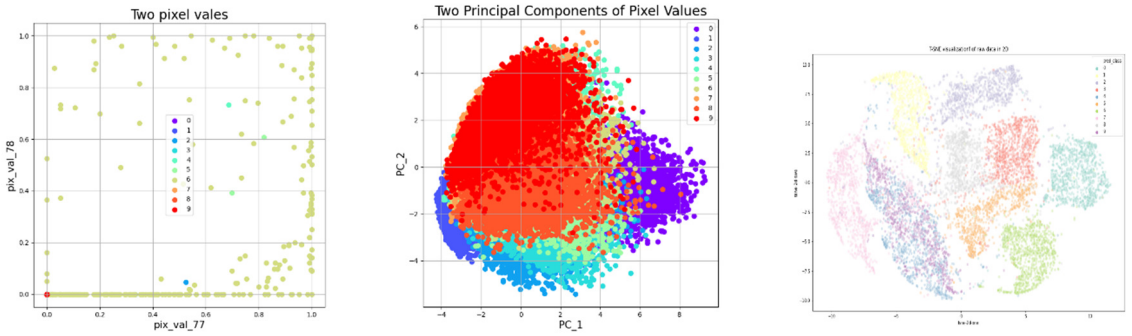

Figure 3. Visualization of image pixels in 2D – raw, PCA, T-SNE

3. **Literature Review:** This database is heavily used in research as it became the defacto test case for any new DNN research in the literature. It was documented that there are few mis-labeled original images in the data, so 100% accuracy is never possible with this data set. The images were perfectly classified upto maximum attainable accuracy by various researchers in the field (i.e., >99.5%). The data website [3] reported test error rates from numerous research papers, some of the lowest errors (near perfect classifier) are from classifiers: K-NN with non-linear deformation (P2DHMDM) - 0.52%, 6-layer NN 784-2500-2000-1500-1000-500-10 (on GPU) [elastic distortions] – 0.35%, committee of 35 conv. net, 1-20-P-40-P-150-10 [elastic distortions] – 0.23. In the survey paper by Kumar et al [2] showed an error of 0.4% with simple CNN with elastic distortion. The best classifier to date reported is Gradient based learning, LeNet-5 by LeCun et al [4]. [1] presented some intuitive and tactical way by modifying LeNet-5 (i.e., by adding more CNN layers, deeper networks and modifying parameters) to dynamically achieve reduced error rate – 0.59% is the reported error rate.
4. **Methods:** In this research study, single layer 1-node and 2-node networks are studied to understand the innerworkings of the DNNs. The analysis & synthesis of the results from these two networks is presented in detail. Next, several experiments are conducted using various network architectures on raw and reduced dimensions using PCA and random forests. The performance metrics are evaluated in terms of classification accuracy and stability of the models. The experimented are conducted on Windows machine with 32GB RAM, while the training time is reported, marginal priority is given to the computation time to make sure it is relatively reasonable. Below table presents the experiments set up, while it is not full design of experiments, 4 themes are explored (raw data single vs 2 layer with low to high number of nodes, PCs as input features to single layer network, RF features as inputs to single layer network). Exp_key serves as the reference id to keep track of results in plots and tables. Pandas, numpy, matplotlib, sklearn, tensorflow keras are the main python libraries used to process data, train & test models and present the results through visualization. In all the experiments, rmsprop

is used as the optimizer & categorical cross entropy is used to minimize the loss with a goal to minimize the error. ReLU is used as the activation function in all the hidden layers and softmax in the output layer.

