MNIST data prediction using MLP, CNN and VGG16

Anas Abufardeh - B00873503
Manpreet Singh- B00853930
Prabhjot Kaur – B00843735
Shree Rama Kamal Kumar Vegu - B00872272

Faculty of Computer Science, Dalhousie University

Abstract

Neural Networks is one of the most efficient machine learning algorithms. One of the key advantages of neural networks is their ability to self-learn. This paper covers the implementation of MPL, CNN and the pre-trained VGG16 models to create a model for the MNIST data. It was evident that the MLP model results in a high train and test accuracy of 98% and 86% respectively. On the other hand, the CNN model results in a much lower accuracy of around 11-12%. To add, the accuracy of the VGG16 model was very impressive as it significantly increases with the number of trained layers. Therefore, one can conclude that pre-trained models are an important tool that helps in improving the efficiency and performance of the model.

1.Introduction

Neural networks are a set of algorithms that were inspired by the neural networks in the human brain. It is one of many different tools used in machine learning algorithms to process complex data. In general, neural networks do not require any human intervention or programming as they self-learn from a sufficient amount of data samples. This allows the model to set the appropriate features and define the output more efficiently. Increasing the number and variety of these samples results in more accurate results. However, this task is very costly in terms of time and computational power. As such, it is common to use pre-trained models to allow for faster and more efficient training. Pre-trained models are models that has already been trained on a large dataset to solve a specific problem. Therefore, one can import and use pre-trained models (e.g. Inception, VGG…) as a starting point for early layers. This allows the network to concentrate on training the higher layers that are specific to the new task. Here we show how neural networks are trained using MLP and CNN algorithms. We then explore how the VGG16 model which was pre-trained on ImageNet can be utilized to create a model for the MNIST data. The learning curves were generated for each case to study and compare the accuracy and performance of each model.

2.Methods

2.1 MLP Model

Taken the MNIST dataset with 60,000 training samples and 10,000 testing samples. Then reshaped it to 1024 training samples and 501 testing samples and did the experiments using different methods.
Firstly, used the MLP model using the five dense layers of each 128 neurons and activation function 'relu'.

2.2 CNN Model

Secondly, used the Convolution Neural Networks with the kernel size of 3*3 with activation function 'relu' and did max pooling with size 2*2 after that did the dimensionality reductions using dropout function with 0.25 and 0.5, in between also flattened the matrix to reduce the size of the image. For compilation used the loss function 'categorical_crossentropy' and 'Adadelta' optimizer. Subsequently fitted the model using training and test samples with batch size of 128, epochs of 12 and verbose of 1. Evaluated the test scores like loss and accuracy

2.3 VGG16 Model without retraining any layers

Took the example to Fine-tune InceptionV3 and optimized to VGG16 model which is trained on ImageNet and created a model for the MNIST reshaped data. This was achieved by using all the 18 existing layers.

2.4 VGG16 Model by retraining last three layers

Performed the similar experiment by changing the ratio of the layers that is using the first 15 existing layers and re-training the last 3 layers.

2.5 VGG16 Model by retraining last nine layers

Performed the similar experiment by changing the ratio of the layers that is using the first 9 existing layers and re-training the last 9 layers.
These are the different experiments done with the samples of MNIST dataset and observed the accuracy and loss scores.

