Handwritten Digit Recognition Using CNN

Mayank Jain
Electronics and Communication Engineering Department
Amity University
Greater Noida, India
mayank14@s.amity.edu

Gagandeep Kaur
Electronics and Communication Engineering Department
Amity University
Greater Noida, India
gkaur@gn.amity.edu

Muhammad Parvez Quamar
Electronics and Communication Engineering Department
Amity University
Greater Noida, India
muhammad.quamar@s.amity.edu

Harshit Gupta
Electronics and Communication Engineering Department
Amity University
Greater Noida, India
harshit.gupta29@s.amity.edu

2021 International Conference on Innovative Practices in Technology and Management (ICIPTM) | 978-1-6654-2530-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICIPTM52218.2021.9388351

***Abstract* - The issue of transcribed digit acknowledgment has for some time been an open issue in the field of example order. A few examined have demonstrated that Neural Network has an incredible execution in information arrangement. The fundamental target of this paper is to give effective and solid procedures to acknowledgment of transcribed numerical by looking at different existing arrangement models. This paper thinks about the exhibition of Convolutional Neural Network (CCN). Results demonstrate that CNN classifier beat over Neural Network with critical improved computational effectiveness without relinquishing execution. Handwritten digit recognition can be performed using the Convolutional neural network from Machine Learning. Using the MNIST (Modified National Institute of Standards and Technologies) database and compiling with the CNN gives the basic structure of my project development. So, basically to perform the model we need some libraries such as NumPy, 'Pandas', TensorFlow, Keras. These are the main structure on which my main project stands. MNIST data contains about 70,000 images of handwritten digits from 0-9. So, it is a class 10 classification model. This dataset is divided into 2 parts i.e. Training and Test dataset. Image representation as 28*28 matrix where each cell contains grayscale pixel value.**

Keywords - CNN, MNIST dataset, Machine Learning, Handwritten Digit

I. INTRODUCTION

The issue of manually written numerals acknowledgment has been broadly concentrated lately and the huge amount of pre-processing strategies and arrangement calculations have been created. Notwithstanding, transcribed numerals acknowledgment is as yet a test for us. The primary trouble of transcribed numerals acknowledgment is the genuine change in size, interpretation, stroke thickness, pivot and twisting of the numeral picture as a result of written by hand digits are composed by various clients and their composing style is not quite the same as one client to another. A few considered have utilized various approaches to manually written digit with various AI procedures Khotanzad et at (1998) who have applied the ideas of Machine Learning and Neural Networks to perceive and decide the transcribed digits from its picture. This investigation has indicated that digit acknowledgment is an amazing model issue for finding out about neural organizations and it gives an extraordinary method to grow further developed strategies like profound learning. Transcribed acknowledgment (HWR) is the capacity of a PC to get and comprehend understandable manually written contribution from sources, for example, paper archives, client input contact screens and different gadgets.[1] The picture of the composed content might be detected from a bit of paper by optical filtering (optical character acknowledgment) or canny word acknowledgment or by client input.

Then again, the developments of the pen tip might be detected "on line", for instance by a pen-based PC screen surface, a for the most part simpler undertaking as there are more hints accessible This paper presents perceiving the manually written digits (0 to 9) from the renowned MNIST dataset utilizing TensorFlow framework (library) and python as language and its libraries as client enters the particular digit the machine would perceive and show the outcomes with exactness rate.

II. DATA SOURCE MODULE

Manually written digit acknowledgment (HDR) is viewed as one of trifling and basic AI issues. It has been utilized generally by analysts as trials for speculations of AI calculations for a long time. Lately, neural organizations and regular neural organization right now give the best answers for some issues in manually written digit acknowledgment.[2] An epic mixture CNN–SVM model for manually written digit acknowledgment is planned by. This crossbreed model consequently extricates highlights from the crude pictures and creates the expectations. For this work, the creator utilized non-immersing neurons and an extremely productive GPU execution of the convolution activity to decrease overfitting in the completely associated layers. To upgrade technique proposed in, handled basic examinations to reduce impediment acquired from. The creator presents a novel representation strategy that gives knowledge into the capacity of highlight layers and the

978-0-7381-3289-1/21/$31.00 ©2021 IEEE

methodology of the classifier have watched convolutional net design that can be utilized in any event, when the measure of learning information is restricted.[3] Have utilized new organization structure, called Spatial Pyramid Pooling SPP-net, can create a fixed-length portrayal paying little mind to picture. Multi-segment DNN (MCDNN) utilized MNIST digits. The outcome has a low 0.23% blunder rate. Hayder M. Albeahdili et al. have played out another CNN engineering which accomplishes cutting edge arrangement results on the distinctive test benchmarks. The blunder rate for this methodology is 0.39 % for MNIST dataset.

