Evaluation of the proposed CNN model to classify the MNIST handwritten dataset

Md. Nafis Rabbi
Computer Science and Engineering
BRAC University
Dhaka Bangladesh
md.nafis.rabbi@g.bracu.ac.bd

Md. Rashik Saif
Computer Science and Engineering
BRAC University
Dhaka Bangladesh
md.rashik.saif@g.bracu.ac.bd

Ifaz Ahmed
Computer Science and Engineering
BRAC University
Dhaka Bangladesh
ifaz.ahmed@g.bracu.ac.bd

Annajiat Alim Rasel
Senior Lecturer
Computer Science and Engineering
BRAC University
Dhaka Bangladesh
annajiat@bracu.ac.bd

Abstract

For a long time, the issue of transcribed digit acknowledgement has been a source of contention in the instance order community. Numerous studies have demonstrated that neural networks are extremely effective at organizing information. The major purpose of this study is to examine many existing model configurations in order to develop accurate and efficient ways for detecting transcribed numerical data. The Convolutional Neural Network exhibition is highlighted in this publication (CCN). Because handwritten information differs from person to person, automated recognition of handwritten numbers from visual data can be challenging. The main purpose of this study is to propose a basic convolutional neural network (CNN) model to classify the MNIST handwritten dataset, with a test accuracy of more than 98%, and to evaluate several optimizers.

I. Introduction

The subject of identifying manually written numbers has recently received a lot of attention, leading to the invention of several pre-processing approaches and arrangement computations. However, interpreting printed digits remains a struggle for us. Because various customers write by hand and each client's writing style changes significantly, the biggest challenge with identifying transcribed numbers is the real variation in size, interpretation, stroke thickness, pivot, and twisting of the numerical picture. To recognize handwritten numbers (0–9) from the well-known MNIST dataset, the TensorFlow framework (library), Python, and associated modules are utilized. When a user enters a digit, the computer scans it and displays the findings as well as an accuracy rate.

A CNN model is fed a labeled collection of specific picture classes to assess image data. MNIST is a database of labeled handwritten character pictures. This database and CNN compilation serve as the foundation for our study. As a consequence, NumPy, Pandas, TensorFlow, and Keras libraries are required to run the model. These are the principal support structures for the main project. In a single dataset, various CNN combinations might generate diverse results. We examined the Adam, SGD, and RMSprop optimizers using two distinct CNN models.

II. Literature Review

Significant Innovation work in machine humanization have benefitted deep learning, machine learning, and artificial intelligence. The sophistication of machines has grown throughout time, from simple math operations to retinal recognition, increasing the safety and control of human lives. Handwritten text recognition, on the other hand, is an essential machine learning and deep learning application for identifying forgeries. According to S. M. Shamim [1], when combined with the SVM classifier, the random forest classifier, the naive bayes classifier (j48), and the random tree classifier, the MLP classifier generated the most accurate results with the fewest mistakes. In a study-comparison, Norhidayu binti[3] employed SVM, KNN, and MLP models to categorize handwritten text. Hongkai[4] Wang's research has mostly focused on contrasting deep learning approaches with machine learning technology.

Handwritten character recognition refers to a computer's ability to recognize and categorize human handwritten numbers from various sources such as images, papers, touch displays, and so on (0-9). With this content, the field of deep literacy has been the focus of endless inquiry. Numerous tasks related to number recognition include processing bank checks, sorting postal mail, and number plate identification. We encounter many difficulties in handwritten number identification.

Character recognition has made extensive use of artificial neural networks due to their robust self-learning, adaptive, classification, fault-tolerance, and quick recognition capabilities. Convolutional neural network (CNN) usage in image processing has increased recently with the development of deep learning. Some attributes of CNN include local connections, weight sharing, and pooling. The CNN model's retrieved features have strong descriptive power. Convolution neural networks are examined and validated using the MNIST data set in this research. To increase the network's recognition rate, this research develops a three-model CNN network and evaluates the Adam, SGD, and RMSprop optimizers. The RMSprop optimizer will perform better in real-world data, according to the experiment on the MNIST data set.

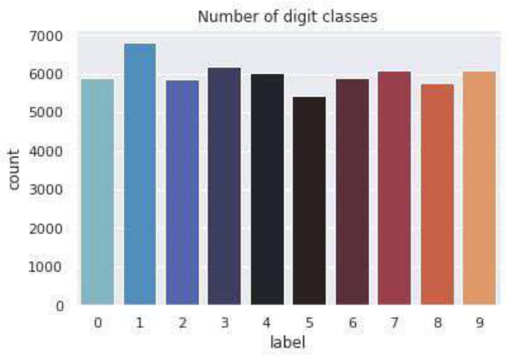


Figure 1. MNIST handwritten digit dataset bar graph

Figure 2. MNIST handwritten digits plotting

III. Methodology

The optimizers (Adam, SGD, and RMSprop) are compared using a typical chart for each method that considers parameters such as dataset, algorithm complexity, accuracy, and runtime under ideal conditions.

