CS F441
Selected Topics from Computer Science

Assignment – 1

Jonathan Samuel	2018AAPS0460H
Kshitij Ingle	2018A7PS0434H
Pratyush Banerjee	2018A7PS0312H

Introduction

We created 12 deep learning models and 5 learning models using traditional approaches (SVM, Logistic Regression, Decision Tree, Random Forest Classifier, Nearest Neighbors) for the classification of MNIST data. We performed a comparative study of these models and sensitivity analysis of the underlying hyperparameters.

Deep Feedforward Networks

Overview

S.No.	Layers	Nodes	Activation	Dropout	Train Accuracy	Validation Accuracy	Train Loss	Validation Loss
1	5	128	ReLU	Yes	0.9849	0.9792	0.0538	0.0791
2	5	128	ReLU	No	0.9955	0.9797	0.0152	0.0960
3	10	1024	ReLU	No	0.9944	0.9799	0.0253	0.2862
4	10	1024	ReLU	Yes	0.9729	0.9733	0.1452	0.2078
5	5	128	Sigmoid	Yes	0.9576	0.9670	0.1424	0.1113
6	2	128	ReLU	Yes	0.9893	0.9823	0.0313	0.0690
7	10	128	ReLU	Yes	0.9731	0.9733	0.1165	0.1237
8	5	32	ReLU	Yes	0.9392	0.9585	0.2349	0.1668
9	5	64	ReLU	Yes	0.9718	0.9727	0.0999	0.1054
10	5	1024	ReLU	Yes	0.9895	0.9798	0.0473	0.1352
11	2	32	ReLU	Yes	0.9446	0.9613	0.1808	0.1243
12	2	1024	ReLU	Yes	0.9956	0.9832	0.0156	0.0893

Fixed Hyperparameters

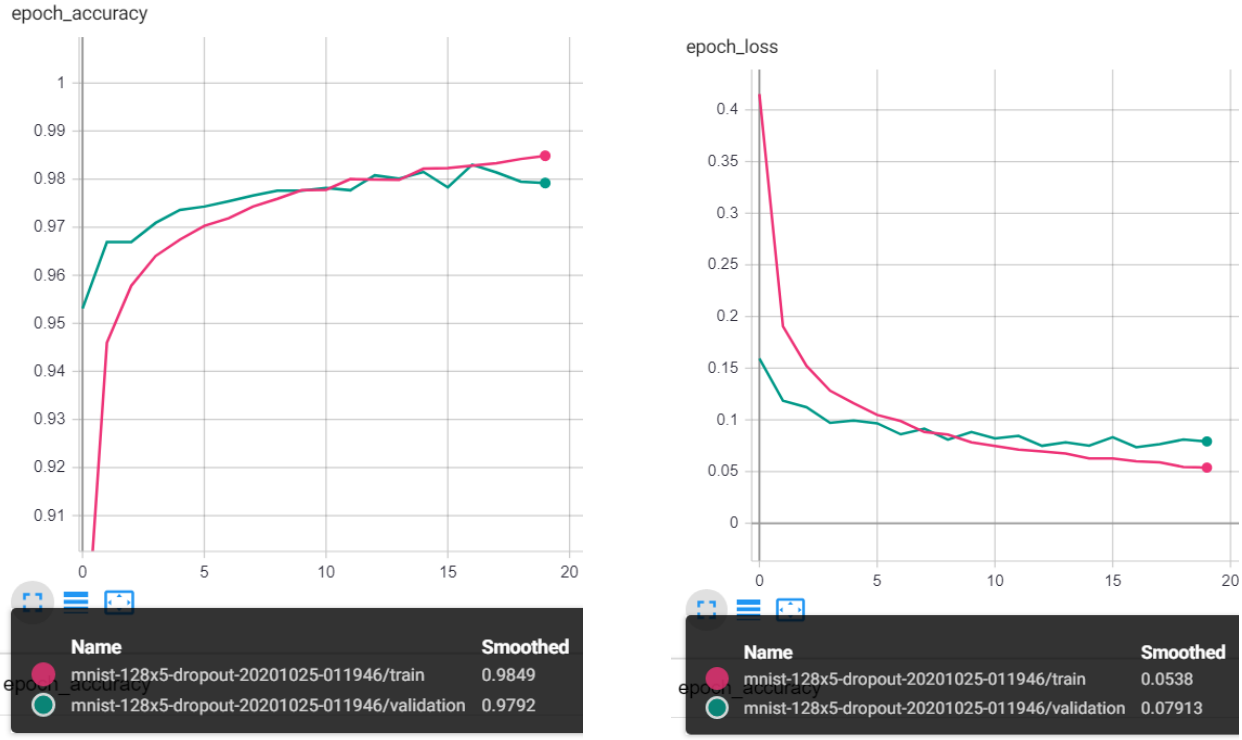
We have used the following hyperparameters in all the models:

