Introduction to Deep Learning

Mohammadreza Akhtari

2024

Contents

Introduction

To perform the task, required libraries are first imported at the beginning of the code. Pandas is used for saving data and making tables. TensorFlow is used for training the model with various parameters. Two libraries of matplotlib and Seaborn are used for visualisation.

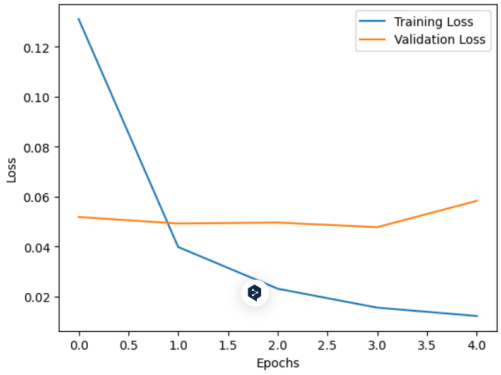
MNIST data containing image datasets of handwriting numbers is loaded using TensorFlow having a 60.000 training size and 10,000 test size. The sample of the dataset can be seen below:

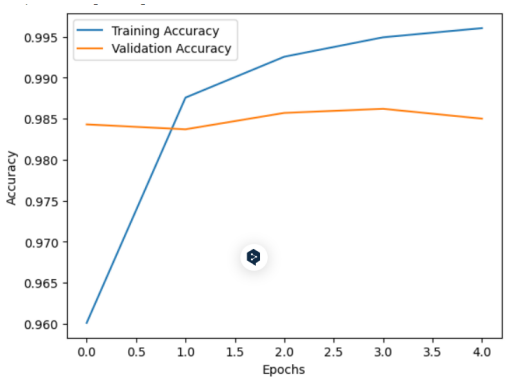
Data is then normalised to remove bias in input data and each data has the same influence on the training and output. Images are defined with numbers between 0 to 255, which are divided by 255 to have normalized data between 0 and 1. Then the number of epochs is important. Each epoch means going through the whole data set for training. If the number of epochs is low, it leads to underfitting and not capturing the main features in the dataset. If it is high, it leads to overfitting or memorizing the data. However, there is a computational limitation which does not allow experiencing large epochs. Therefore, 5 epochs at the end are selected for final calculation.

CNN

Here, a model with 2 convolutional neural networks with relu activation function, the size of 32 and kernel size of (3,3), a hidden layer and an output layer of 10 with softmax activation function for classification is made. TO compile a model, Adam optimizer, sparse_categorical_crossentropy loss function and accuracy metrics for performance

evaluation. The model leads to an accuracy of 0.98500. The results can be seen graphically below.



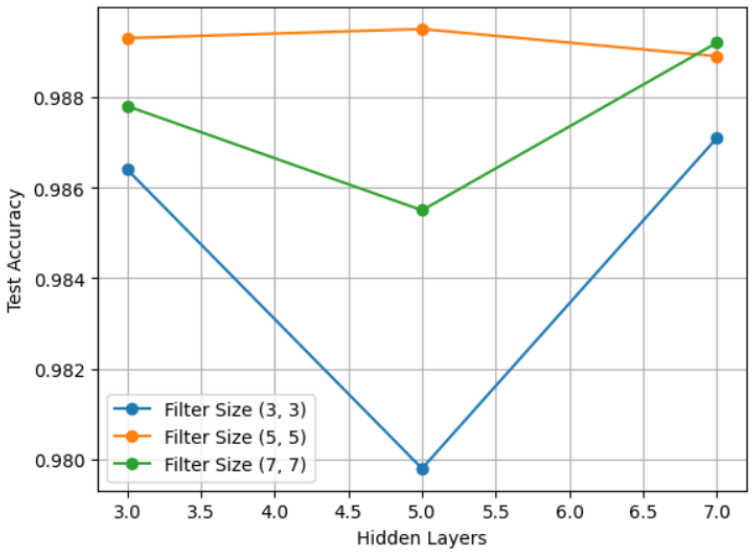


The size and depth

The number of hidden layers increased to 3, 5 and 7 while keeping other parameters the same. The kernel size is also increased considering the three various sizes of (3, 3), (5, 5), (7, 7).

