Transfer Learning using MNIST dataset

Charmaine Aurette, Prince Mensah, Leonard Sanya

April 8, 2024

1 Introduction

The dataset we will use is the MNIST dataset, a classic dataset in the machine learning community, which has been around for almost as long as the field itself and has been very intensively studied. It is a set of 60,000 training images, plus 10,000 test images, assembled by the National Institute of Standards and Technology (the NIST in MNIST) in the 1980s. You can think of ”solving” MNIST as the ”Hello World” of deep learning.

1.1 Problem Statement

The problem we are trying to solve here is to classify grayscale images of handwritten digits (28 pixels by 28 pixels) into their 10 categories (0 to 9) using Neural Networks.

2 Methodology

In this section, we discuss the phases of the methodology used in building an neural network. We first understand download the MNIST dataset and do the preprocessing and secondly we build a neural network to perform the classification.

2.1 Data preprocessing

For our project, we separated the MNIST dataset into two subsets according to the class labels. Pictures of even numbers are included in one dataset, whereas pictures of odd numbers are included in the other. Table 3 represents the summary of image representation per class for the the MNIST-Even and MNIST-Odd datasets.

Table 1: Image distribution per class

MNIST - Even	**Image Counts**	**MNIST - Odd**	**Image Counts**
Class 0	980	Class 1	1135
Class 2	1032	Class 3	1010
Class 4	982	Class 5	892
Class 6	958	Class 7	1028
Class 8	974	Class 9	1009

2.2 Neural network

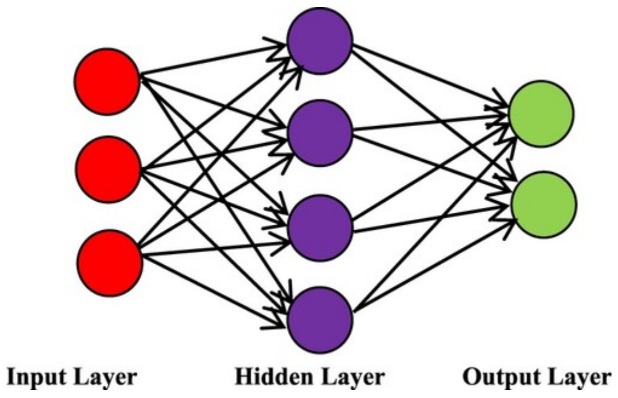


Figure 1: One Hidden Neural Network

We used a one-hidden-layer neural network composed of an input layer, a hidden layer, and an output layer. The input layer takes the 28 pixels by 28 pixels features as inputs for every image, implying that the input layer will have 784 nodes. In the hidden layer, we used 500 nodes, ReLu activation function was employed in the hidden layer. Table 2 summarizes the different parameters that we have used to tune our model.

Parameter	**Value**
Number of hidden layers	1
Activation functions	Softmax for Output layer
	ReLU for Hidden layer
Learning Rate	0.001
Number of Epochs	100
Hidden-layer Dimensions	500

Table 2: Summary of the Model Structure

2.3 Performance Metrics

Performance metrics refer to measures used for model evaluation in machine learning. Therefore, in this section , we present performance measures that can be used in a binary classification to evaluate the model's performance. The confusion matrix is a table used to describe the performance of a model in a given set of tests where the true values of the test data are known. It is a summary of the prediction of the model by giving the counts of those classified correctly and incorrectly by the model. Figure 3.4 shows a sample confusion matrix for a binary classification problem.

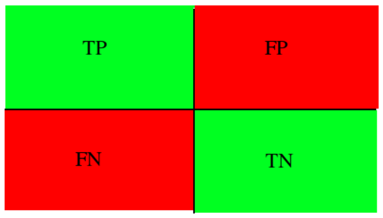


Figure 2: Confusion matrix in a binary classification

- **TP** refers to the model predicting a positive values and it was actually a positive value,
- **FP** refers to the model predicting a positive value when it is a negative value,
- **TN** refers to the model predicting a negative value, and it was actually a negative value,

- **FN** refers to the model predicting a negative value when its a positive.

