

Programming Test: Applying CNN and Autoencoders on MNIST Dataset

RUDRANEEL DUTTA

Work Link: <https://github.com/rudraneel18/MNIST-Dataset-Model>

BTECH, COMPUTER SCIENCE & BUSINESS SYSTEMS

Institute of Engineering and Management, Kolkata, India

Abstract:- This paper presents an implementation using Back-propagation Neural Network to achieve the classification of the MNIST handwritten digit database. Here, we utilise the accuracy of classification and the plot of training loss to assess the neural network's performance. According to the experimental findings, back-propagation neural networks can be used to real-world categorization problems. Additionally, we attempt to perform picture compression using Autoencoder and analyse its results. The performance is improved by increasing the speed of the neural network and decreasing the network size. The accuracy rate cannot be guaranteed, though, and various causes are covered. Then, we attempt to change the original neural network's structure into a convolutional neural network (CNN). The outcomes show that CNN can be used to enhance performance while tackling an image recognition challenge. Additionally, we create a Conv Autoencoder structure by combining CNN and Autoencoder, and we run certain tests. Finally, the output of our network system is compared to output from a different study that used the same MNIST handwritten digit database.

Keywords: MNIST handwritten digit database; Neural Network; Back-propagation; Classification; Autoencoder; Convolutional Neural Network

I. BACKGROUND

An important issue in optical character recognition is handwritten digit recognition, which can be utilized as a test case for theories of pattern recognition and machine learning techniques. A number of common databases have arisen to support the study of machine learning and pattern recognition. To evaluate the recognition outcomes of different algorithms and to lighten the burden, the handwritten digits are preprocessed, including segmentation and normalization.

This paper aims to define the application of neural networks to the MNIST handwritten digit classification problem. To address the categorization issue, we must first create a neural network model and then put it into practice. In addition, additional research has been conducted to examine a variety of approaches that might ultimately affect how well our model performs.

II. MATHEMATICAL FRAMEWORK

2.1 Method:-

As mentioned above, in this paper we use MNIST handwritten digit database in which the handwritten digits have been preprocessed including segmentation and normalization. There are 60,000 training images and 10,000 test images, and the dimensionality of each image sample vector is $28 * 28 = 784$, where each element is binary.

2.2 Model Summary

Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 28, 28, 1)	0
random_rotation (RandomRotation)	(None, 28, 28, 1)	0
conv2d (Conv2D)	(None, 24, 24, 32)	832
activation (Activation)	(None, 24, 24, 32)	0
batch_normalization (BatchNormalization)	(None, 24, 24, 32)	128
max_pooling2d (MaxPooling2D)	(None, 12, 12, 32)	0
dropout (Dropout)	(None, 12, 12, 32)	0
conv2d_1 (Conv2D)	(None, 10, 10, 32)	9248
activation_1 (Activation)	(None, 10, 10, 32)	0
batch_normalization_1 (BatchNormalization)	(None, 10, 10, 32)	128
max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 32)	0
dropout_1 (Dropout)	(None, 5, 5, 32)	0
flatten (Flatten)	(None, 800)	0
dense (Dense)	(None, 512)	410112
activation_2 (Activation)	(None, 512)	0
dropout_2 (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 512)	262656
activation_3 (Activation)	(None, 512)	0
dropout_3 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 10)	5130
activation_4 (Activation)	(None, 10)	0
Total params: 688,234		
Trainable params: 688,106		
Non-trainable params: 128		

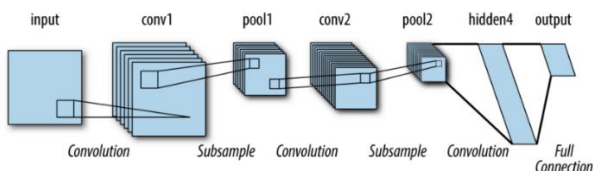
2.3 Preprocessing and Augmentation

In order to prepare images for use in model training and inference, a method known as image preprocessing is used. This includes adjustments to the size, orientation, and color. Since a model should have a specific type or sequence of inputs, we reshaped the input picture in this instance to be 28*28(width*height) for our model input. After which we divided the RGB values of our input pixels by 255 to convert them to black and white.

Image augmentation refers to image alterations that are used to create alternative versions of comparable information in order to expose the model to a broader range of training instances. However, there is a crucial distinction between image augmentation and image preprocessing: although image preprocessing techniques are applied to both training and test sets, image augmentation is only applied to the training data. As a result, in some circumstances, a transformation that would be an augmentation may be better as a pretreatment step. In this model we chose to rotate the image by 10 degrees

2.4 Convolutional Neural Network (CNN):-

Convolutional Neural Networks are a special kind of multi-layer neural networks. They are also trained with the backpropagation algorithm but with the different architecture from other neural networks. Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal pre-processing and they can recognize patterns with extreme variability such as handwritten characters even digits



2.5 Back-propagation neural network setting up:-

As discussed in the section above, Back-propagation neural networks are supervised multi-layer feed forward neural networks, commonly consist of an input layer, an output layer, and one or several hidden layers. Here we build a neural network with only one hidden layer. Input layer contains $28 * 28 = 784$ neurons, representing the features; Hidden layer contains 300 neurons, using Sigmoid as activation function; And output layer contains 10 neurons representing the digits from 0 to 9.

