Hand Written Digit Recognition Using Different Neural Network Architectures: A Comparative Study

Md. Saiful Bari Siddiqui*, Std. ID: 0421062551, Bangladesh University of Engineering and Technology

***Abstract*- Perhaps the most crucial aspect of any task involving Neural Networks is recognizing exactly what sort of an architecture will produce the best results. This paper presents a comparative analysis on hand written digit recognition tasks using different kinds of Neural Network architectures. To carry out the task, Convolutional Neural Networks (CNN) with different structures and kernels and also simple Feed forward Neural Networks (ANN) were implemented using the MNIST dataset consisting of 70000 greyscale images. In this study, CNNs consisting of 2 Convolutional layers followed by a Dense hidden layer achieved more than 99% accuracy and outperformed both simpler Feed forward networks and more complex CNNs with increased number of Convolutional layers and kernels. The results imply that the structure and complexity of data is to be taken into account to achieve the optimum performance. The code and implementations of this study are publicly available at https://github.com/Saiful185/Comparative-Study-on-Hand-Written-Digit-Recognition**

I. Background

From Object Recognition tasks to Natural Language Processing, Neural Networks are at the very center of Data Science and AI research works lately. One of the major domains that benefitted from the reemergence of Deep Neural Networks is OCR (Object Character Recognition) [1]. Handwritten Digit Recognition (HDR) is a snippet of OCR where instead of taking the whole character's data, HDR detects digits. Comparing to OCR, HDR is light and faster. In fields like medical, banking, student management, and taxation process, HDR possesses great flexibility.[2] HDR is a pretty basic yet tedious task to implement using traditional Machine Learning algorithms due to the distinctiveness in the style of writing. Neural Networks are particularly useful in these tasks because of the capacity to extract deep level features. The rise of CNNs [3] meant that the task is even simpler now. However, choosing the right architecture for a particular task is key to optimal performance. We discuss that in the context of HDR in this study.

II. Dataset

The MNIST dataset was used as a primary dataset to train the model, and it consists of 70,000 handwritten raster images from 250 different sources out of which 60,000 are used for training, and the rest are used for training validation.

III. Methodology

Each image in MNIST [4] Dataset is of size 28X28. At the start of the task the images are reshaped and normalized as part of preprocessing. Then the model is constructed. We use 2 to 3 Convolutional (8 to 64 kernels) and Maxpooling layers(2X2) along with 1 dense hidden layer (512 neurons) in CNN models and 2 hidden layers (512 & 256 neurons) in Feed forward models. One of the model summaries are shown in Figure 1. After this we compile and train the model for 10-15 epochs using RMSprop as optimizing algorithm and 0.001 as learning rate. Finally, we measure accuracy, loss and other metrics to evaluate the model.

IV. Results & Conclusion

We measure the training and validation accuracy for each epoch and plot it to tune the model. A CNN consisting of 2 Convolutional layers followed by a Dense hidden layer achieves 99.11% accuracy with 0.0325 log loss on validation set. This is up there with the top performing models on this dataset. CNNs with 3 convolutional layers reach decent accuracy too but slightly less than that consisting of 2 Convolutional layers. Simple Feed forward NN models also produce quite high accuracy scores but tend to overfit a bit more compared to CNN models. The model with no hidden layers, expectedly shows the worst accuracy scores by some margin. The overall performance metrics for different NN architectures are summarized in Table 1 & Figure 2 & 3.

In conclusion, it's evident that Neural networks, particularly CNNs perform very well at the complex task of Handwritten Digit Recognition due to the ability of extracting complex and sparse features, as we can see from the outputs of CNN layers in Fig. 4. It also makes sense that comparatively shallow CNNs perform better when it comes to relatively simple greyscale images like those in MNIST as the size is only 28X28 pixels, and more Maxpooling means losing pixel information in deeper layers. Also, 2 Convolutional and 1 Dense layer is enough to get almost all the features from these images.

REFERENCES

[1] Junli Gu, Maohua Zhu, Zhitao Zhou, Feng Zhang, Zhen Lin, Qianfeng Zhang, Mauricio Breternitz. "*Implementation and evaluation of deep neural networks (DNN) on mainstream heterogeneous systems*" APSys '14: Proceedings of 5th Asia-Pacific Workshop on Systems, June 2014 Article No.: 12 Pages 1–7.