- Adam optimizer
- Categorical cross-entropy loss function
- 20 training epochs

The majority of the models use Rectified Linear Unit as the activation function and Dropout for regularization.

Baseline model - 5 layers, 128 nodes, ReLU, 0.8 Dropout rate

We are going to use this model as our baseline model to aid in the comparison with other models. It achieves a good validation accuracy of 97.92%. Additionally, it has relatively low overfitting which is evident from the accuracy and loss curves for training and validation that are close to each other.

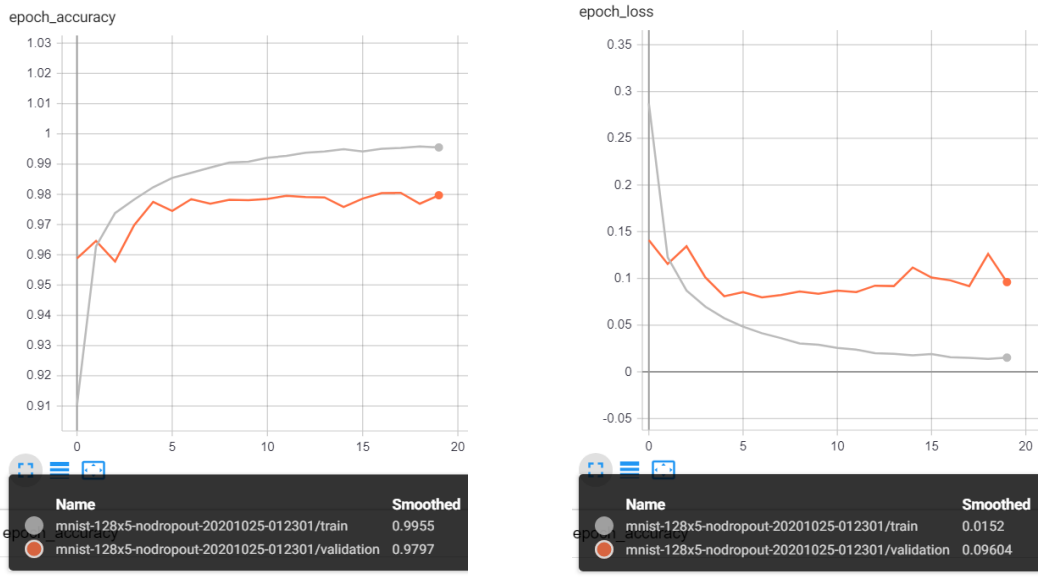


Regularization

Regularization is a technique which makes slight modifications to the learning algorithm such that the model generalizes better. This in turn improves the model's performance on the unseen data as well.