Type_of_features	No_hidden_layers	No_hidden_units	input_shape	exp_key	2n_Layer_hidden_units
Norm_raw	1	1	784	Norm_raw_1_1_784	N/A
Norm_raw	1	2	784	Norm_raw_1_2_784	N/A
Norm_raw	1	54	784	Norm_raw_1_54_784	N/A
Norm_raw	1	128	784	Norm_raw_1_128_784	N/A
Norm_raw	1	238	784	Norm_raw_1_238_784	N/A
Norm_raw	1	543	784	Norm_raw_1_543_784	N/A
Norm_raw	2	54	784	Norm_raw_2_54_784	128
Norm_raw	2	128	784	Norm_raw_2_128_784	128
Norm_raw	2	234	784	Norm_raw_2_234_784	128
PCs	1	54	200	PCs_1_54_200	N/A
PCs	1	128	200	PCs_1_128_200	N/A
PCs	1	238	200	PCs_1_238_200	N/A
RF_Features	1	54	70	RF_Features_1_54_70	N/A
RF_Features	1	128	70	RF_Features_1_128_70	N/A
RF_Features	1	238	70	RF_Features_1_238_70	N/A

Table 1: Matrix of Experiments

5. Results:

Experiment 1 – single hidden layer with one node model:

Figure 4 shows the architecture of single hidden layer DNN with 784 nodes in the input layer, 1 node in the hidden layer and 10 nodes in the output layer for 10 classes. We can see that this simple network itself has 805 trainable parameters (complex space) that need to be optimized to give best possible minimum error.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
hidden_layer (Dense)         (None, 1)                 785
_________________________________________________________________
output_layer (Dense)         (None, 10)                20
=================================================================
Total params: 805
Trainable params: 805
Non-trainable params: 0
```

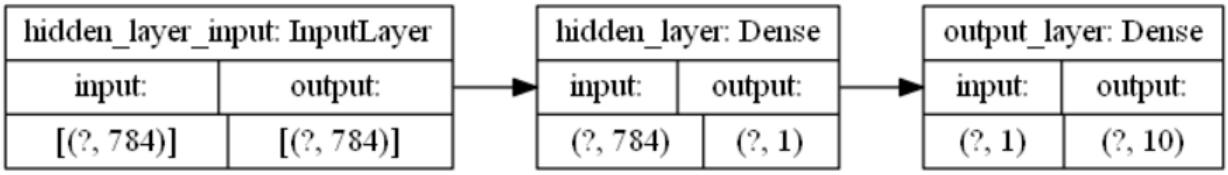

Figure 4. Single hidden layer network architecture