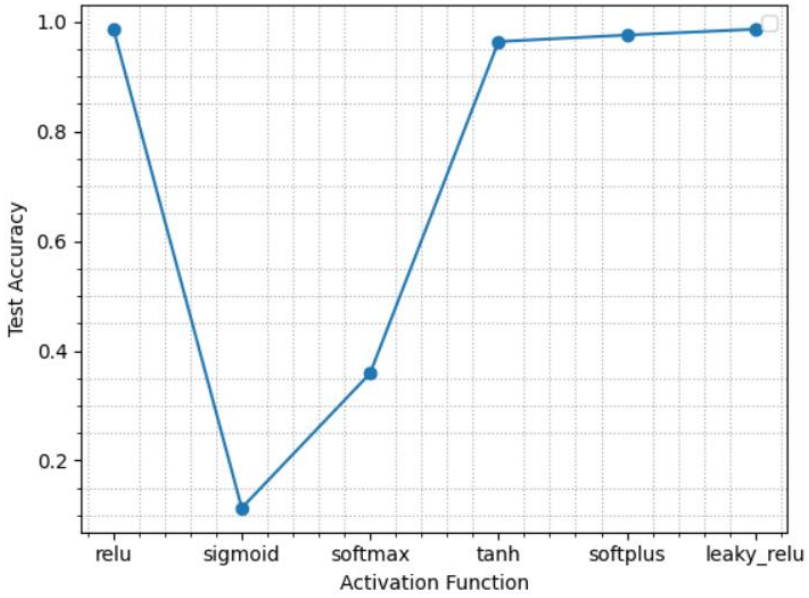
As the number of hidden layers increases, the model could capture more complex features and it will be prone to overfitting. It is important to find the appropriate number of hidden to capture the features but not memorizing the data. If the layers are too small, it is not able to capture the main features of training data.

Similarly, a larger size of filter or kernel leads to overfitting and the small size of kernel could not capture the main features in the dataset. It is also important to find the appropriate number and size of layers and depth considering overfitting, underfitting, computational limitation, and complexity of data. As could be seen the small size of kernel has the lowest accuracy and it increase by increasing its size. As the hidden layers also increase, the accuracy tends to increase getting near 1 due to the overfitting. There is a reduction point at the medium size of hidden layers which could be that the small number of hidden layers did not capture the features' complexity and by increasing the hidden layers, features are extracted leading to the lower amount of accuracy. This accuracy increases again as the model tend to overfit the data by increasing the hidden layers.



Various activation functions

To see the effect of activation layers, different ones are employed namely 'relu', 'sigmoid', 'softmax', 'tanh', 'softplus', and 'leaky_relu'.

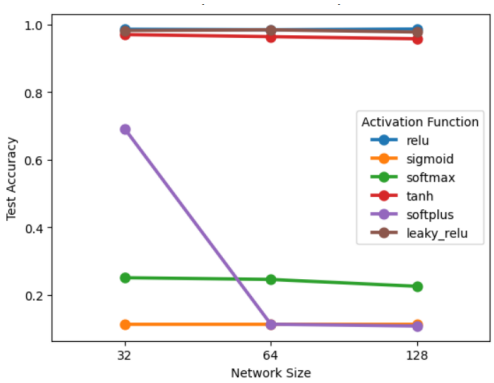


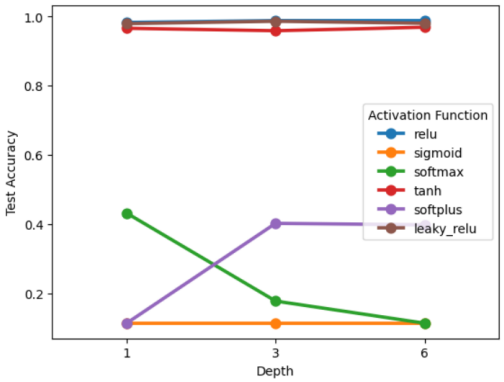
Among these 6 activation functions, relu performs the best giving an accuracy of 0.9866 and sigmoid performs the worst having an accuracy of 0.1135. Relu is a typical function which is widely used it is worth mentioning that the activation function could add non-linearity to the model to make it more powerful to capture complex features. The sigmoid function is usually used for binary classification having 0 and 1 datasets. Softmax also is usually used for multi-class classification and this function may suffer from capturing non-linearity.