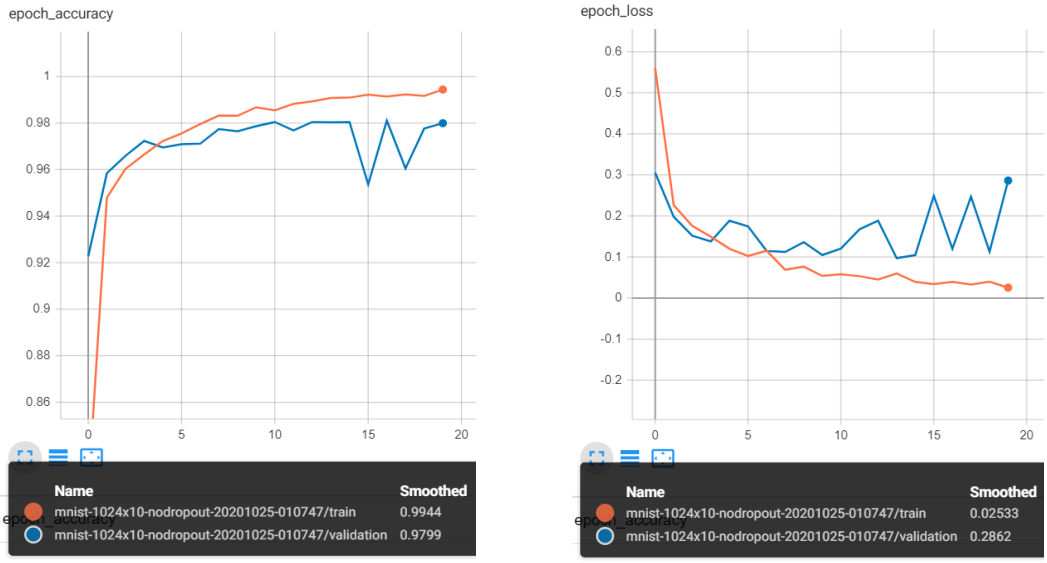
Dropout is a frequently used regularization technique that produces very good results. At every iteration, Dropout randomly selects some nodes and removes them along with all of their incoming and outgoing connections. This allows the model to capture more randomness. The probability of choosing how many nodes should be dropped is the hyperparameter of the dropout function.

5 layers, 128 nodes (without Dropout)



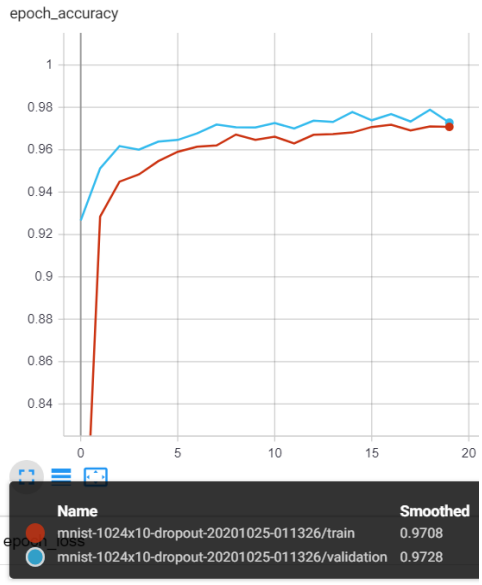
This model performs much worse compared to the baseline model. There is clear evidence of severe overfitting from the accuracy and loss curves. Training accuracy is nearly 100%.

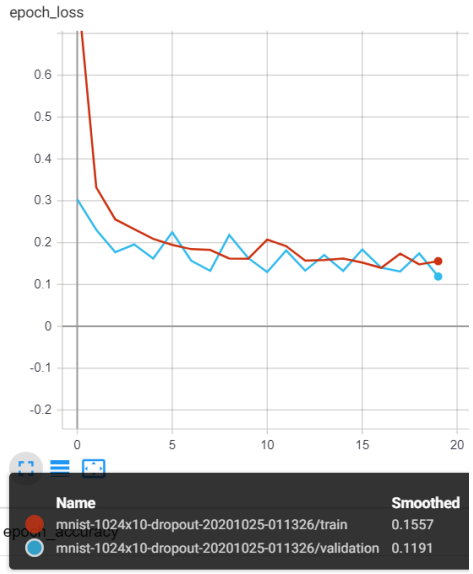
10 layers, 1024 nodes (without Dropout)



The effect of overfitting is even more prominent in this model. The validation accuracy curve is below the training accuracy curve and validation loss is above training loss, and increasing. Train accuracy again becomes nearly 100%.

10 layers, 1024 nodes (with Dropout)





When we introduce Dropout for regularization, the model is significantly improved. Overfitting is less evident from the accuracy and loss curves.

Due to these improvements, we have used Dropout in the subsequent models.

