Handwritten Digit Recognition By Use Of Convolutional Neural Network In Python With Keras Tensorflow

Nazmul Karim & Navsangeet Kaur

Abstract

With the advent of Artificial Intelligence (AI), there has been much work with regards to application of Artificial neural Network (ANN) in the field of machine learning [2]. Deep learning has been used in diverse fields ranging from Robotics, surveillance, sports, health and drones. Convolution Neural Network (CNN) has taken centre stage in this as it has been used in sentence classification, pattern recognition, document analysis, face recognition and text categorization. This paper is concerned with the designing of an algorithm meant to process and recognize images of handwritten digits by use of CNN in python using Keras Tensorflow [6].

Keywords- Handwritten Digit Recognition, Deep Learning, Convolutional Neural Network (CNN), Epochs, Hidden Layers and Dropout.

1 Introduction

Application of deep learning models is increasing day by day. Convolutional Neural Network (CNN) is applied in deep learning majorly [2] for the purpose of visual imagery analysis, face detection and recognition, video analysis, object detection among other uses. The accuracy and efficiency of these models have with time reached human perfection level. The architecture of CNN is inspired by the mammalian visual biological system models. The cat and dog identification model has been the baseline for image recognition models for years. The neocognitron [4], a pattern recognizing model which was inspired by the works of D.H. Hubel et al. [5,6] was indeed the first attempt at computer vision [1]. This model was introduced in 1980 by Fukushima. It was an attempt at digit character recognition directly from pixel values from the images [8]. In digit recognition,handwritten image integer characters from 0-9 are provided an inputs, and the model is expected to accurately classify the image and give the output from the system. This model is created using the CNN which is made up a series of layers, with each layer having several neurons. The weighted sum of neurons in a given layer becomes the input of a neuron in the next layer in addition to a bias value which is used to correct the error from the previous layer. This paper is concerned with the designing of an algorithm meant to process and recognize images of handwritten digits by use of CNN in python using Keras Tensorflow [3]. The end result is to focus on the perfomance measure of the algorithm as well as the error rate; confusion matrix; per class accuracy and overall accuracy of the model[5].

[*] Use footnote for providing further information about author (webpage, alternative address)—*not* for acknowledging funding agencies.

2 Methodology

Convolutional neural network in Python programming language with Keras tensorflow was used in developing the model used in this paper.

2.1 Exploratory Data Analysis

The handwritten digit image dataset was loaded and the frequency plots for the data was done to get a view of how the 10 digits were distributed in the dataset. Null values were also be checked so as to clean and prepare the data for modeling

2.2 Modeling

To recognize human handwritten digits, a seven-layered convolutional neural network having a single input layer, five hidden layer and a single output layer designed as illustrated below in figure 1.

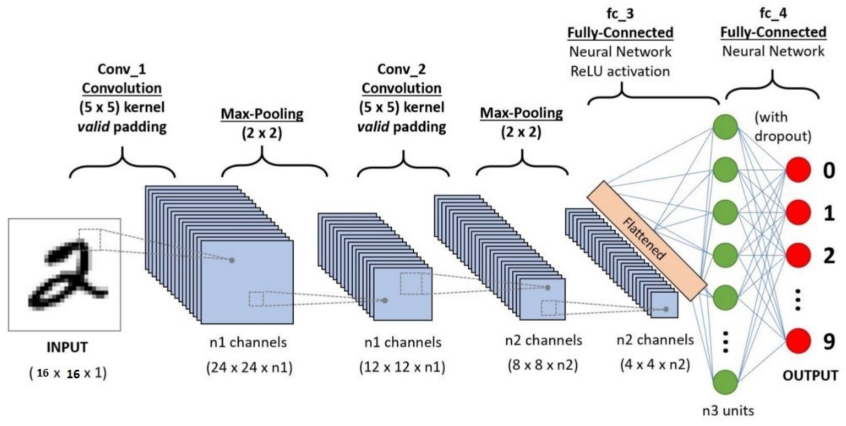

Figure 1: A 7-layered convolutional neural network for handwritten digit recognition.