Activation function, network sizes, and depths

Although it takes a lot of time and sometimes there is a limitation for using Google Colab, previous activation functions ('relu', 'sigmoid', 'softmax', 'tanh', 'softplus', 'leaky_relu') are considered with layer sizes of 32, 64, and 128 as well as 1, 3, and 6 numbers of layers.

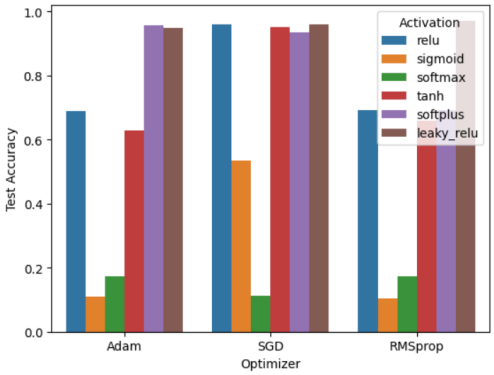
Generally, the accuracy gets better as the depth of the model or number of layers increases making it possible to capture more complexity and features. However, it should be noted that the large number of layers may lead to overfitting or memorization of data, which should be opted based on the complexity of data on experience. If the model overfit, it performs the best on the training dataset but poorly on the unseen dataset. Similar things could be seen and explained for the layer size.

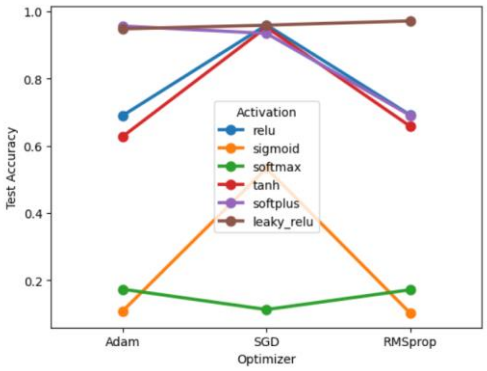




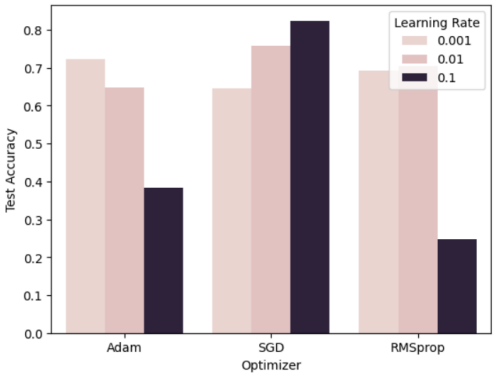
Various optimizers and learning rates

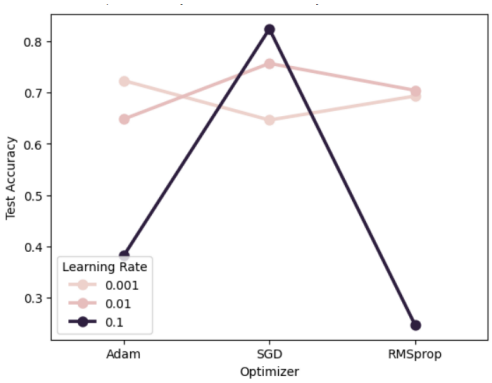
Three learning rates of 0.001, 0.01, and 0.1 as well as previous activation functions of 'relu', 'sigmoid', 'softmax', 'tanh', 'softplus', and 'leaky_relu' are considered for this task. Due to the computational limitation, three different optimizers are employed namely Adam, SGD, and RMSprop. Some optimizer works better based on the problem and activation function. SGD seems to work generally well and the leaky_relu activation function seems to be insensitive to various optimization methods. Here, the leaky_relu activation function works better than others considering all optimizer





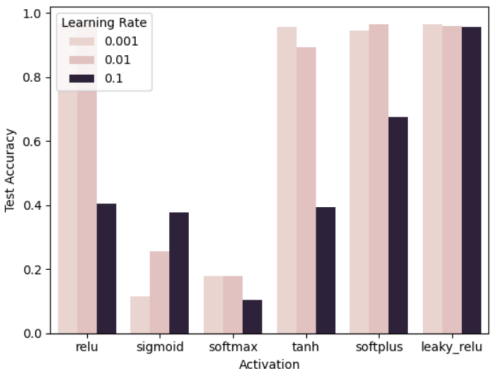
Regarding learning rate, the lower amount of learning rate will increase the computation time but more reliable results. Graphs show that the higher amount of learning rate of 0.1 is less stable as it jumps from one local minimum to another and makes it less effective. Both 0.001 and 0.01 seem to work nicely and the higher one or 0.01 is preferable as reduces the computational effort.