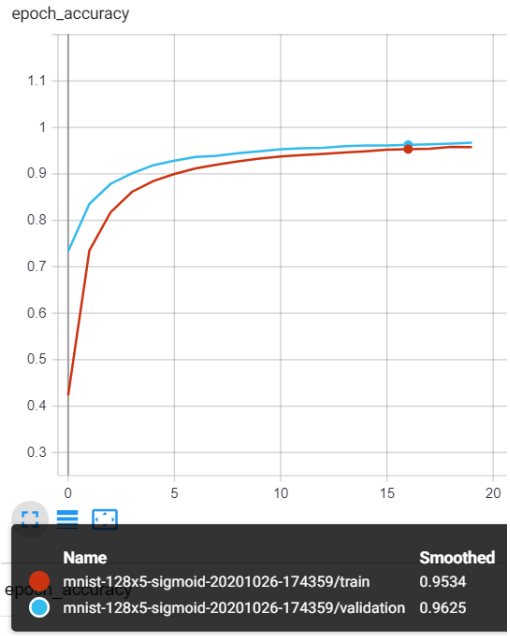
Activation Function

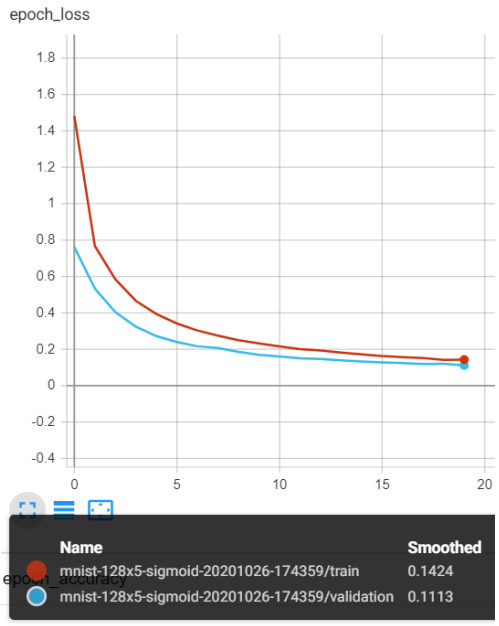
When there is no activation function, each neuron performs a linear transformation. Such a neural network is not powerful and will not be able to learn complex patterns from the data. Thus, we use an activation function to apply a non-linear transformation on the input of the neurons.

Sigmoid is a widely used non-linear activation function. However, it does not perform well with deep neural networks. The sigmoid curve saturates for large values of weights. The gradient is too small, slowing down learning.

Rectified Linear Unit (ReLU) has gained popularity as a non-linear activation function for deep neural networks. It is defined as f(x)=max(0,x). Since only a certain number of neurons are activated, the ReLU function is far more computationally efficient when compared to the sigmoid and tanh function. Moreover, it does not saturate for large values of weights.

5 layers, 128 nodes (sigmoid activation)





Compared to the baseline model with ReLU activation, this model has the very low accuracy and high loss value. This shows how sigmoid activation performs poorly in deep neural networks and is unsuitable for this purpose.

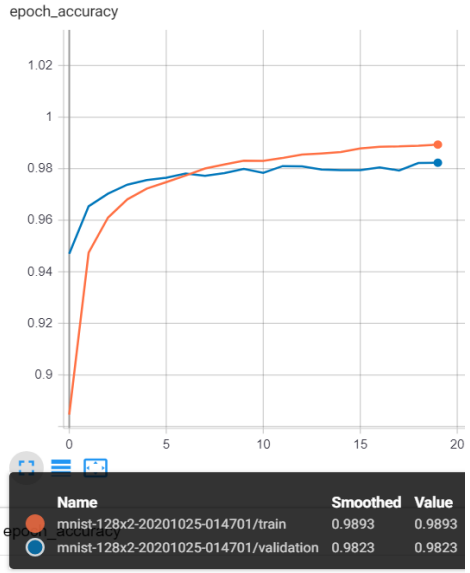
Therefore, we have used ReLU activation in the subsequent models as well.

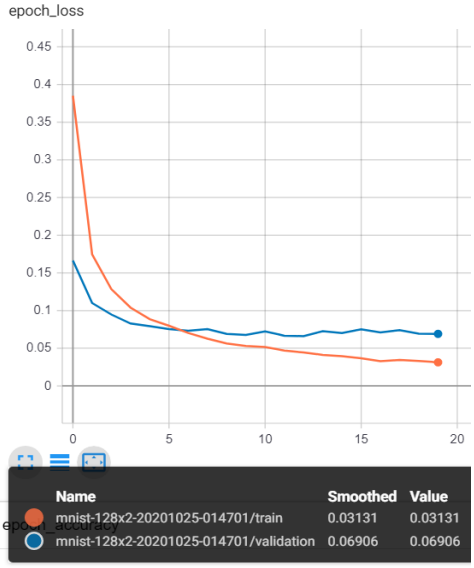
Network Depth

The depth of a neural network is determined by the number of hidden layers. We experiment with different numbers of layers to see how the depth of a network affects its performance.

