Deep Learning for Digit Recognition

Manoj Krishna Marupudi
Dept. of Electrical and Computer Engineering
Rowan University
Glassboro, NJ, USA
manojk35@students.rowan.edu
916308731

Mehdi Barhoumi
Dept. of Mechanical Engineering
Rowan University
Glassboro, NJ, USA
barhoumi@rowan.edu
916275064

Shahjahan Ahmed
Dept. of Electrical and Computer Engineering
Rowan University
Glassboro, NJ, USA
ahmeds56@students.rowan.edu
916308575

***Abstract*— Deep learning methods had achieved high performance in image classification. In the recent times, Researchers even started developing it to object detection issues. One type of deep learning is Convolutional Neural Network, which is most commonly applied to analyze visual imagery. Where the convolution is replaced the general matrix multiplication in standard NNs, hence, the number of weights is decreased and the complexity of the network is reduced. Inputs can be directly imported to the network without the need for extracting features so the CNN model requires minimal preprocessing. In this project, we have a multiple digits in the images that are present in the dataset given to us. Firstly, we have masked the digits using the OpenCV code. Later, we have introduced two architectures. One is the standard ResNet18 architecture and the other one is the LenNet - a basic Convolutional Neural Network introduced by us to train the dataset.**

Keywords— Convolution, Convolutional Neural Network, Deep learning, Data Pre-processing, Masking.

I. Introduction

Data pre-processing:
Data preprocessing is a technique that involves transforming raw dataset into a clean dataset format. The more disciplined you are in your handling of data, the more consistent and better results you are likely to achieve. The process for getting the data ready is a time-consuming process and is generally divided in these three steps.

1. Select the data
2. Preprocess the data
3. Transform the data

After you have selected the data, you need to consider how you are going to use the data. This preprocessing step is about getting the selected data into a form that you can work. Three common data preprocessing steps are Formatting, Cleaning and Sampling. The detailed explanation of our work is mentioned in the project description.

OpenCV:
Visual information is the most important type of information perceived, processed and interpreted by the human brain. Image processing is a method to perform some operations on an image, in order to extract some useful information from it. OpenCV (Open Source Computer Vision Library) is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products.

ResNet18:
ResNet, short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model was the winner of ImageNet challenge in 2015. The fundamental breakthrough with ResNet was it allowed us to train extremely deep neural networks with 150+layers successfully. Prior to ResNet training very deep neural networks was difficult due to the problem of vanishing gradients. The main idea of ResNet architecture is introduction of skip connections and without the skip connections. In some problem, introducing the skip connection gives us good accuracy and in some cases it might give poor accuracy.

II. Project Description

Formatting:
The data you have selected may not be in a format that is suitable for you to work with. In this project, our first goal is to mask the digits and save it as different images in the selected location. We used the OpenCV code for the above operations.

In our project, we have done the following operations using the OpenCV package and they are:

1. Selecting the dataset and finding number of images present in it.
2. Resizing all the present smaller images 25x25 into the bigger images 100x100.
3. Introducing the masking of resized image in three locations of an 100x100 image and saving those cropped as different images in a selected path. Each image size now we have is 18x100. That is, our input image size is now 18x100.
4. Performing the above operations for all images.

We are now having around 200,000 images combined with noisy images, the images that are unpredictable and the clear images.

Cleaning: Cleaning data is the removal of the unclear data or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. In our project, the digits which are noisy, which we think that not useful and which digits are unpredictable are removed and finally the cleaned dataset is used to train and test our network model. It taken so much time to clean the dataset for us.

After cleaning the dataset, we kept the images and organized in the way that all the individual predicted digits will be in separated file location.

III. Network Architecture

After the cleaning the data, the input size of our image is 18x100. In this project, we tried to use two architectures. One is the In-built architecture present in the keras "ResNet18" and the other one is the basic CNN introduced by us "LenNet". The input image size is again resized to the 64x64x3 by using the OpenCV operations in keras. In our Architecture, we firstly kept three convolutional layers with 32 filters of size 11x11,7x7, 5x5 respectively. The activation function we used is ReLu. Later, we introduce the Max-Pooling layer of pool size 5x5. After that, the whole network is again feed with two convolutional layers where each layer has 32 filters with the filter size of 3x3. The resultant network is passed with the Max-pooling layer with the pooling size of 3x3 and followed by the dense layer. We used the dropout regularizers also. At the end, Softmax activation function is introduced. The loss function we used is Categorical Cross Entropy and the optimizer rule we used is Adam. Below fig 1 and fig 2 will explain the block diagram of our network and the architecture of the ResNet18 respectively.