Training this network with input and output data, gave the following insights:

The network is trained using 30 epochs & the training and validation performance metrics (accuracy and loss) are plotted in Figure 5. While the performance metrics are quite stable, the accuracy is worse than a random guess. Since the hidden layer has only one node, the extracted activation values can be visualized using boxplot (Figure 6) to see the separation of predicted classes in training data. Only the digit 6 seem to have been classified well whereas all other classes seem to show lot of overlap, the confusion matrix also shows that not so good classification is achieved. To examine few digits for their mis-classifications, Figure 7 shows predictions of few individual digits. The figure with 4s and 9s mis-classifications clearly shows the network did not capture the features quite yet as most of those seem to be human readable.

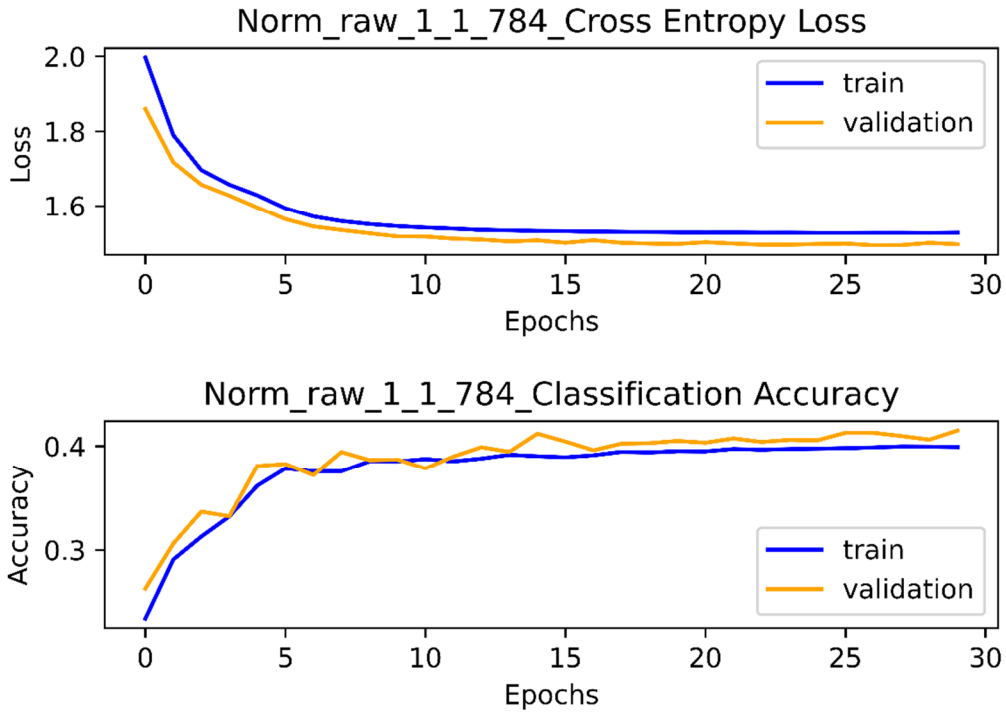

Figure 5. Training and validation accuracy and loss of single node network

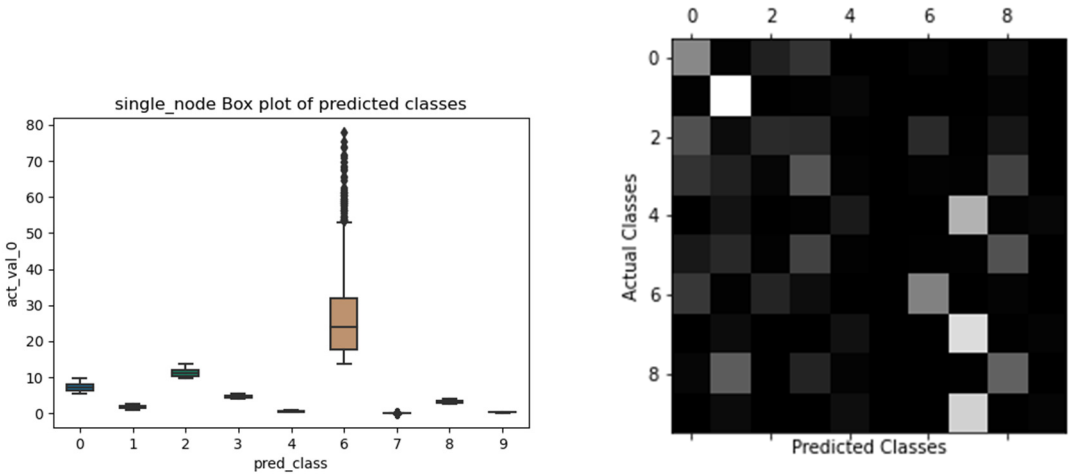

Figure 6. Box plot & confusion matrix of training data classifications

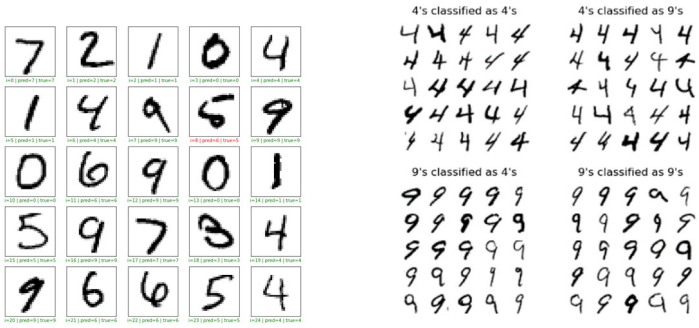

Figure 7. Example predictions using single hidden node network

Experiment 2 – single hidden layer with two node model:

The activation values from two hidden nodes are extracted and their box plots are shown in Figure 8. It is quite encouraging to see the emerging non-overlapping classes. Since it is 2-D data, boxplots are hard to perceive the impact from both activations.