Greater depth of a neural network allows deep learning to represent the world as a nested hierarchy of concepts, with each concept defined in relation to simpler concepts, and more abstract representations computed in terms of less abstract ones. It involves the study of composition of functions. Depth in a neural network lends itself to this idea.

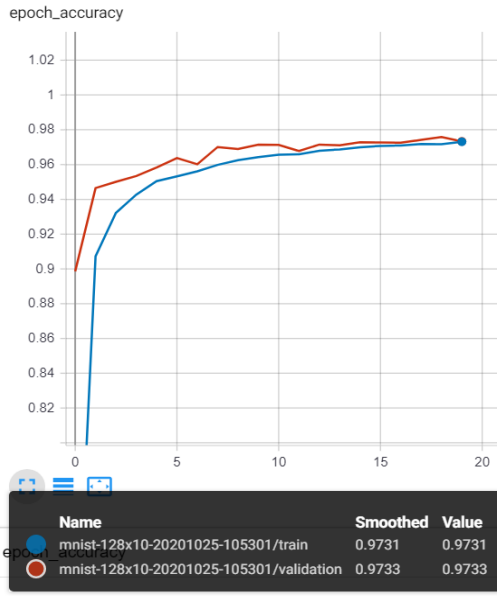
2 layers, 128 nodes

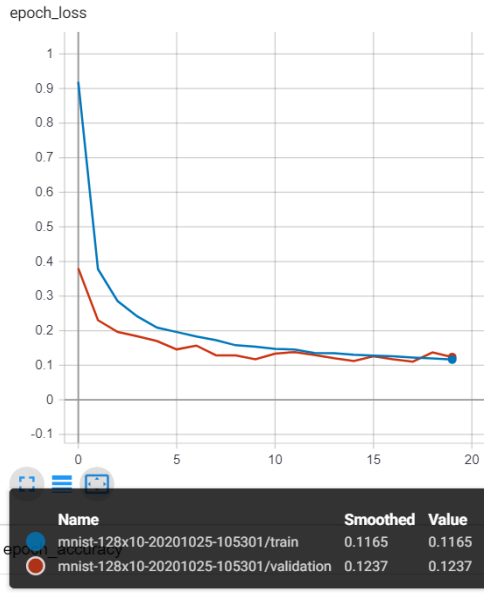




This model gives better accuracy compared to the baseline model which has 5 layers and the same number of nodes per layer. It also appears to converge to a minima in fewer iterations. Since the model has fewer parameters, it takes fewer iterations to train. The validation curve crosses the training curve in the earlier iterations so it can be said that the model is worse at generalizing.

10 layers, 128 nodes





This model gives worse accuracy compared to the baseline model. This can be attributed to the increasing complexity of the model with increased depth. It has a larger number of parameters so it takes more time to train. However, the graph shows that the model can generalize better since the validation curve only crosses the training curve in the later iterations.