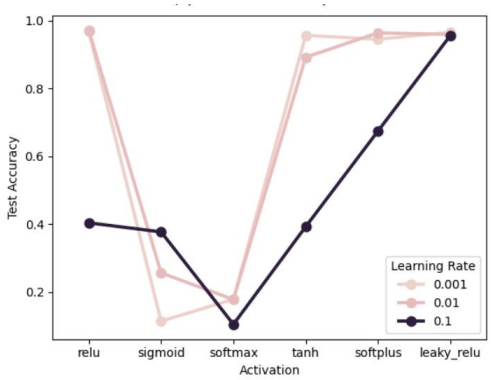




Another important issue which should be considered is that learning rate could affect various activation functions differently. The relu and tanh functions are highly affected by the amount of

learning rate while leaky_relu activation is the least affected function by learning rate followed by softmax. Therefore, it is important to select the appropriate learning rate to achieve reliable and stable results with reasonable accuracy.

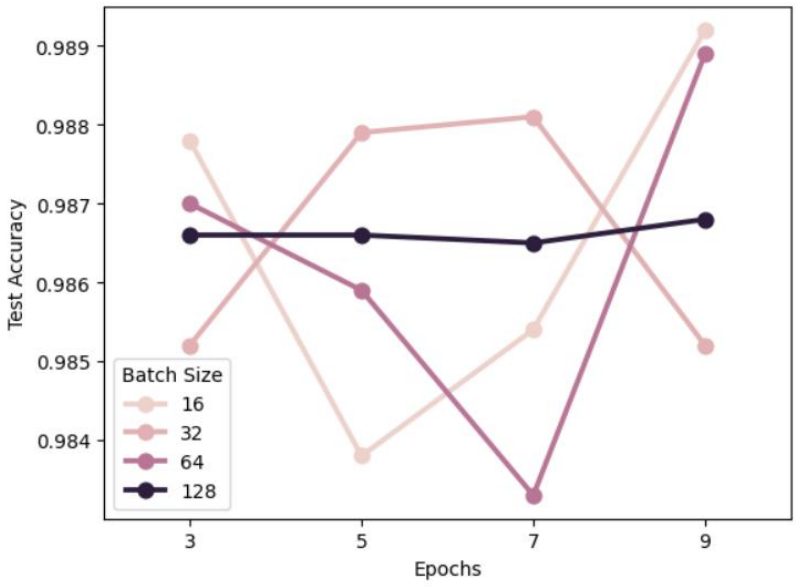




Various batch sizes and epochs

Batch sizes of 16, 32, 64, and 128 as well as epochs number of 3, 5, 7, and 9 are considered for training in this part. The appropriate number of epochs is necessary to capture the features and complexity of data. Each epoch means that the model goes through the whole training set and it leads to overfitting and optimistic results if it increases inappropriately. Batch size indicates the number of samplings in each epoch. The large batch size makes the training process faster and captures more general features. On the other hand, making a batch size small leads to the capture of noises and every single feature. It should be noted that a very large batch size requires higher memory capacity.

As can be seen in the following graph, the large batch size of 128 is more stable as it captures more general features with less effect of noise. The smallest batch of 16 has the most variation and the highest accuracies at the high epochs as it captures almost all features and noises. generally as also discussed earlier, the increased number of batches enhance the accuracy due to the overfitting.



No CNN layer

The model consists of an input layer, dense layer and output layer without using any convolutional neural network layer. The training is done for two epochs of 2 and 4 as well as various batch sizes of 32, 64, 128, and 256 together with layer sizes of 32, 64, 128, and 256.

Results indicate the better performance of the model with a higher number of epochs and layer size. Although accuracies are high and almost near each other, the model with CNN has better accuracy compared to the one without CNN to the point that accuracies struggle to reach one with CNN even with a higher number of epochs and dense layer size.