Besides, we use cross-entropy error function as network loss function. It seems that cross-entropy error function is more suitable for classification problems than mean squared error (MSE) function. And it is convenient to use CrossEntropyLoss () to achieve it. As for the optimizer, here we tried Adam and RMSProp choose to use Adam optimizer implementing the Adam algorithm which can be achieved by using Adam ()

After defining the neural network, we train the model by batch. In order to set the appropriate parameters such as batch_size, number_of_epochs and learning rate, we conducted several tests to evaluate. There should be a balance among these parameters, contributing to the performance of the neural network together. Finally, the system parameters are given as number_of_epochs = 60, learning rate = 0.001, decay = 1e-4 and batch_size = 32.

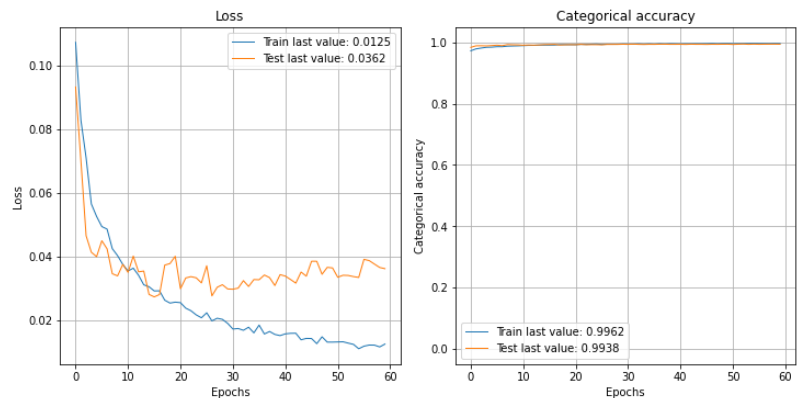
2.6 Evaluation:-

We decide to assess the training accuracy and testing accuracy in order to evaluate the effectiveness of the neural network, make predictions, and present the outcomes generated by our neural network. Another visual depiction used during network learning is plotting historical loss from "all losses."

III. EXPECTED RESULTS AND DISCUSSION

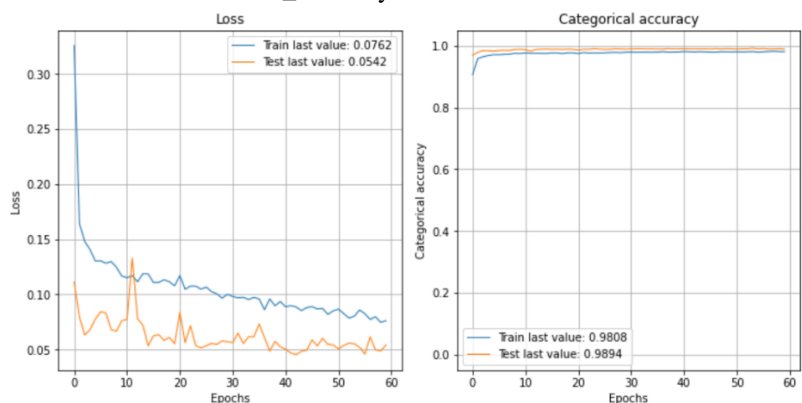
The result with Adam as optimizer has:

Loss = 0.0362 and model_accuracy=99.38%

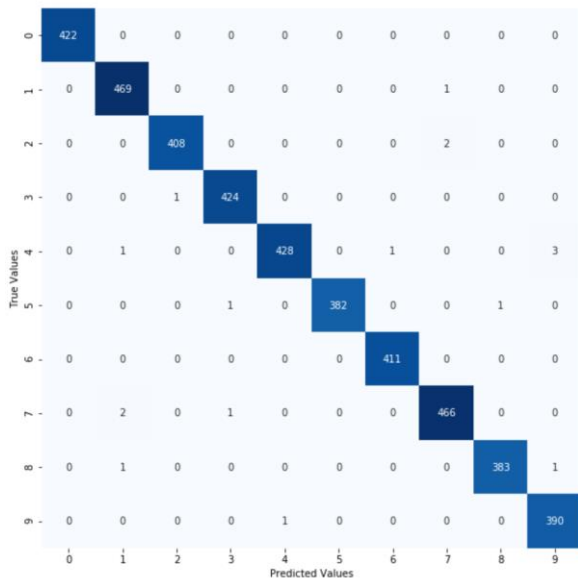


The result with Adam as optimizer has:

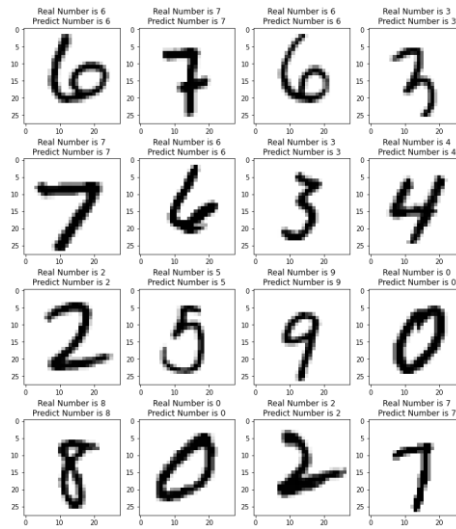
Loss=0.0542 and model_accuracy=98.94%



Confusion Matrix:



Output:-



REFERENCES:

- Dataset: <https://www.kaggle.com/competitions/digit-recognizer>
- https://www.researchgate.net/publication/354999294_Knowledge_Extraction_in_Digit_Recognition_Using_MNIST_Dataset_Evolution_in_Handwriting_Analysis
- <https://ieeexplore.ieee.org/document/6296535>
- <https://arxiv.org/pdf/1811.08278.pdf>