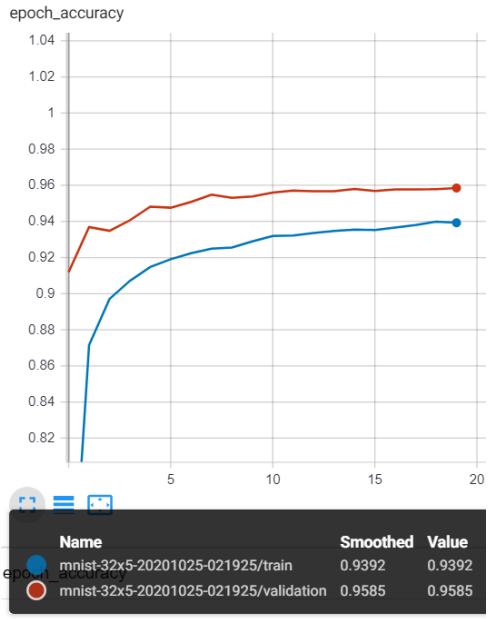
Overfitting

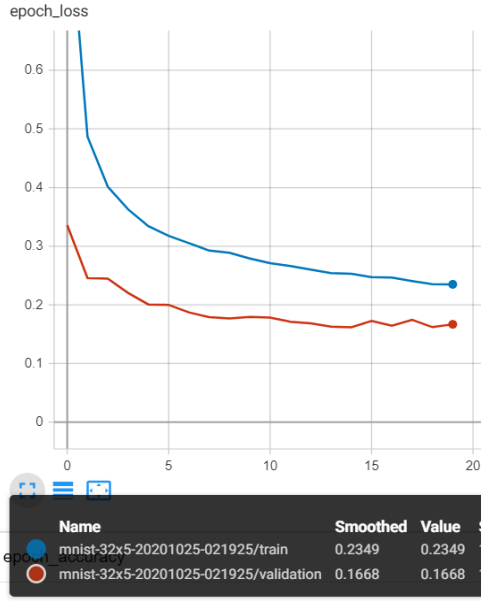
Adding more layers appears to have decreased accuracy. Deep models provide greater flexibility that could increase the ability to make predictions. However, there is a tradeoff between model complexity and ability to generalize. Increasing depth beyond a certain threshold tends to overfit to the training data and thus decrease the accuracy. The best examples are the models without Dropout

Layer Width

The number of nodes in each hidden layer is called the width of the layer. As with adding more layers, making each layer wider increases the total number of tunable parameters. We compare a few neural networks with different layer widths in this section.

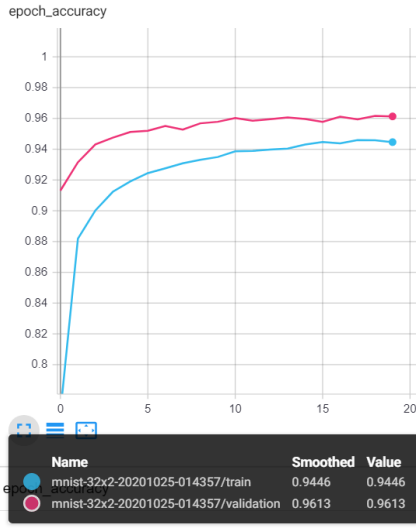
5 layers, 32 nodes

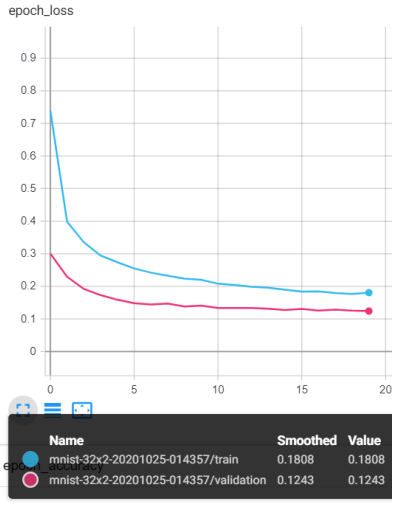




This model has the worst accuracy and highest loss of all the deep learning models. The width of the model is insufficient for learning which leads to loss of information as the data goes from layer to layer. This model seems to have severe underfitting.