A. Dataset

About 70,000 images of handwritten numbers from 0 to 9 can be found in MNIST data. There are two sections to this dataset: the training dataset and the test dataset. The test image contains 10,000 images, and the training image 60,000. Each image is 28*28 pixels with a pixel value between 0 and 255.

B. Adam Optimizer

Adam is a stochastic gradient descent variant that has lately gained traction in computer vision and natural language processing. The technique used to optimize a deep learning model can have a major impact on how well the model performs over minutes, hours, and days. Adam recognizes the benefits of RMSProp and AdaGrad. As learning occurs, a learning rate is recorded and independently adjusted for each network weight (parameter).

C. RMSprop

The AdaGrad gradient descent algorithm is a version of RMSProp. Gradient descent follows the negative gradient of an objective function to find its minimum. For all input variables, AdaGrad employs the same step size (learning rate). To modify the step size for each parameter, RMSProp employs a falling average of partial gradients.

D. SGD

SGD, or stochastic gradient descent, is an iterative method for maximizing an objective function. It may be thought of as a stochastic version of gradient descent optimization. Because it recalculates gradients for related samples before each parameter change, batch gradient descent performs redundant operations on big datasets.

E. Visualization

Using the MNIST dataset, we investigate various levels of deep learning optimization in terms of execution time, complexity, and accuracy rate. We were able to present the data acquired from the deep analysis of the algorithms using graphs and tabular format charts with the aid of the matplotlib module, which delivers the most accurate visualizations of the algorithms' step-by-step achievements in detecting the number.

IV. Implementation

We employed two models and three optimizers to compare the algorithms based on their working efficiency, processing time, and complication:

1. Adaptive Moment Estimation (Adam)
2. Root Mean Squared Propagation (RMSprop)
3. Stochastic Gradient Descent (SGD)

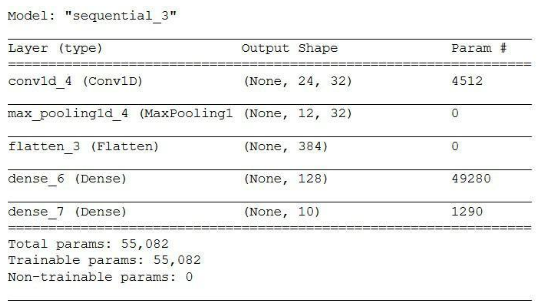
An input layer, a one-dimensional convolutional layer, a max pooling layer, a flattening layer, a dense layer, and an output layer comprise Model 1. Model 2 includes an extra layer set that includes a max pooling layer and a one-dimensional convolutional layer.

A. Data Preprocessing

In machine learning and deep learning, pre-processing is the first stage. This is meant to improve the input data by removing undesired pollutants and duplication. The Keras API is used in the code to import the MNIST dataset and show the first sixty photographs in the training dataset. In addition, the images are square, measuring 28 by 28 pixels.

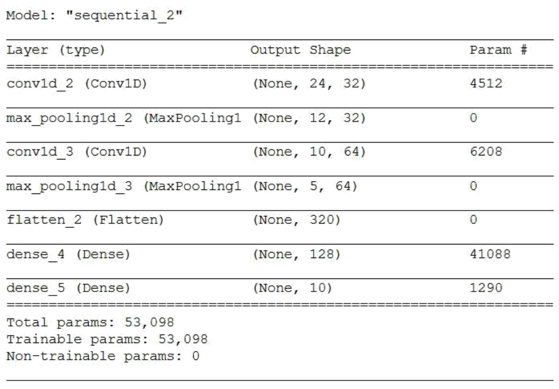
B. Model Preparation

The model is separated into two parts: the front end, which comprises convolutional and pooling layers, and the backend, which predicts. Given that the job is a multi-class classification assignment, we know that an output layer of ten nodes will be needed to predict the probability distribution of each of the ten classes. This will necessitate the usage of a SoftMax activation mechanism as well. The baseline model will be trained for 10 training epochs with a batch size of 64 cases and a validation split of 0.3. We might begin the convolutional front-end with a single convolutional layer with a small filter size (3,3) and a few filters (32) followed by a max pooling layer.



```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_4 (Conv1D)            (None, 24, 32)            4512
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 12, 32)            0
_________________________________________________________________
flatten_3 (Flatten)          (None, 384)               0
_________________________________________________________________
dense_6 (Dense)              (None, 128)               49280
_________________________________________________________________
dense_7 (Dense)              (None, 10)                1290
=================================================================
Total params: 55,082
Trainable params: 55,082
Non-trainable params: 0
_________________________________________________________________
```