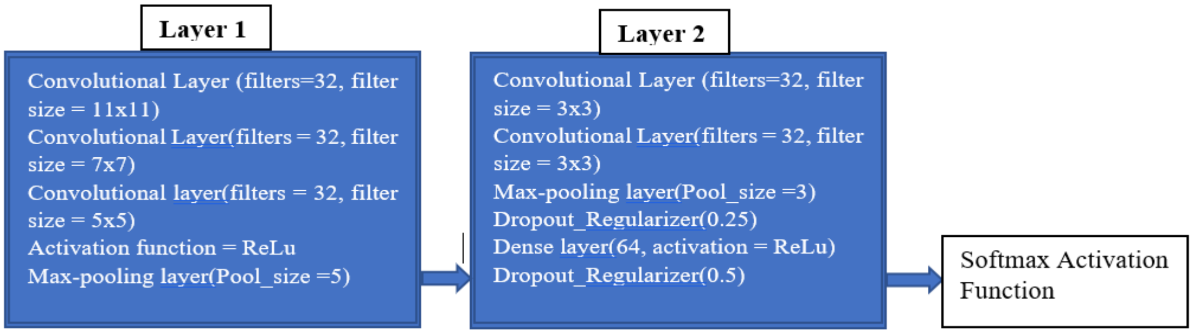
Layer 1
Convolutional Layer (filters=32, filter size = 11x11)
Convolutional Layer(filters = 32, filter size = 7x7)
Convolutional layer(filters = 32, filter size = 5x5)
Activation function = ReLu
Max-pooling layer(Pool_size =5)
Layer 2
Convolutional Layer (filters=32, filter size = 3x3)
Convolutional Layer(filters = 32, filter size = 3x3)
Max-pooling layer(Pool_size =3)
Dropout_Regularizer(0.25)
Dense layer(64, activation = ReLu)
Dropout_Regularizer(0.5)
Softmax Activation Function

Fig 1. Block diagram of our Proposed Network architecture

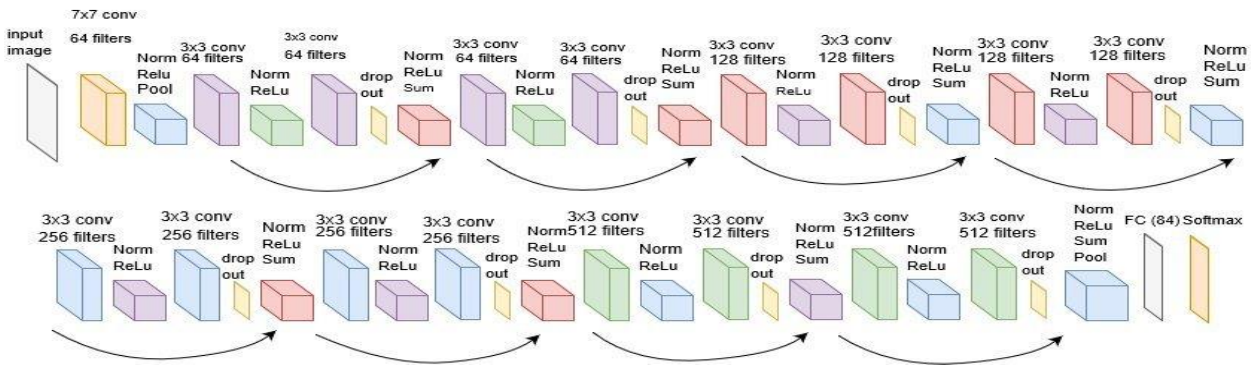

Fig 2. Network Architecture of ResNet18

IV. Results and Discussions

We have cleaned dataset effectively and we concluded with 11,811 images in it where we training dataset is having 10,654 images and the validation dataset is having 1157 images. We have used an inbuilt network model “ResNet18” first and we found that the training accuracy is about 90 percent and the validation accuracy is around 23 percent. By looking into it, we understand that the ResNet18 model is overfitting. Later we tried to prepare our own network model “LenNet” and it given us the training accuracy of around 70 percent and the validation accuracy of around 33 percent. The number of epochs we kept is 20. We written our code in such a way that the accuracies will be stored in the excel file and We plot the graphs from the excel sheets. Fig 2 and Fig 3 below shows the graphs of the Accuracies of the ResNet18 and LenNet with respect to the Epochs respectively.