The first layer consists of 16 by 16 pixel images (256 pixels) which means the network uses 256 neurons as input data. The images are in grayscale hence has only one color channel with value 0 for a white pixel value and 1 for a black pixel value. the model has got 5 hidden layers. The first hidden layer is the convolution layer 1 responsible for extracting the features from input images. It performs convolution operation on small localized areas by convolving a filter with the previous layer. It also consists of multiple feature maps having learnable kernels and Rectified Linear unit (ReLU), which determines the location of the filters. It has a kernel size of 5 x 5. Leaky ReLU is used as the activation function at the end of each convolution layer in order to improve model performance. We also use the pooling layer with a kernel size of 2x2 to reduce the the output information from convolution layer as well as reduce the number of parameters and computational complexity of our model. A flatten layer is used after the Max-pooling layer which converts 2-D maps to 1-D feature vector and makes it possible for the outputs to be handled by the fully connected layers. In order to reduce over-fitting, drop-out regularization is used at the fully connected layer. This makes it possible for the model to generalize better and over-fit less. The output layer consists of 10 neurons that determine the numbers from 0-9. The output layer uses an activation function called softmax [9], used to enhance model performance.

2.3 Performance Evaluation

The model was evaluated using several parameters to ensure that its accuracy and performance was top-notch. Among the evaluation matrix used in this paper include class-based accuracy model accuracy, overall accuracy, confusion matrix, training/validation accuracy and loss and the error rate.

3 Experimentation

3.1 Description of the dataset

The dataset contains normalized handwritten digits, which were automatically scanned from envelopes by the postal service of the U.S.A. Even though the original scanned digits were binary and of different orientations and sizes; the images which were used in this paper were deslanted and size normalized resulting in 16x16 single channel images (Le Cun et al., 1990) [8].

The data were in two g-zipped files, with each line consisting of the digit id (0-9) followed by the 256 grayscale values. The two files were the training and validation datasets respectively

The test set was notoriously "difficult", and required more tweaking to work as it has alot of noise. These data were kindly made available by the neural network group at AT&T research labs (thanks to Yann Le Cunn).

3.1.1 Dataset Visual Overview

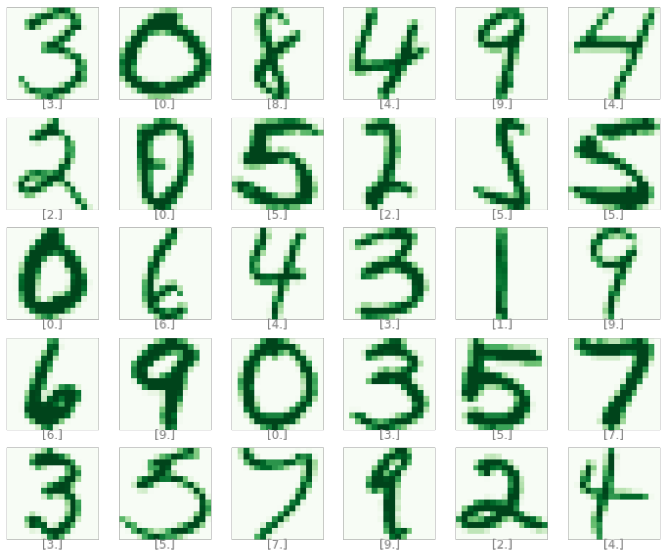

Figure 2: Sample images of the scanned handwritten digit dataset.

3.1.2 Distribution of dataset

	Distribution Of The Digits In Proportion									
Data Sets	0	1	2	3	4	5	6	7	8	9
Train Set	0.16	0.14	0.1	0.09	0.09	0.08	0.09	0.09	0.07	0.09
Validation Set	0.18	0.13	0.1	0.08	0.10	0.08	0.08	0.07	0.08	0.09

Figure 3: The proportional distribution of the digits in both datasets.

The figure 2 above shows the distribution by proportion of the two dataset.There are 7291 training observations and 2007 validation observations [7].

3.1.3 Freqency Distribution Plots Training Set

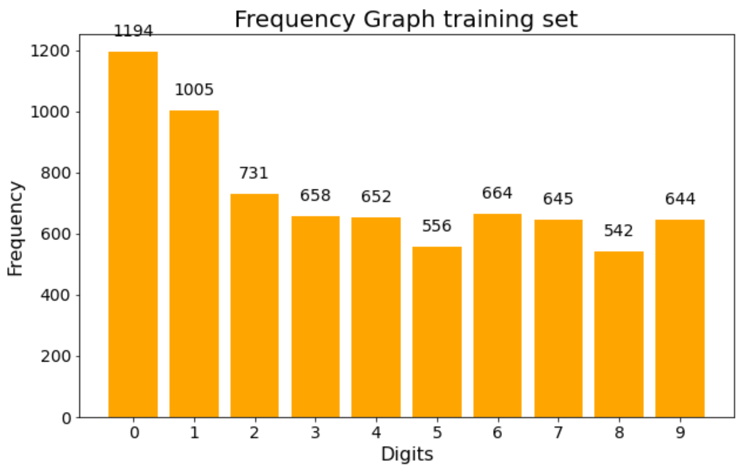

Figure 4: The Frequency Distribution of The Training Set.