Accuracy
Accuracy is the fraction of the correctly classified observation divided by the total number of observations. It tells us how good our model is in terms of making the correct predictions on the test set. The best accuracy is 1, meaning that all the data points were correctly classified by the model and the worst accuracy is 0, implying that the model classified all the points incorrectly. Accuracy in binary classification is given by

$$\frac{TP + TN}{TP + FP + TN + FN}$$

3 Results

In this section we present the results obtained on the model evaluation on the test sets to determine how good the model is in terms of generalization on the train set.

3.1 Softmax Regression

We trained a softmax regression model MNIST-Even and MNIST-Odd dataset and after 100 epochs of training, we noted an accuracy of 96.02% and 95.13% for the test sets from MNIST-Even and MNIST-Odd dataset respectively. Considering the losses, Figure 3 and Figure 5 shows that the model converges well implying a good generalization the training sets from both datasets. This is visualized in the confusion matrices Figure 4 and Figure 6, softmax regression model classifies images into their respective classes with a high level of accuracy.

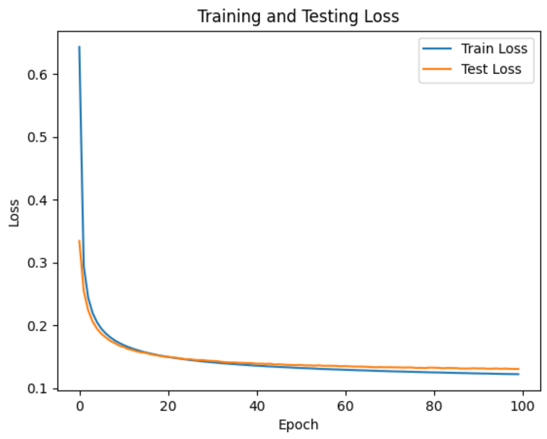


Figure 3: Loss on MNIST-Even

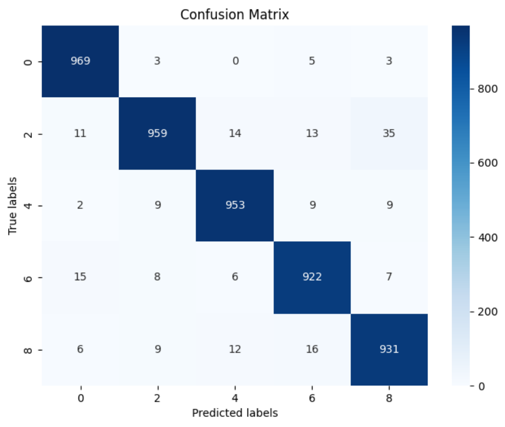


Figure 4: Evaluation on MNIST-Even testset

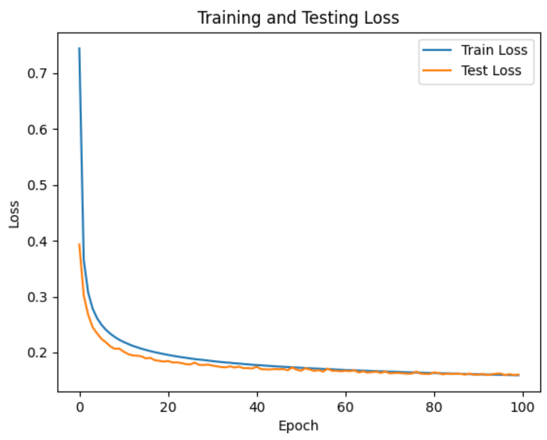
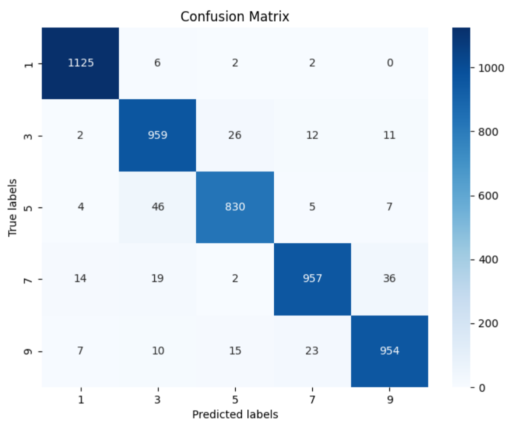