Figure 3. Model 1



```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_2 (Conv1D)            (None, 24, 32)            4512
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 12, 32)            0
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 10, 64)            6208
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 5, 64)             0
_________________________________________________________________
flatten_2 (Flatten)          (None, 320)               0
_________________________________________________________________
dense_4 (Dense)              (None, 128)               41088
_________________________________________________________________
dense_5 (Dense)              (None, 10)                1290
=================================================================
Total params: 53,098
Trainable params: 53,098
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. Model 2

V. Results

In the case of model 1, the optimizer SDG had train accuracy of 92.08 percent, validation accuracy of 92.27 percent and test accuracy was 93.66 percent. With Adam as optimizer the train accuracy shoots up to 98.66 percent, validation accuracy to 97.96 percent and overall test accuracy to 98.33 percent. Train accuracy reaches highest ever with RMSprop optimizer to 99.53 percent.

The upwards trend continues both for the validation accuracy and test accuracy which reaches 98.45 and 98.72 percent respectively. Result of the CNN ‘Model 1’ for different optimizer are given below

Table 1: Results of CNN Model 1

	Optimizer		
Accuracy	**Adam**	**SGD**	**RMSprop**
Training	98.66%	92.08%	99.53%
Validation	97.96%	92.27%	98.45%
Test	98.33%	93.66%	98.72%

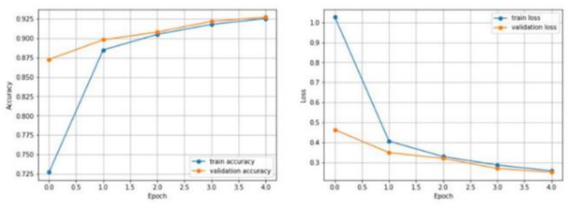


Figure 5. Model 1 loss for SGD

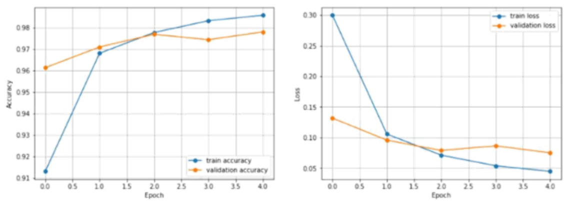


Figure 6. Model 1 loss for Adam

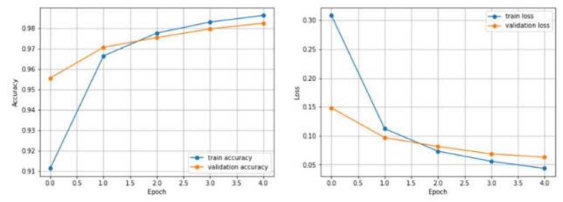


Figure 7. Model 1 loss for RMSprop

When it comes to model 2 with the same set of optimizers SDG shows better train accuracy with comparison to model 1 which is 93.79 percent. Both the validation accuracy and test accuracy also increase to 94.05 and 94.49 percent. In terms of Adam as optimizer train accuracy shoots to 99.39 percent, validation accuracy to 98.10 percent which isn't much of a jump compared to model 1. Same for test accuracy which jumps from 98.33 to 98.36 percent. Result of the CNN 'Model 2' for different optimizer are given below

Table 2: Results of CNN Model 2

	Optimizer		
Accuracy	**Adam**	**SGD**	**RMSprop**
Training	99.39%	93.79%	99.39%
Validation	98.10%	94.05%	98.50%
Test	98.36%	94.49%	98.79%

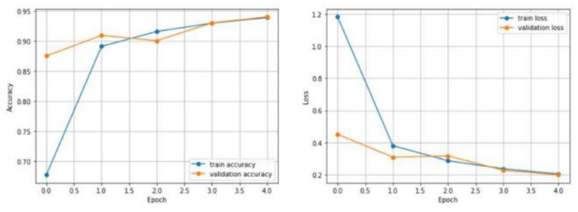


Figure 8. Model 2 loss for SGD

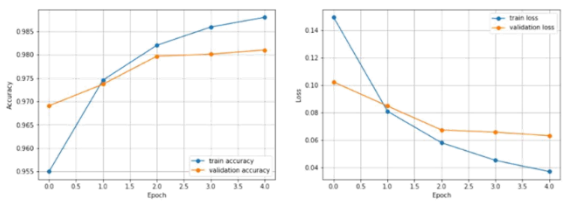


Figure 9. Model 2 loss for Adam

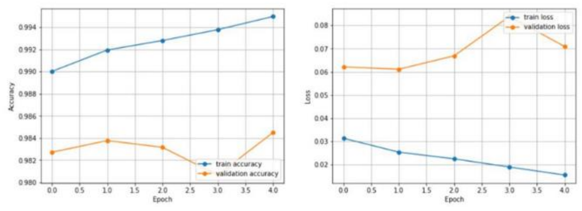


Figure 10. Model 2 loss for RMSprop

After using RMSprop as optimizer test accuracy reaches 99.39 percent, validation accuracy 98.50 percent and test accuracy 98.79 percent.

VI. Conclusion