[2] S. Aly and A. Mohamed, "*Unknown-Length Handwritten Numeral String Recognition Using Cascade of PCA-SVMNet Classifiers*" in IEEE Access, vol. 7, pp. 52024-52034, 2019

[3] Y. Yin, W. Zhang, S. Hong, J. Yang, J. Xiong, and G. Gui, *"Deep Learning-Aided OCR Techniques for Chinese Uppercase Characters in the Application of Internet of Things,"* in IEEE Access, vol. 7, pp. 47043-47049, 2019.

[4] L. Deng, *"The MNIST Database of Handwritten Digit Images for Machine Learning Research [Best of the Web]"* in IEEE Signal Processing Magazine, vol. 29, no. 6, pp. 141-142, Nov. 2012.

Models	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss	ROC_AUC Score	Cohen Kappa Score	F1 Score	Matthews Correlation Coefficient
CNN with 2 Conv layers	**0.9972**	**0.9911**	**0.0096**	**0.0325**	**0.9999**	**0.9901**	**0.9910**	**0.9901**
CNN with 3 Conv layers	0.9954	0.9861	0.0143	0.0639	0.9997	0.9845	0.9861	0.9846
ANN with 2 hidden layers	0.9931	0.9797	0.0343	0.1805	0.9991	0.9774	0.9795	0.9775
ANN with no hidden layer	0.9299	0.9273	0.2807	0.3033	0.9932	0.9192	0.9263	0.9192

Table 1: Performance Metrics Summary for all Models

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_28 (Conv2D)           (None, 26, 26, 8)         80
_________________________________________________________________
max_pooling2d_28 (MaxPooling (None, 13, 13, 8)         0
_________________________________________________________________
conv2d_29 (Conv2D)           (None, 11, 11, 16)        1168
_________________________________________________________________
max_pooling2d_29 (MaxPooling (None, 5, 5, 16)          0
_________________________________________________________________
flatten_11 (Flatten)         (None, 400)               0
_________________________________________________________________
dense_22 (Dense)             (None, 256)               102656
_________________________________________________________________
dense_23 (Dense)             (None, 10)                2570
=================================================================
Total params: 106,474
Trainable params: 106,474
Non-trainable params: 0
_________________________________________________________________
```

Figure 1: Model summary for CNN with 2 Conv layers

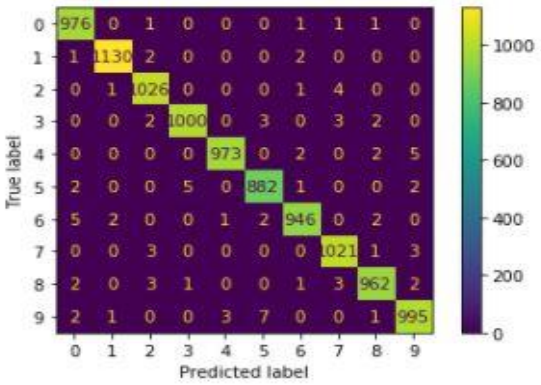

Figure 2: Confusion Matrix for CNN with 2 Conv layers

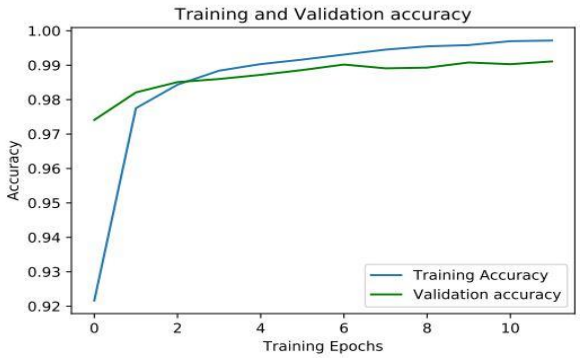

Figure 3: Validation Accuracy vs Epochs for CNN with 2 Conv layers

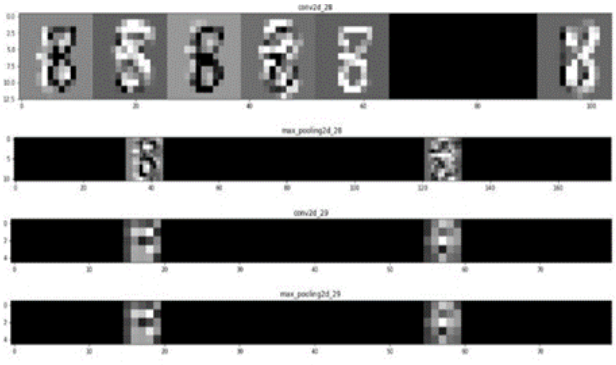

Figure 4: CNN Conv layers extracting sparse features from image