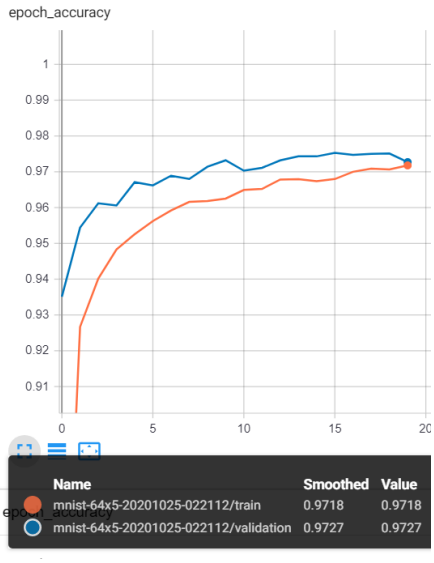
2 layers, 32 nodes

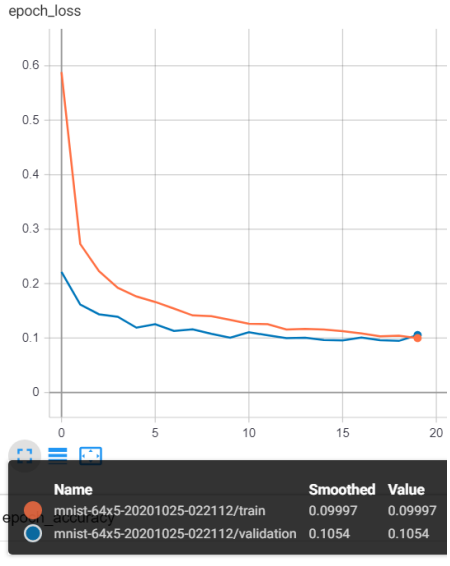




The loss of information due to the small width of layers is less pronounced when there are only 2 layers. This shallow model also seems to have severe underfitting like the previously mentioned model. However, the accuracy is slightly better and the loss is slightly lower compared to the earlier model.

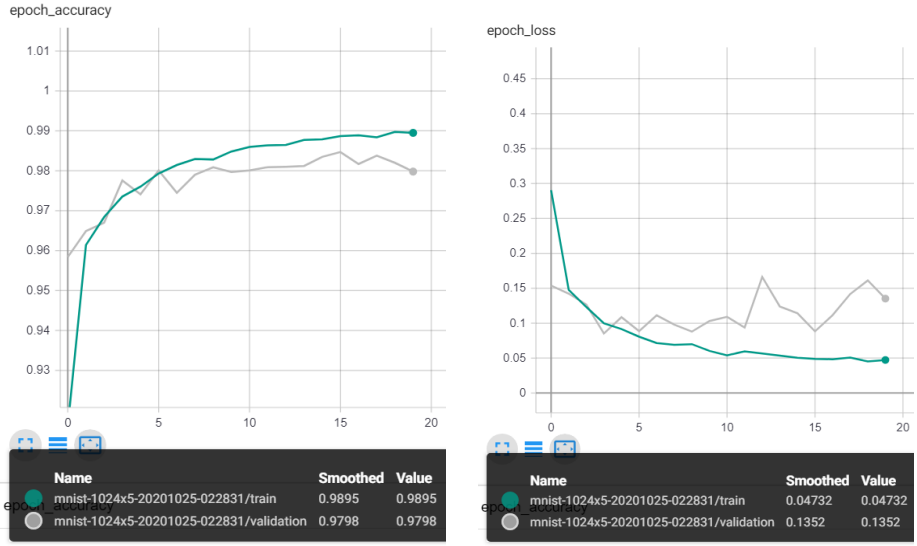
5 layers, 64 nodes





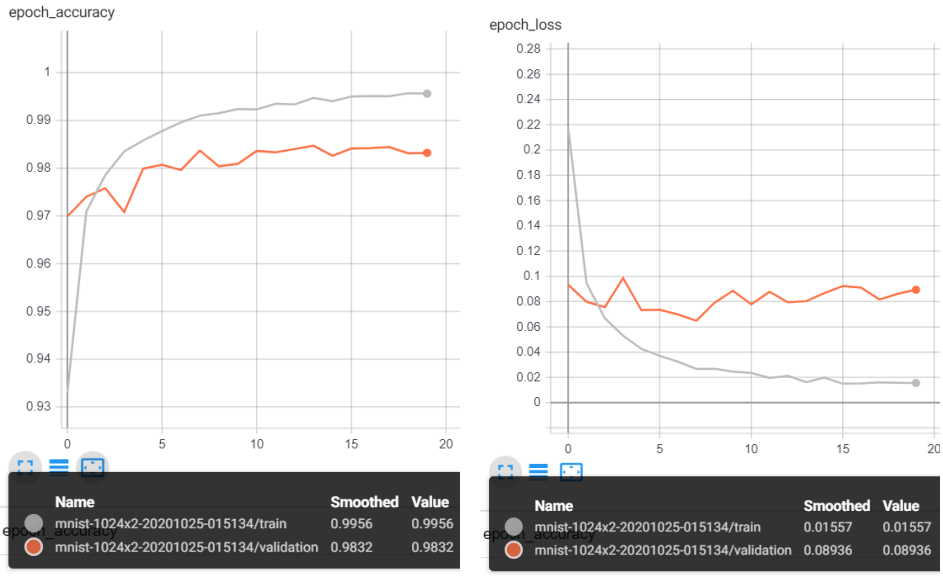
This model has accuracy and loss comparable to the baseline model with 5 hidden layers and 128 nodes per layer. However, there is less evidence of overfitting as can be seen from the train and validation curves only coming close near the last iterations.