3.1.4 Freqency Distribution Plots Testing Set

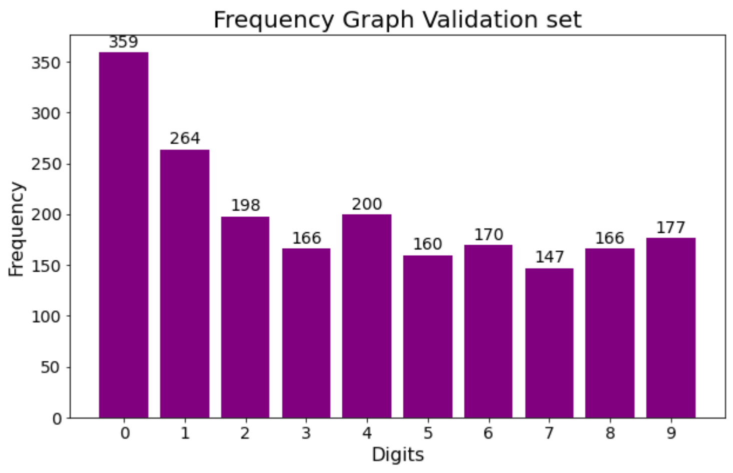

Figure 5: The Frequency Distribution of The Validation Set.

3.2 Performance Metric And Evaluation

Several performance metrics were used to ensure that our model performed accurately on the data.

3.2.1 Confusion Matrix

A confusion matrix[5] is used to show how accurate the images have been classified with their true values. The leading diagonal in the matrix shows the number of correct prediction for each handwritten image. This is clearly shown in figure 6.

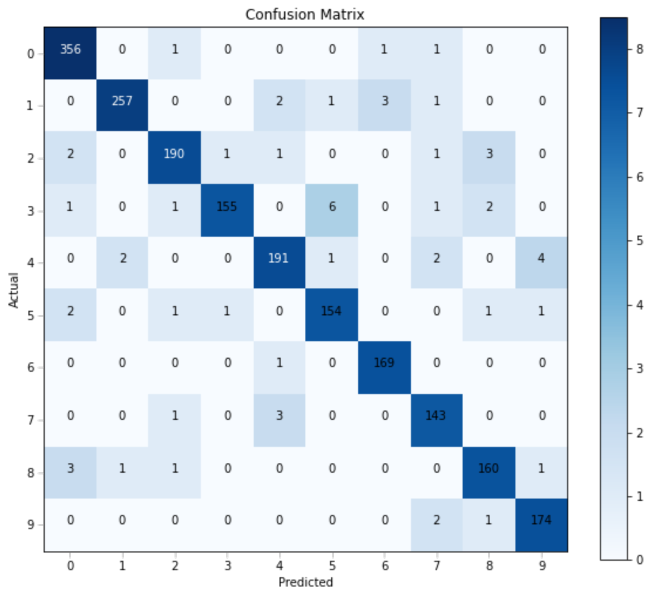

Figure 6: The confusion matrix of the validation set.

3.2.2 Training and Validation Loss/Accuracy

Training and validation loss/accuracy is used to measure the performance of the model [4]. The validation and training loss should be as low as possible. They are used to observe for over-fitting and under-fitting. There should not be much differences in the accuracy curves as well as the loss curves.

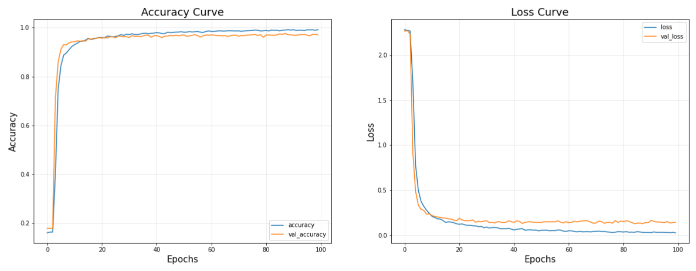

Figure 7: The confusion matrix of the validation set.

3.2.3 Class based accuracy And Overall accuracy

Class based accuracy refers to the ratio of correct predictions for each class to the number of samples in that class. Overall accuracy refers to the ratio of total number of correct predictions to the total samples in the set. They are usually expressed as a percentage [10].

3.3 Results

The model was trained for 100 epochs with a batch-size of 256. After the prediction of the validation set, the following accuracies and error rates were obtained as in the figure 8 below:

	Class Labels									
	0	1	2	3	4	5	6	7	8	9
Error Rate (%)	2.8	1.2	2.6	1.3	3.5	5.0	2.3	5.3	4.2	3.3
Class Accuracy (%)	97.2	98.8	97.4	98.7	96.5	95.0	97.7	94.7	95.8	96.7

Figure 8: Class accuracy And Error Rate.