3.Results

3.1 Using MLP Model on MNIST dataset

Table 1 Accuracy and Loss for training and test data

Train accuracy	*Train loss*	*Test accuracy*	*Test loss*
0.982	0.086	0.860	0.472

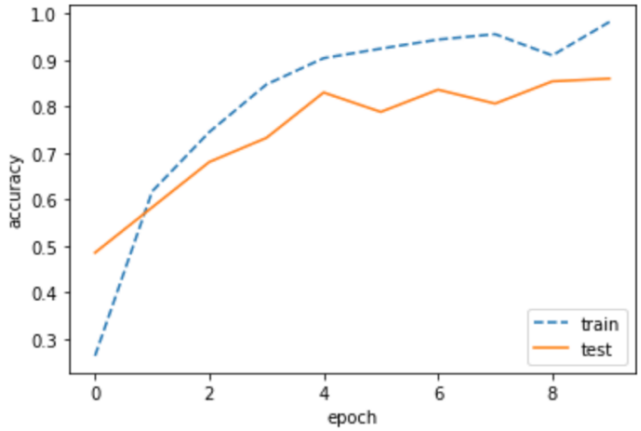

Figure 1: Learning curve for accuracy and epoch for train and test data

3.2 Using CNN Model on MNIST Dataset

Table 2 Accuracy and Loss for training and test data

Train accuracy	*Train loss*	*Test accuracy*	*Test loss*
0.1152	2.303	0.123	2.298

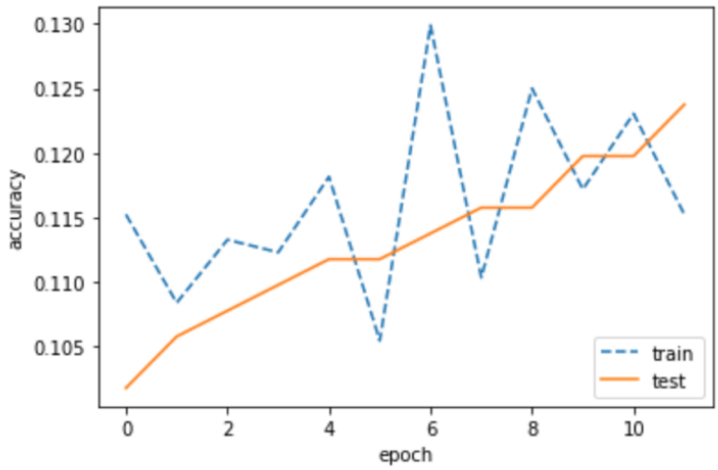

Figure 2 Learning curve for accuracy and epoch for train and test data

3.3 Using VGG16 Model on MNIST dataset without retraining any layers

Table 3: Accuracy and Loss for training and test data

Train accuracy	*Train loss*	*Test accuracy*	*Test loss*
0.852	0.6672	0.7944	0.7222