Figure 5: Loss on MNIST-Odd



	1	3	5	7	9
1	1125	6	2	2	0
3	2	959	26	12	11
5	4	46	830	5	7
7	14	19	2	957	36
9	7	10	15	23	954

Figure 6: Evaluation on MNIST-Odd testset

3.2 MNIST-Even Dataset

We trained our model for 100 epochs on the MNIST-Even dataset, and then tracked its performance to see how it learned and got better over time. Analyzing the training loss, a measurement of how well the model's predictions match the actual labels in the training data, is one technique for evaluating the model's performance during training. We note that the model converges as shown by the graph of training loss in Figure 7. The model attains 98.52% accuracy on the even test set. From Figure 8, we note that the model classifies the images in the even test set correctly implying that the model captured the underlying features and generalized on the training set.

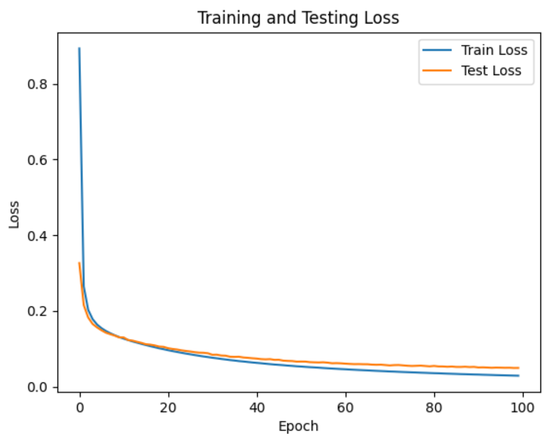
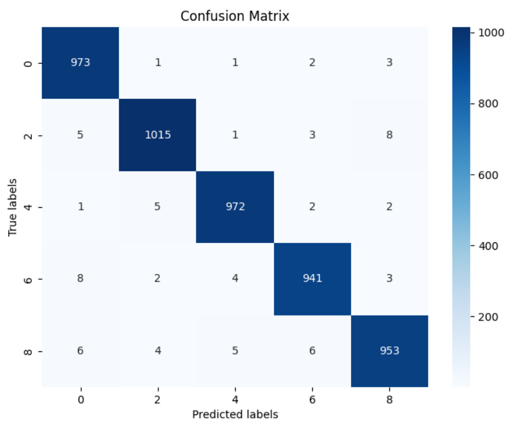


Figure 7: Loss on MNIST-Even



	0	2	4	6	8
0	973	1	1	2	3
2	5	1015	1	3	8
4	1	5	972	2	2
6	8	2	4	941	3
8	6	4	5	6	953

Figure 8: Evaluation on MNIST-Even testset

3.3 MNIST-Odd Dataset

Similarly, We trained our model for 100 epochs on the MNIST-Odd dataset, and assessed its performance to see how it learned and got better over time. We also note that the model converges, as shown by the graph of training loss in Figure 9. The model reaches 98. 34% precision in the test set, and using the confusion matrix in Figure 10, we note that the model can correctly classify odd images in the odd test set.