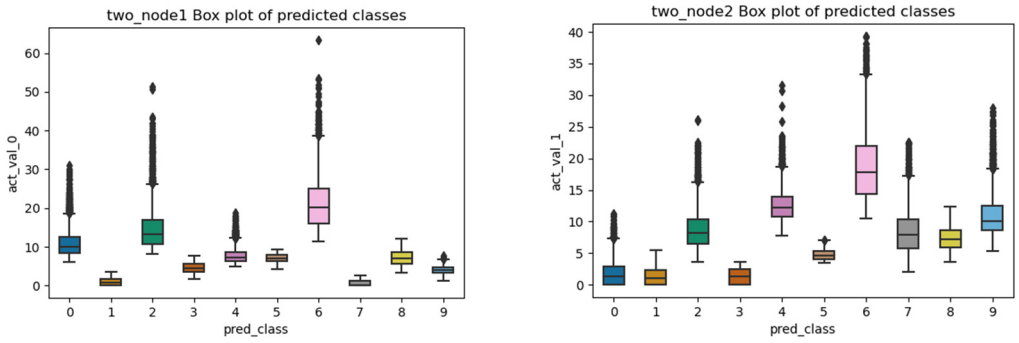

Figure 8. Box plots of two activation values

Figure 9 shows the scatter plot of 2 activation values, it is quite interesting to see the class boundaries. It is fascinating to see by just adding another node, it was able to perform better than a random guess with stable metrics. Since the accuracy is still in lower 60s further architectures with more nodes are explored in the below section.

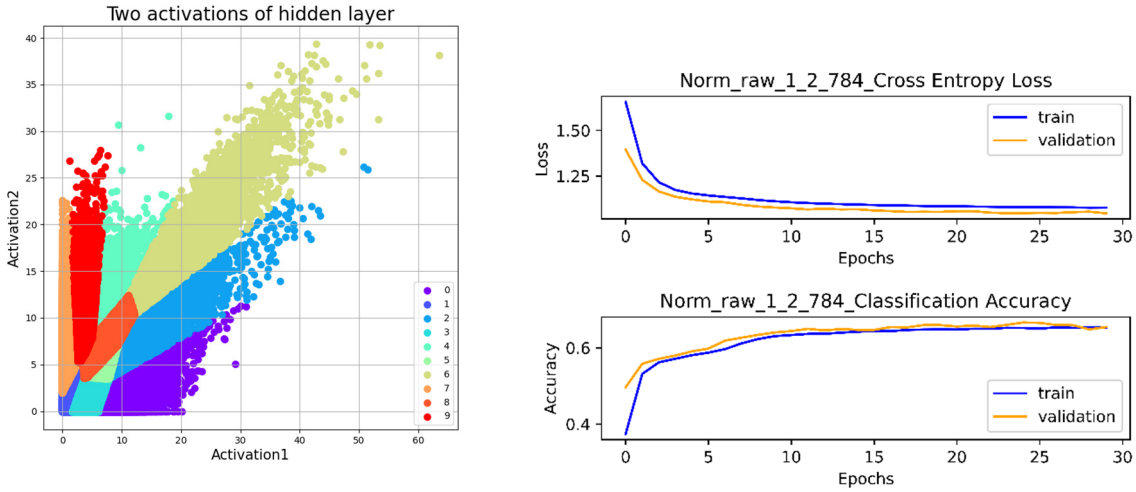

Figure 9. Scatter plot of activations and classification performance metrics

Experiment 3, 4 & 5 – raw data, reduced dimensions with PCA & Random Forest as inputs

As shown in methods section numerous experiments are conducted using single and 2-layer networks with various dimensions. Due to high dimensions of activation values, it is not possible to use simple box plots and scatter plots to decide on best model. The train, validation and test performance metrics along with computation times are recorded and top three models in each metric are highlighted in green, yellow and red as shown in below table.