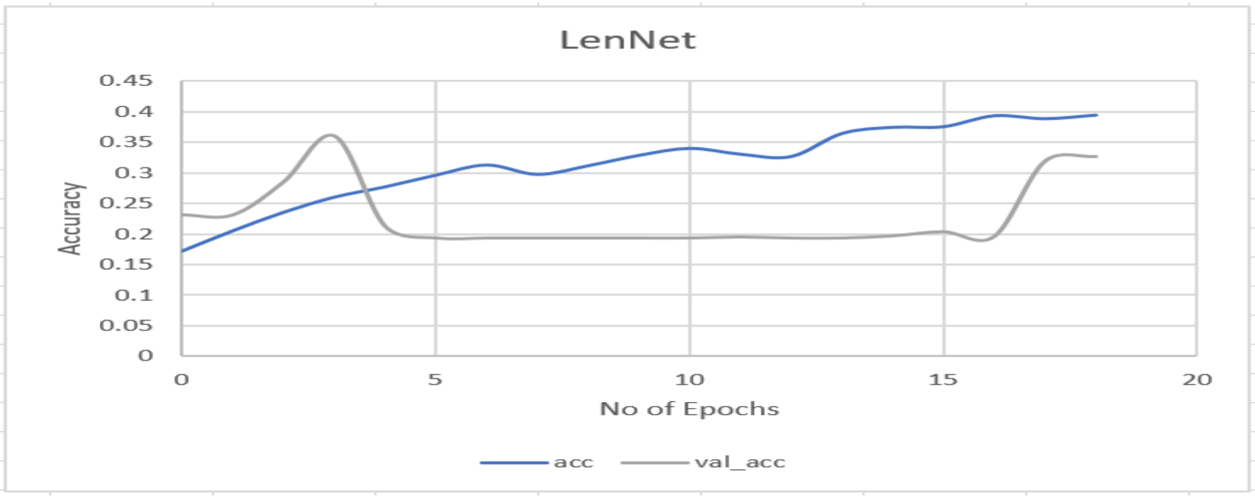

Fig 3. Accuracy Vs No of Epochs graph for LenNet model

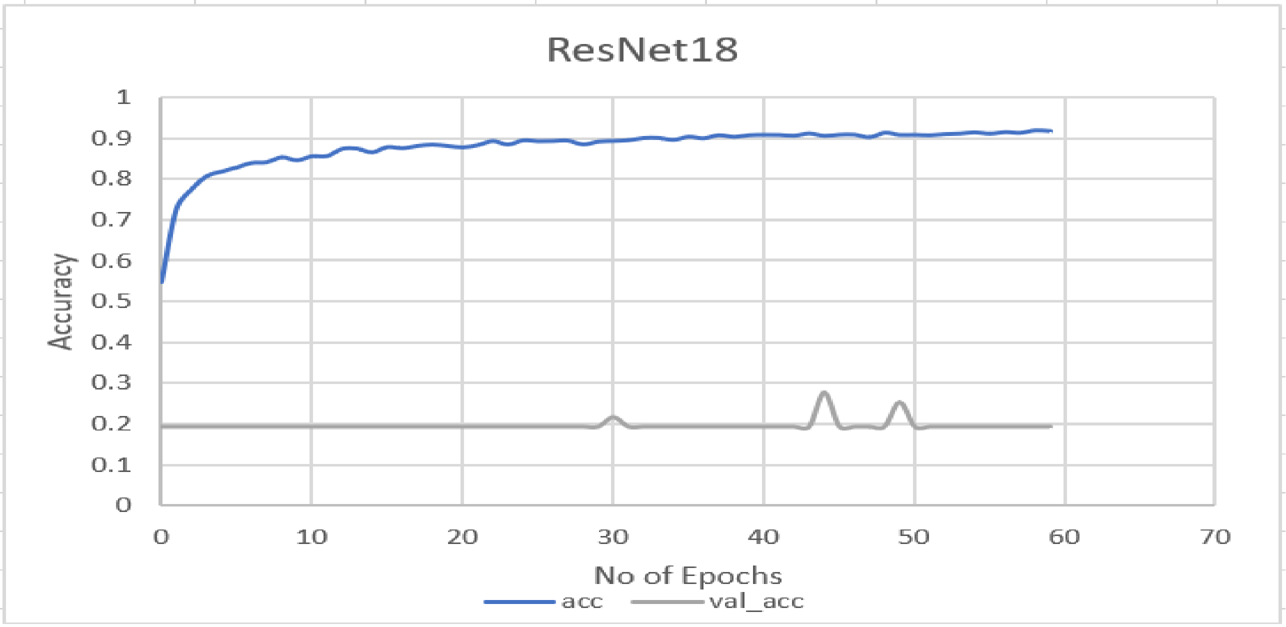

Fig 4. Accuracy Vs No of Epochs graph for ResNet18 model

V. Conclusion

Seeing the above results, we understand that the ResNet18 model is overfitting for our dataset. We assume that the main reason for producing the low validation accuracy is that the ResNet18 architecture requires very huge data and it has large number of parameters. Later we tried to prepare our own network model “LenNet”. We are happy with the result as our model not overfitting like the ResNet18.

Github Link: https://github.com/Mah-douch/Digit-recognition?fbclid=IwAR38qVOmSfl_9n4-okDw2RZU9obLdsFWs16NUdjc9kmgyYWJJdtFUeEAG0s