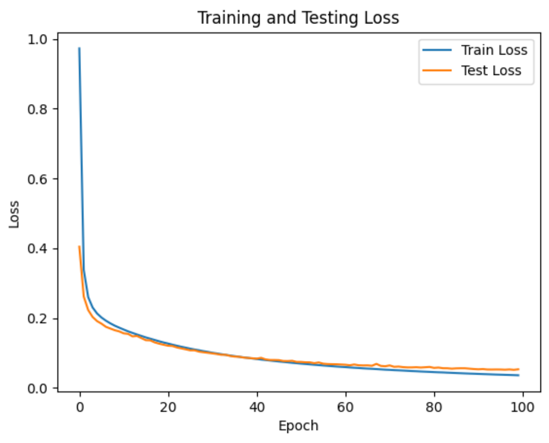
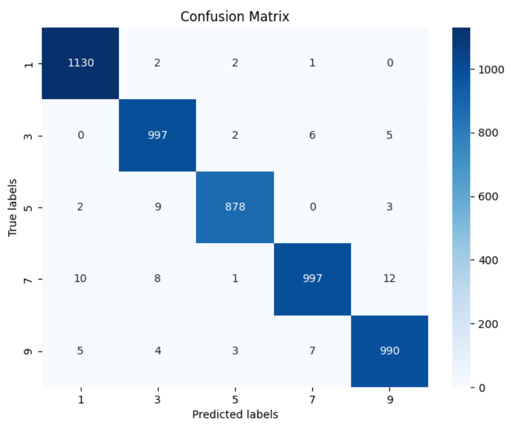


Figure 9: Loss on MNIST-Odd



	1	3	5	7	9
1	1130	2	2	1	0
3	0	997	2	6	5
5	2	9	878	0	3
7	10	8	1	997	12
9	5	4	3	7	990

Figure 10: Evaluation on MNIST-Odd testset

3.4 Transfer Learning

Transfer learning is a machine learning and deep learning technique that involves adapting or reusing a model that has been trained on one task for a related activity. Transfer learning uses the knowledge obtained from addressing one problem to another that is related, rather than starting from scratch when training a model. In transfer learning, the learnt representations of the pre-trained model, also known as the base model, are adjusted on the target dataset. Using the target dataset, part or all of the pre-trained model's layers are retrained during this fine-tuning step, with the weights of the previous layers either left unchanged or slightly adjusted.

3.4.1 Model trained on MNIST-Even dataset

Our first base model was a neural network that was pretrained using the MNIST-Even dataset. After 30 training epochs, as shown in Figure 11, it is clear that the model converges and the test set loss starts to decrease dramatically. Moreover, Figure 12 shows the model's high generalisation ability on the training set, as it correctly assigns classes to images in the test set.

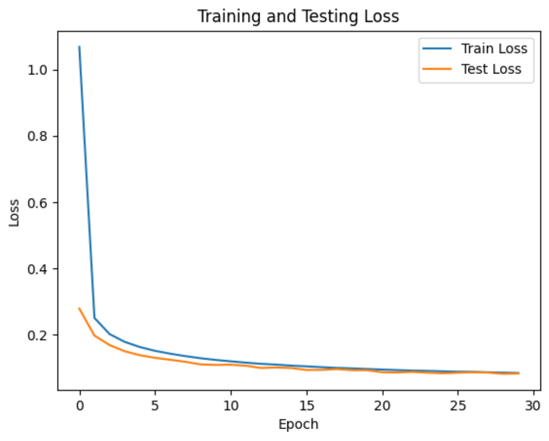
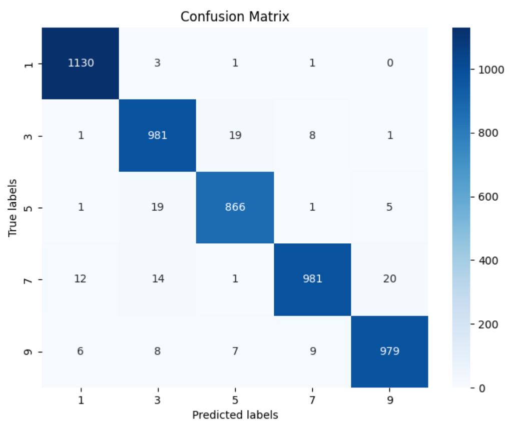


Figure 11: Loss on MNIST-Odd



	1	3	5	7	9
1	1130	3	1	1	0
3	1	981	19	8	1
5	1	19	866	1	5
7	12	14	1	981	20
9	6	8	7	9	979

Figure 12: Evaluation on MNIST-Odd testset

3.4.2 Model trained on MNIST-Odd dataset

We used the neural network trained on the MNIST-Odd dataset as the base model. After 30 epochs of training, in Figure 13, it can be seen that the model converges because the loss on the test set is very small.