exp_key	test_accuracy	train_accuracy	train_loss	train_time	validation_accuracy	validation_loss
Norm_raw_1_238_784	0.9816	0.9924	0.0266	0 days 00:01:39.460938000	0.9759	0.1458
PCs_1_238_200	0.9814	0.9952	0.0179	0 days 00:01:29.842088000	0.9774	0.1422
Norm_raw_1_543_784	0.9794	0.9939	0.0214	0 days 00:04:39.667686000	0.9774	0.1673
Norm_raw_2_234_784	0.9793	0.9925	0.0268	0 days 00:01:47.833961000	0.9754	0.2342
Norm_raw_2_128_784	0.9782	0.9912	0.0304	0 days 00:01:26.477400000	0.9734	0.2246
Norm_raw_1_128_784	0.9766	0.9899	0.036	0 days 00:01:13.959968000	0.9749	0.1378
PCs_1_128_200	0.9742	0.9932	0.0252	0 days 00:01:20.625937000	0.9738	0.1490
Norm_raw_2_54_784	0.9726	0.9862	0.0476	0 days 00:01:11.987224000	0.9706	0.1813
Norm_raw_1_54_784	0.9699	0.9811	0.0678	0 days 00:00:59.895551000	0.967	0.1493
PCs_1_54_200	0.9685	0.9869	0.0485	0 days 00:01:08.773848000	0.9662	0.1494
RF_Features_1_238_70	0.9445	0.9482	0.1703	0 days 00:01:14.389509000	0.9375	0.2274
RF_Features_1_128_70	0.9389	0.9375	0.2071	0 days 00:01:01.615784000	0.9316	0.2405
RF_Features_1_54_70	0.9281	0.9174	0.2749	0 days 00:01:14.209579000	0.9188	0.2792
Norm_raw_1_2_784	0.6459	0.6199	1.1296	0 days 00:00:49.939737000	0.6334	1.0883
Norm_raw_1_1_784	0.4032	0.3773	1.573	0 days 00:00:48.332070000	0.387	1.539

Table 2. Performance metrics from various experiments

While it is tempting to select a model with highest test accuracy, it is extremely important to examine model stability and mis-classification rates to assess under/overfitting. Training network with reduced dimensions gives additional advantage of less complex model and computational performance during scoring. While the test accuracy with PCA 238 model is among the top 3 models, overfitting is observed in all three PCA networks through their performance plots.

While the models using reduced dimensions from random forest did not give accuracies as close to raw models (and did not meet our requirement), the stability of these models is noteworthy to consider if input data dimension has a constraint in the production system. As can be seen in Figure 10, PCA with 54 nodes has high over fitting, whereas RF model seem to be very stable.

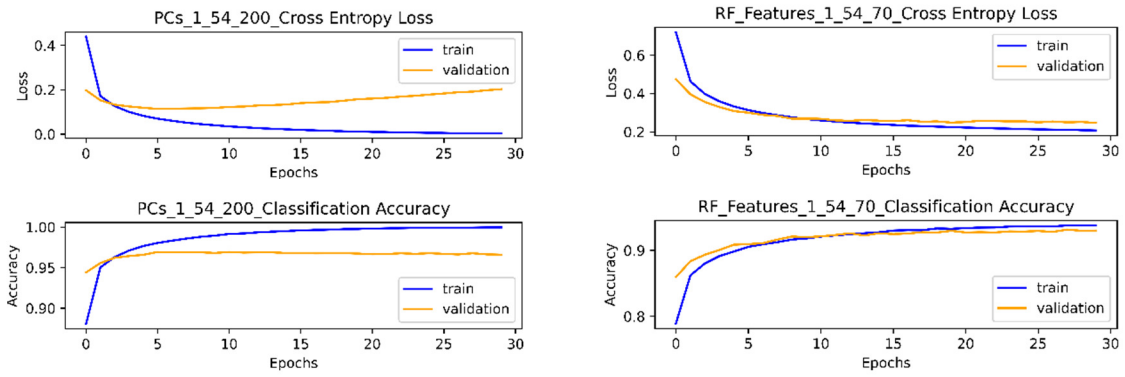

Figure 10. Performance metrics of models with PCA & RF features

Between the raw data models, two-layer models are ignored as single layer seem to perform better in terms of accuracies and relative stabilities. In order to qualify single hidden layer 238-node model as the best choice among competing models, further analysis is done comparing results from 54, 128, 238 node networks as shown in figures 11 & 12.