In this particular evaluation two different CNN models were used and 3 different optimizers which are Adam, SGD and RMSprop were used. With two different CNN models and 3 different optimizers we get a wide variety of test results. But after evaluating all of the test results a decision has been reached. The best test accuracy which is 98.72% was reached on RMSprop optimizer and it was seen on model 1. By evaluating the results, we get from model 1 we can also see that the lowest test accuracy is reached on SGD optimizer which is 93.66%. In the case of model 2, best test accuracy is reached on RMSprop which reaches 98.79%. At the same time on SDG optimizer still shows the result with least accuracy which reaches 94.49 percent. After plotting the data of train and validation accuracy in a graph, we can see that both of their rate is different in model 2. Which indicates that even if the test result accuracy is higher on paper, in real life scenarios this model may not be as consistent as we would like it to be. On the other hand, if we look at the graph, we get following the RMSprop optimizer on model 1 we can evaluate that the train and validation accuracy rate of the graph analysis is to some extent similar. Which indicates that with real life data and in real life scenarios, this specific model will perform better and more constantly. At this point it can be concluded that although test accuracy for model 2 on RMSprop optimizer has reached the highest percentage, as most consistent with real life data and in real life scenarios model 1 is the better performing model in this evaluation.

VII. References

1. "Handwritten Digit Recognition using Machine Learning Algorithms", S M Shamim, Mohammad Badrul Alam Miah, Angona Sarker, Masud Rana & Abdullah Al Jobair.
2. "Handwritten Digit Recognition Using Deep Learning", Anuj Dutt and Aashi Dutt.
3. "Handwritten recognition using SVM, KNN, and Neural networks", Norhidayu Binti Abdul Hamid, Nilam Nur Binti Amir Sharif.
4. "Comparison of machine learning methods for classifying mediastinal lymph node metastasis of non-small cell lung cancer from 18 FFDG PET/CT images" by Hongkai Wang, Zongwei Zhou, Yingci Li, Zhonghua Chen, Peiou Lu, Wenzhi Wang, Wanyu Liu, and Lijuan Yu.
5. "LeCun et al., "Backpropagation applied to handwritten zip code recognition," Neural computation, vol. 1, no. 4, pp. 541-551, 1989.
6. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, 2012, pp. 1097-1105.
7. R. Hecht-Nielsen, "Theory of the backpropagation neural network," in Neural networks for perception: Elsevier, 1992, pp. 65-93.
8. B. Siddique, M. M. R. Khan, R. B. Arif, and Z. Ashrafi, "Study and Observation of the Variations of Accuracies for Handwritten Digits Recognition with Various Hidden Layers and Epochs using Neural Network Algorithm," in 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), 2018, pp. 118-123: IEEE.
9. R. B. Arif, M. A. B. Siddique, M. M. R. Khan, and M. R. Oishe, "Study and Observation of the Variations of Accuracies for Handwritten Digits Recognition with Various Hidden Layers and Epochs using Convolutional Neural Network," in 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), 2018, pp. 112-117: IEEE.
10. E. Kussul and T. Baidyk, "Improved method of handwritten digit recognition tested on MNIST database," Image and Vision Computing, vol. 22, no. 12, pp. 971-981, 2004.
11. B. Zhang and S. N. Srihari, "Fast k-nearest neighbor classification using cluster-based trees," IEEE Transactions on Pattern analysis and machine intelligence, vol. 26, no. 4, pp. 525-528, 2004.
12. Y. Liu and Q. Liu, "Convolutional neural networks with largemargin softmax loss function for cognitive load recognition," in 2017 36th Chinese Control Conference (CCC), 2017, pp. 4045- 4049: IEEE
13. Tavanaei and A. S. Maida, "Multi-layer unsupervised learning in a spiking convolutional neural network," in 2017 International Joint Conference on Neural Networks (IJCNN), 2017, pp. 2023-2030: IEEE