Also, the generalization of the model on the training set is great since it can be observed from Figure 14, the model classification of the images on the test set to their respective classes is good.

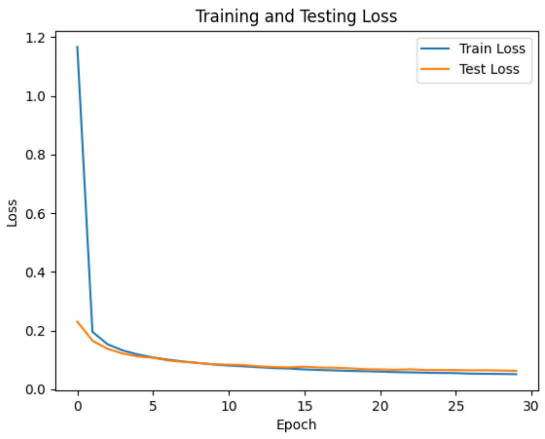


Figure 13: Loss on MNIST-Even

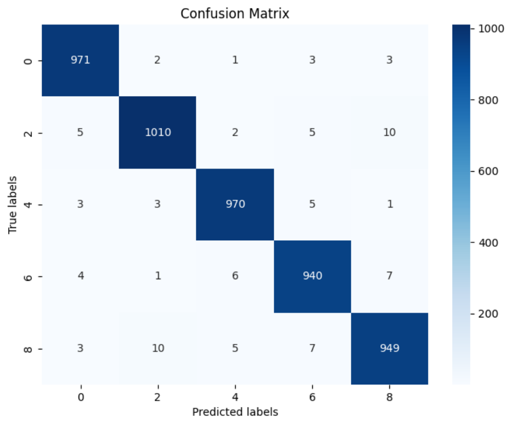


Figure 14: Evaluation on MNIST-Even testset

3.5 Summary

Table 4 summarizes the results obtained using the different approaches explained above.

Table 3: Summary table for model performances

Datasets	Softmax Regression	Neural Network	Transfer Learning
MNIST - Even	96.02%	98.56%	98.21%
MNIST - Odd	95.13%	98.34%	98.06%

Table 4 gives a summary of the results obtained when using different proportions of the training data on the softmax regression and and when applying the transfer learning technique on the already pretrained neural network.

Table 4: Summary table for model performances on different sizes of training data

Datasets	Softmax Regression			Neural Network		
	20%	50%	80%	20%	50%	80%
MNIST - Even	95.18%	95.86%	96.08%	95.25%	96.47%	97.2%
MNIST - Odd	94.05%	74.72%	95.17%	95.64%	96.33%	97.05%

4 Conclusion

The main goal of this project was to build a softmax regression and neural network models for image classification using the MNIST dataset putting into consideration the transfer learning technique. We fitted both softmax regression and one-hidden-layer neural networks on both MNIST-Even and MNIST-Odd datasets and performing the same on transfer learning. The results shows a good performance on image classification using the models trained on MNIST-Even and MNIST-Odd datasets and also when considering the transfer learning techniques. However, it can be noted that using neural networks out performs the use of softmax regression even though the difference is not too high.

References

[1] https://www.kaggle.com/code/riteshsinha/neural-networks-with-pytorch-mnist

[2] https://adeveloperdiary.com/data-science/deep-learning/neural-network-with-softmax-in-python/

[3] https://github.com/pmensah28/TL-From-Scratch