FTDL-Part II

Deep Learning with Tensor flow-MNIST

Kush Gupta

1 Introduction

For the following submission, i explored the MNIST dataset and built a classifier using tensor flow.

2 Try a hidden layer size of 200. How does the validation accuracy of the model change? What about the time it took the algorithm to train? Can you find a hidden layer size that does better?

The table below show the different combinations of hidden layers and number of epochs used to train the model and the corresponding results obtained for each run.

Hidden layers	Num epochs	val accuracy	Val loss	Train accuracy	Test accuracy	Test loss
50X2	5	0.9693	0.0973	0.9719	0.9622	0.1192
50X2	10	0.9870	0.0361	0.9920	0.9772	0.0864
100X2	10	0.9903	0.0293	0.9927	0.9775	0.0824
200X2	10	0.9943	0.0221	0.9945	0.9811	0.0756
300X2	50	0.9983	0.0057	0.9983	0.9823	0.1353

I tried a hidden layer of size 200 and could see that for 10 epochs the validation accuracy increased to 99.03% from 98.7% and the validation loss decreased. It took around (3 to 5) seconds more to train the model, however, the difference was not much noticeable. To obtain better results, i tried two models of hidden layer size 400 and 600. I observed that for hidden layer size 600, the validation accuracy of the model was 99.83% and the validation loss dropped significantly to 0.57%. Also, the train accuracy was 99.83% and test accuracy was 98.23%.

3 The *depth* of the algorithm. Add another hidden layer to the algorithm. This is an extremely important exercise! How does the validation accuracy change? What about the time it took the algorithm to train? Hint: Be careful with the shapes of the weights and the biases.

When i used a model, with 3 hidden layers i.e total size is 100X3 and the number of epochs was 10. It took around 1 minute 25 seconds to train the model. I observed that the validation accuracy was 99.15%, it was **increased** by 0.12% and the validation loss was 3.04%, it **increased** by 0.11%. Also, the Train accuracy was 99.19%, it **decreased** by 0.08% and test accuracy was 97.53%, it **decreased** by 0.22% when compared to results obtained with a model of 200 hidden layers.

Below table shows the comparison between the model having 200 and 300 hidden layers.

Hidden layers	Num epochs	val accuracy	Val loss	Train accuracy	Test accuracy
100X2	10	0.9903	0.0293	0.9927	0.9775
100X3	10	0.9915	0.0304	0.9919	0.9753

4 The *width and depth* of the algorithm. Add as many additional layers as you need to reach 5 hidden layers. Moreover, adjust the width of the algorithm as you find suitable. How does the validation accuracy change? What about the time it took the algorithm to train?

For a model of 5 hidden layers, i used 70 units. So, total size was 70X5=350. It took around 01 minutes 29.17 seconds to train the model, which is around 4 seconds more than the time to train a model of size 200 layers. The validation accuracy was 98.67%, validation loss was 3.84%, training accuracy was 98.71%, test accuracy was 97.69% and Test loss was 8.92%. The Validation accuracy decreased may be because the model over fitted.

5 Fiddle with the activation functions. Try applying sigmoid transformation to both layers. The sigmoid activation is given by the string 'sigmoid'.

I choose a model with hidden layer size of 100X2 and sigmoid as the activation function for both the layers. I got validation accuracy as 98%, validation loss as 6.75%, Train accuracy as 98.33%, test accuracy as 97.22% and Test loss as 8.69%.

6 Fiddle with the activation functions. Try applying a ReLu to the first hidden layer and tanh to the second one. The tanh activation is given by the string 'tanh'.

For a model of hidden layer size 100X2,i used 'relu' as activation function for 1st layer and 'tanh' for second layer. These are the results that i got- validation accuracy: 99.18%, validation loss:2.60%, Train accuracy: 99.39%, test accuracy:97.76% , Test loss:7.76%. It took around 47.768 seconds to train the model.

7 Adjust the batch size. Try a batch size of 10000. How does the required time change? What about the accuracy?

For a model of hidden layer size 100X3. I used a batch size of 10000 and it took around 27 seconds to train the model, which was the **fastest** till now. Below are the results that i got.

validation accuracy:91.45%, validation loss: 29.35%, Train accuracy: 91.49%, test accuracy: 91.78% , Test loss: 28.45%. I observed that the model accuracy dropped significantly.

8 Adjust the batch size. Try a batch size of 1. That's the SGD. How do the time and accuracy change? Is the result coherent with the theory?

For a model of hiddesn layer size 100X3 and batch size of 1. It took around 20 minutes and 38 seconds to train the model. The results obtained are: validation accuracy: 96.10%, validation loss:14.02%, Train accuracy: 96.34%, test accuracy:95.79%, Test loss:15.71%. The accuracy is reduced.

9 Adjust the learning rate. Try a value of 0.0001. Does it make a difference?

For a model of size 100x2 and a learning rate of 0.02,it took 45.432 seconds to train the model. The result obtained are validation accuracy: 46.17%, validation loss:20.51%, Train accuracy:43.69%, test accuracy:47.07% , Test loss:20.42%. The validation accuracy dropped significantly to half of the value of previously obtained results due to smaller learning rate.

Code snippet:
$custom_optimizer = tf.keras.optimizers.SGD(learning_rate = 0.0001)$
$model.compile(custom_optimizer, loss =' sparse_categorical_crossentropy', metrics = ['accuracy'])$

10 Adjust the learning rate. Try a value of 0.02. Does it make a difference?

For a model of size 100x2 and a learning rate of 0.02,it took 50.926 seconds to train the model.The result obtained are validation accuracy: 95.57%, validation loss:15.27%, Train accuracy:95.32%, test accuracy:95.46% , Test loss:15.77%. The validation accuracy increased as it dropped when we used a much smaller learning rate.

Code snippet:
$custom_optimizer = tf.keras.optimizers.SGD(learning_rate = 0.02)$
$model.compile(custom_optimizer, loss =' sparse_categorical_crossentropy', metrics = ['accuracy'])$

11 Combine all the methods above and try to reach a validation accuracy of 98.5+ percent.

I used a model with hidden layer size of 300x2 and 50 epochs. The batch size was 100 and the activation function used was 'relu', for both layers. I got a validation accuracy: 99.83%, validation loss:0.5%, Train accuracy: 99.83%, test accuracy:98.23% and Test loss:13.53%.

Final Results:

Hidden layers	Num epochs	val accuracy	Val loss	Train accuracy	Test accuracy	Test loss
300X2	50	99.83	0.57	99.83	98.23	13.53