5 layers, 1024 nodes



This model shows severe overfitting. The validation accuracy is much lower than the training accuracy and the validation loss is much higher than training loss, with an upward trend. Having too much layer width results in overfitting.

2 layers, 1024 nodes



This model has better accuracy than the previous one. However, the overfitting is even more prominent here as the difference between the validation and training curves is clearly visible. A shallow model with large width might give good accuracy but it is bad at generalizing. Training accuracy is almost 100%.

Thus, we have observed that increasing the width of the model generally maps to better performance on the validation data.

Traditional Models

Overview

S.No.	Model	Accuracy	Execution Time (sec)
1	Support Vector Machine	0.9216	6.23
2	Logistic Regression	0.9234	14.35
3	Decision Tree	0.8723	24.32
4	Random Forest Classifier	0.9680	45.22
5	Nearest Neighbors Classifier	0.973	840.85

Support Vector Machine

Support Vector Machine produces significant accuracy with less computation power. In a SVM model, the dataset is represented as points in space. The space is separated in clusters by several hyperplanes. Each hyperplane tries to maximize the margin between two classes (i.e. the distance to the closest points is maximized).

We have used LinearSVC which implements “one-vs-the-rest” multi-class strategy, thus training k models (where k is number of classes). It only considers a linear kernel. Other SVM implementations are slow and not scalable.

SVM gives an accuracy of 92.16% with the best execution time.

Logistic Regression

Logistic regression is a probabilistic, linear classifier. Classification is done by projecting an input vector onto a set of hyperplanes, each of which corresponds to a class.

We have fit a multinomial logistic regression with L1 penalty on the MNIST data. We use the SAGA algorithm that is fast when the number of samples is significantly larger than the number of features.

Logistic Regression gives an accuracy of 92.34% with second best execution time.

Decision Tree

A decision tree is one of most frequently and widely used supervised machine learning algorithms. A decision tree contains at each vertex a "question" and each descending edge is an "answer" to that question. The leaves of the tree are the possible outcomes. A decision tree can be built automatically from a training set.

Decision Tree gives the worst accuracy of all the models discussed in this section, 87.23% with the third best execution time.

Random Forest Classifier

Random forests are an ensemble learning method that can be used for classification. It works by using a multitude of decision trees and it selects the class that is the most often predicted by the trees.

Each tree of the forest is created using a random sample of the original training set, and by considering only a subset of the features. The number of trees is controlled by cross-validation.

Random Forest Classifier gives the best accuracy of all the models discussed in this section, which is 96.8%. It has the fourth best execution time.

Nearest Neighbors Classifier

The nearest neighbour classifiers use some or all the patterns available in the training set to classify a test pattern. These classifiers essentially involve finding the similarity between the test pattern and every pattern in the training set.

In k-Nearest Neighbor Classifier the test sample is allotted the majority class of all the k closest training samples. If $k = 1$, then the test sample is simply assigned to the class of that single nearest neighbor among the training sets.

The k-NN Classifier gives us an accuracy of 97.3% where $k = 5$. This is the best accuracy we get among all the other classical ML models. Although this model takes a lot of time to classify the testing set. Thus this takes the longest time to execute among all the models.

Deep Feedforward Networks vs Traditional Models

For classification of MNIST data with high accuracy, Deep Feedforward Networks consistently outperform the traditional models discussed here. The best accuracy using a traditional model is using Nearest Neighbors Classifier which gives an accuracy of 97.3% with a very high execution time. Deep Feedforward Networks give around 98% accuracy at approximately a fifth of that execution time.

Random Forest Classifier has lower accuracy than Nearest Neighbors Classifier at 96.8% but it is almost 20 times faster. Compared to Deep Feedforward Networks, Random Forest Classifier trades some accuracy for better speed.

When speed is more important than accuracy, traditional models like SVM and Logistic Regression are good options.

Note: Deep Feedforward Networks were implemented using Keras which uses GPU. On the other hand, traditional models were implemented using Sklearn which only uses CPU.

Conclusion

We conclude that deep learning models are very powerful and versatile. They outperform traditional machine learning models in most cases.

By choosing appropriate width and depth, along with suitable activation functions and regularization, we can build models that have good performance and generalize well.