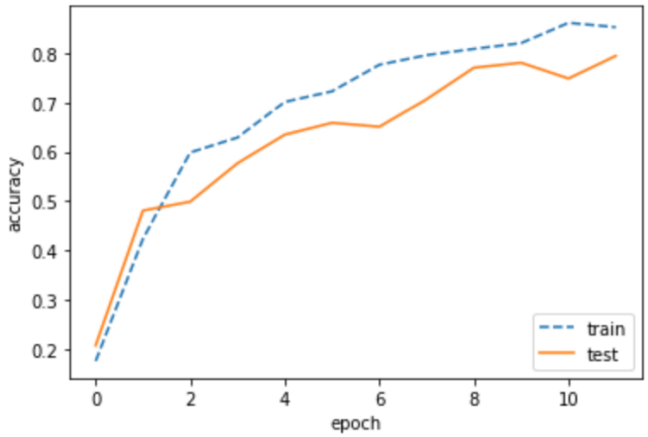

Figure 3 Learning curve for accuracy and epoch for train and test data

3.4 Predicting MNIST dataset using VGG16 pre-trained model, retraining last three layers

Table 4 Accuracy and Loss for training and test data

Train accuracy	*Train loss*	*Test accuracy*	*Test loss*
0.94	0.26	0.898	0.366

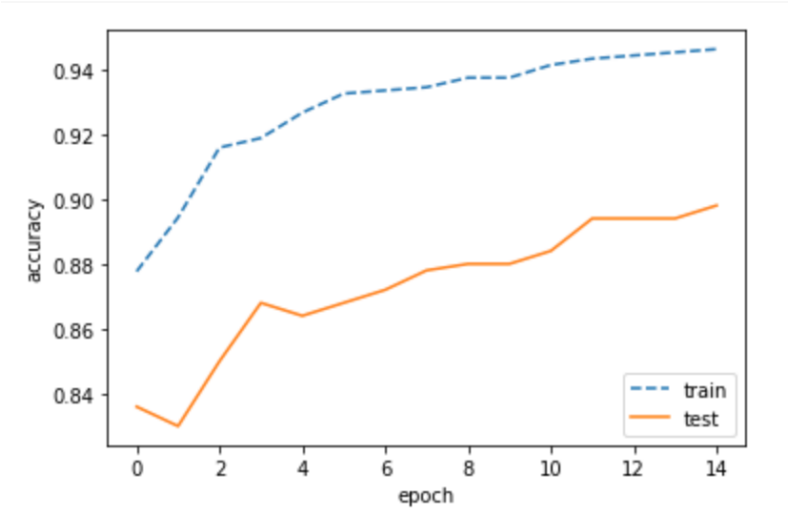

Figure 4 Learning curve for accuracy and epoch for train and test data

3.5 Predicting MNIST dataset using VGG16 pre-trained model, retraining last nine layers

Table 5 Accuracy and Loss for training and test data

Train accuracy	*Train loss*	*Test accuracy*	*Test loss*
0.981	0.111	0.934	0.203

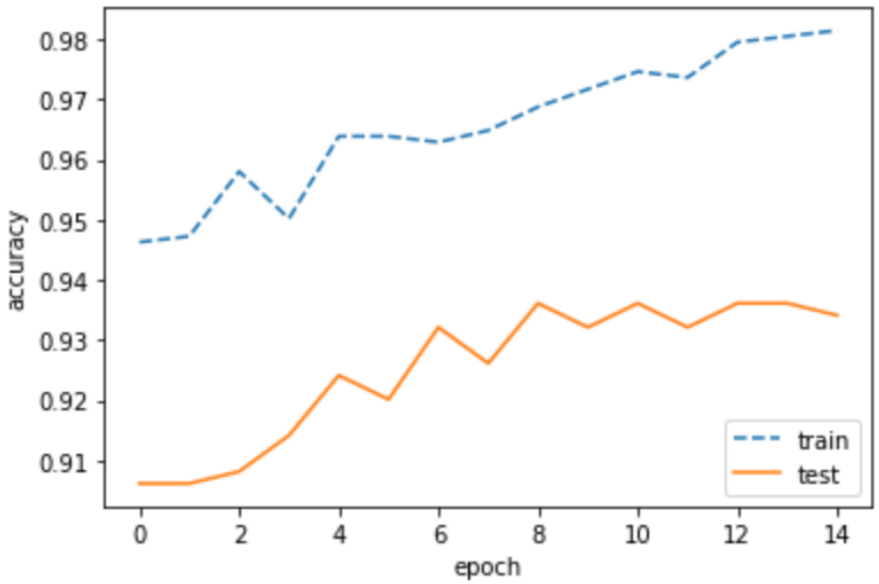

Figure 5 Learning curve for accuracy and epoch for train and test data

4.Discussion

4.1 MNIST data prediction using MLP

After training MLP model with MNIST data using 1024 train samples and 501 test samples, with dense layers or all connected layers has resulted in high train accuracy more than 98%. Whereas, the test accuracy for the model is 86%. If we observe the plot in Figure 1, the train accuracy curve keeps rising with every epoch and test accuracy is not making much progress after fourth epoch which shows that model is overfitting the training data in this case.

4.2 MNIST data prediction using CNN

After training CNN model with MNIST data with five layers two convolutional layers, one max-pooling layer and two dense layers, accuracy achieved for train and test data is really low around 11-12% for both as can be seen for Table 2. The reason behind this low accuracy data is we have trained the model only 1024 images which is very less data to get more performance. On the other hand, if we observe the plot of the model from Figure 2 the good thing is that training curve and test curve are both rising through all epochs which shows that there is no overfitting and that is also because we have used dropout layers in between the layers.

4.3 MNIST data prediction using VGG16 (0 layers retrained)

For our third experiment, we used the pre-trained VGG16 network available in Keras and did not retrain any layer of the model. We further added the dense layer of 1024 neurons before the output layer. As it can be seen from Table 3, that it has shown significant improvement from our CNN model. The train accuracy is 85% and test accuracy is 79%, with the same amount of data and same number of epochs as given in our last CNN model. Also if we look at the plot in figure 3, train accuracy and test accuracy, both the curves are rising with each epoch which shows that model is generalised and not overfitted.

4.4 MNIST data prediction using VGG16 (last 3 layers retrained)

The same model as in 4.3 was used and last three layers of VGG16 network were retrained which lead to increase in the train accuracy 94% and test accuracy 89% as can be seen in Table 4. But if we look at the plot in Figure 4, the raise in the train accuracy is more as compared to test accuracy which might indicate some level of overfitting on train data.

4.5 MNIST data prediction using VGG16 (last 9 layers retrained)

The same model as in 4.4 was used and last nine layers of VGG16 network were retrained which lead to increase in the train accuracy 98% and test accuracy 93% as can be seen in Table 5. But if we look at the plot in Figure 5, the raise in the train accuracy is more as compared to test accuracy and both the curves are moving parallelly which indicates the increase in performance.

5.Conclusion

From above experiments, we can conclude that for training basic CNN model for image processing we need a greater number of layers and huge datasets. Also, if we use pre-trained models than we can get better performance on the smaller dataset as well. For further research we should try retraining all layers and observe the performance curves. We should also try using measures to prevent overfitting like dropout layers, PCA to observe the performance of the model.

6. References

T. P. Trappenberg, *Fundamentals of machine learning*. Oxford, United Kingdom: Oxford University Press, 2020.