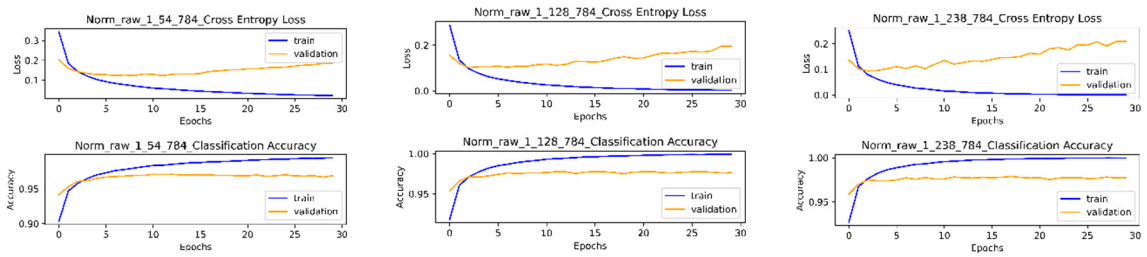

Figure 11. Performance metrics of 54, 128, 238 nodes networks with raw data

While the train and validation accuracies seem to be stable, some overfitting seem to be a common theme among all three models based on loss as shown in Figure 11. Figure 12 shows the separability of classes from 2 principal components of activation values from 54 & 128 networks. While 128-node PCs seem to show better separation, it is still hard to perceive in 2 dimensions.

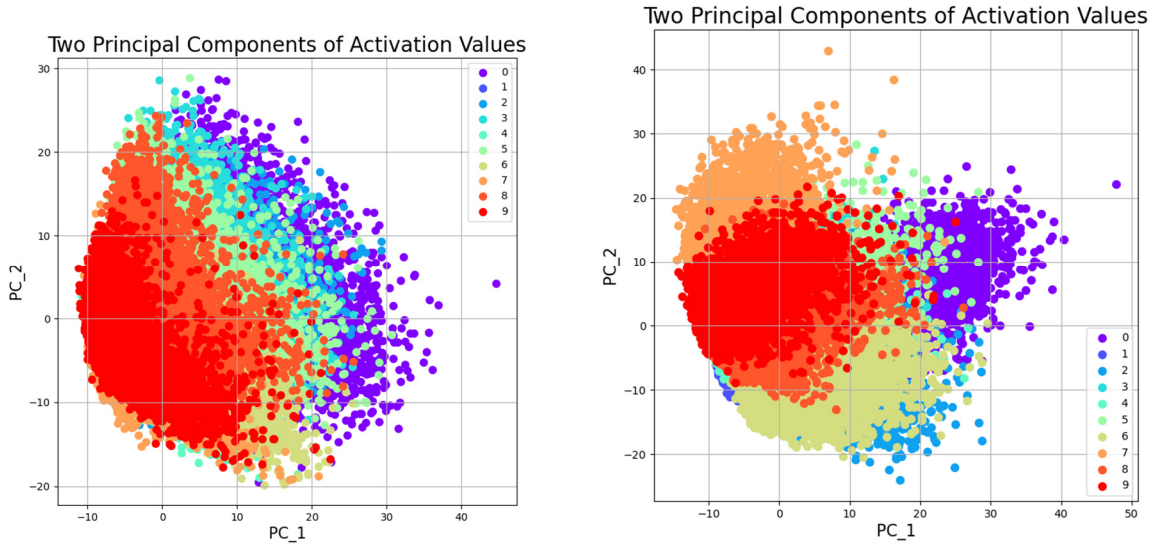

Figure 12. PCs of activation values from 54-node and 128-node networks

T-SNE visualizations are compared between raw data, 128-activations & 238-activations in 2-D. Clearly both the networks were able to detangle the overlap between 4s and 9s.