The model achieved an overall accuracy of 97.11% with an over-all error rate 0f 2.89%. The maximum accuracy and minimum error rate was in class label 1 (98.8% and 1.2% respectively), while the minimum accuracy and maximum error rate was in class label 5 (95.0% and 5.0% respectively).Figure 9 shows all the samples (58 samples) which were predicted incorrectly indicating the predicted and true label.

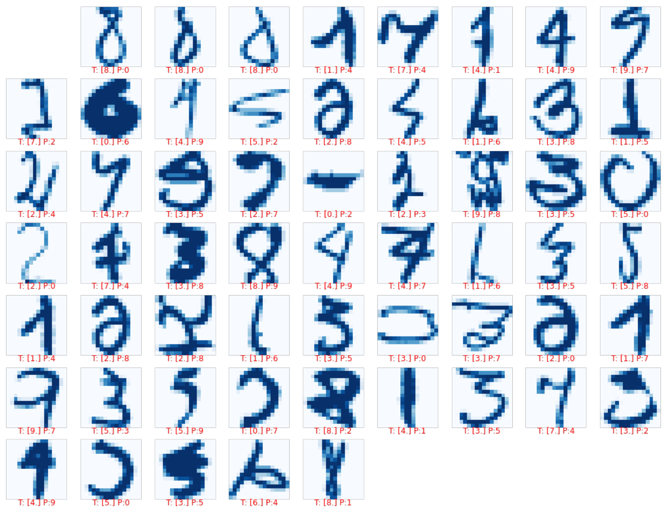

Figure 9: Samples predictly incorrectly

4 Discussion And Conclussion

4.1 Discussion

Tavanaei et al. proposed multi-layered unsupervised learning in a spiking CNN model by use of the MNIST dataset [7] to clear blur images and achieved an accuracy of 98% and range of loss in performance varied from 0.1%-8.5% [20]. Compared to the above performance, even though our dataset was totally different from the MNIST dataset as the handwritten digits in envelops contained a lot of noisy and ineligible digits, our model was still able to achieve a significantly high accuracy of almost 98%. Nonetheless; the error rate ranged between 1.2%-5.0% which is slightly better than that of the MNIST model, hence this model has been found to be effiecient and worth implementing for digit recognition systems.

4.2 Conclussion

In this paper, we observed the accuracy of the model after 100 epochs using convolution network architecture. The results have been more than promising as we achieved an accuracy of 97.11%. In future, we expect better improvements to the model so that it is able to model correctly even more noisy images. We also plan to observe the effect of ensembling in handwritten digit recognition and the effect of varying the number of hidden units and batch size on the overall accuracy of the model.

5 References

References

[1] R. B. Arif A. B. Siddique, M. M. R. Khan 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*. International Conference on Electrical Engineering and Information & Communication Technology iCEEiCT, Boston, MA, fourth edition, 2018.

[2] I. Sutskever A. Krizhevsky and G. E. Hinton. Imagenet classiffication with deep convolutional neural networks.

[3] Y. Kim C. C. Park and G. Kim. *Retrieval of sentence sequences for an image stream via coherence recurrent convolutional networks,*. IEEE transactions on pattern analysis and machine intelligence, Boston, MA, fourth edition, 2018.

[4] J. Masci L. M. Gambardella D. C. Ciresan, U. Meier and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[5] Y. LeCun et al. Backpropagation applied to handwritten zip code recognition, 1989.

[6] D. Hubel and T. Wiesel. Aberrant visual projections in the siamese cat. *The Journal of physiology*, 218(1):33–62, 1971.

[7] S. Cheon K. B. Lee and C. O. Kim. A convolutional neural network for fault classification and diagnosis in semiconductor manufacturing processes. *IEEE Transactions on Semiconductor Manufacturing*, 30(2):135–142, 2017.

[8] E. Kussul and T. Baidyk. Improved method of handwritten digit recognition tested on MNIST database. *Image and Vision Computing*, 22(12):971–982, 2004.

[9] Y. Liu and Q. Liu. Convolutional neural networks with largemargin softmax loss function for cognitive load recognition. *2017 36th Chinese Control Conference (CCC)*, 30(2):4045–4049 :IEEE, 2017.

[10] K. G. Pasi and S. R. Naik. Effect of parameter variations on accuracy of convolutional neural network. *International Conference on Computing, Analytics and Security Trends (CAST)*, 30(2):398–403 :IEEE, 2016.