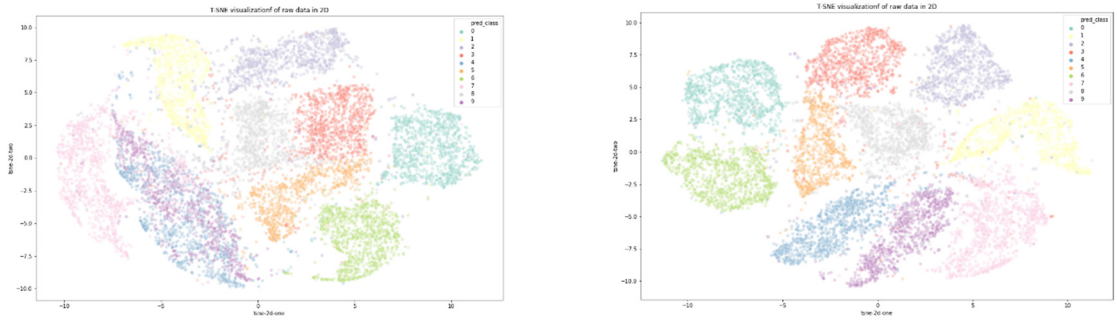

Figure 13. T-SNE of raw data and 128-node activations

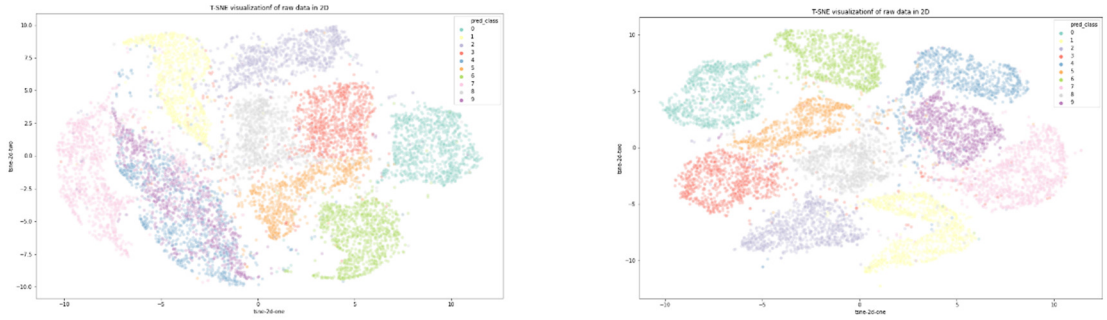

Figure 14. T-SNE of raw data and 238-node activations

Labels	True_counts	54_node_preds	128_node_preds	238_node_preds
0	980	0.9837	0.9908	0.9929
1	1135	0.9930	0.9947	0.9921
2	1032	0.9690	0.9593	0.9738
3	1010	0.9574	0.9802	0.9762
4	982	0.9705	0.9715	0.9695
5	892	0.9686	0.9630	0.9731
6	958	0.9676	0.9802	0.9823
7	1028	0.9679	0.9679	0.9689
8	974	0.9579	0.9764	0.9784
9	1009	0.9683	0.9752	0.9782

Table 3. Classification accuracies of each class from three networks

To further analyze performance of these 3 networks, the individual class classification rates on test data are compared as shown in Table 3. While 238-node model is the winner in majority of the classes, given relative better metrics of 128-node model (classification accuracy of digit 4, train & validation & test accuracies relatively close to each other, time of computation, overfitting) 128-node model is chosen as the winner.

6. **Conclusion:** In order to classify handwritten digit images from MNIST database for building an automatic classification system, set of DNN models are explored with requirements on 1. High test accuracy, 2. Reasonable computational performance with stability. While some overfitting is observed, given all other relative metrics compared to rest of the models explored (test classification accuracy of digit 4, train & validation & test accuracies being relatively close to each other, time of computation, overfitting), one hidden layer with 128 nodes network model (with ReLU & Sigmoid activation functions in hidden and output layers respectively) is recommended for deployment. The model performance need be closely watched post production due to the overfitting observed during training. In case of unstable performance, less accurate 54-node RF model with more stability may be deployed as a back-up.

Lessons Learned:

- Up-to a certain performance level (based on obvious features in the data), modeling with default parameters seems to be a trivial task. After that every point gain in performance metrics (accuracy, stability, computation) requires thorough experimentation, trade-off studies and judgement calls – DNN with complex parametric space makes it even harder to tune unlike other ML models (GLMs, RF etc)
- Lots and lots of visualizations and developing some intuitions around inner workings of these networks can help as guiding principles for setting up different architectures/tunables etc.

- Final recommendations require thorough evaluation of risks & documenting them
- All models are wrong, some are useful – George Box

References:

[1] MNIST Handwritten Digits Recognition by Jay Gupta in @TDataScience https://towardsdatascience.com/going-beyond-99-mnist-handwritten-digits-recognition-cfff96337392?source=social.tw

[2] Kumar, N., & Beniwal, H. (2018). Survey on Handwritten Digit Recognition using Machine Learning.

[3] http://yann.lecun.com/exdb/mnist/

[4] LeCun et al., Gradient-based learning applied to document recognition (1998), Proc of the IEEE, November 1998