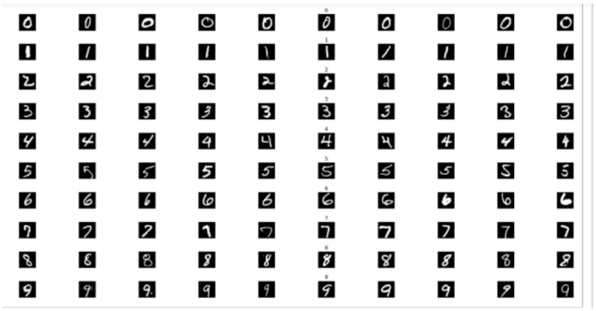


Fig 1. MNIST Dataset

Our digit dataset contains about 70,000 images of digits 0-9. This is further divided into two parts i.e., Training set data and Testing set data.[4] The training dataset contains about 60,000 handwritten digits and the test dataset contains about 10,000 handwritten digits.

Pre-processing the dataset. We will change some parameters such as colour of the images, size of the images to ease our processing. Next step will be to build the model that will help us in prediction. Here comes the part of CNN. The model is based on CNN and all the layers of CNN. After building the model it's time for training and testing the model and also check the accuracy that it's providing in the validation set. At the end, it's time for evaluation and prediction. We will evaluate the model and then start with our prediction.

III. DIGITAL IDENTIFICATION MODULE

A. *Pre-Processing*

When we are required to build a predictive model, we have to look and manipulate the data before we start modelling which includes multiple pre-processing steps such as importing the images, changing the size of the images, changing the colour of the images, visualizing the image dataset and converting them from categorical to vector form.[5] All these steps are combined in total under one thing that is knownas Exploratory Data Analysis. We are performing these steps to ease our computing speed and reduce the complexity of the model.

```
In [13]: #Normalising the pixel values
         #For this,first convert the integer values to float
         x_train=x_train.astype("float32")
         x_test=x_test.astype("float32")
         x_train/=255
         x_test/=255
         print(x_train.shape)
         print("x_train sample :",x_train.shape[0])
         print("x_test sample:",x_test.shape[0])

         (60000, 28, 28, 1)
         x_train sample : 60000
         x_test sample: 10000
```

Fig. 2 Pre-Processing Data

In the reference to Fig 2, we have converted our image from RGB to Gray scale for easy computation by dividing with 255-pixel value to get the value between 0-1.[6] In this process we are also converting the data from categorical value to vector form or binary form.

B. *Principle of Convolutional neural network*

After we are done with the pre-processing part, the next step is to create the CNN model. CNN stands for Convolutional Neural Network. CNN consists of 4 hidden layers which help in extraction of the features from the images and is able to predict the result.[7] The layers of CNN are (a) Convolutional Layer (b) ReLu Layer (c) Pooling Layer (d) Fully Connected Layer.Reason we are using CNN is because the fundamental favourable position of CNN contrasted with its archetypes is that it consequently recognizes the significant highlights with no human management.

1. *Convolutional Layer*

Convolutional layer is a simple application of a filter which acts as an activation function.[8] What this does is takes a feature from a input image, then filter different features from that image and makes a feature map. Some of the features are location, strength etc. the filter is then moved over the whole image and the value of each pixel is calculated.

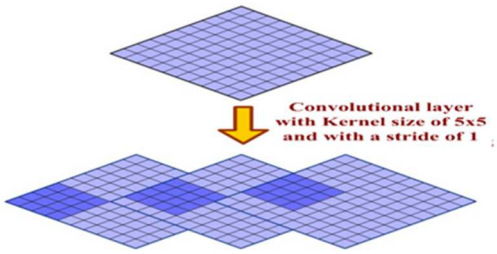


Fig. 3 Representation of Kernel and Stride in a Convolutional Layer

2. *Relu Layer*

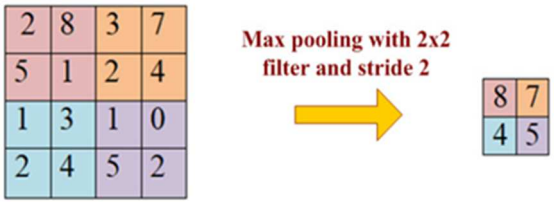
In simple language, the function of ReLu layer is to remove all the negative pixel values from the

image and replace them with zero. This is done to avoid the summing up of pixel values to zero.

$$R(z) = \begin{Bmatrix} z & z > 0 \\ 0 & z <= 0 \end{Bmatrix} \quad (1)$$

3. *Pooling Layer*

The main function of this layer is to shrink the image size. This is done to ease the computation speed and also decrease the computational cost.[9] What this layer basically does is takes a matrix of 2 x 2 and stride (movement from one pixel to another) of 1 and move the window to whole of the image. In each of the window the highest value is taken and this process in carried on to each part of the image. Taking an example, if before pooling layer we had a matrix of 4 x 4 then after pooling layer the image matrix will reduce to 2 x 2 matrix.



Fir. 4 Pooling Layer

4. *Fully connected Layer*

This is the last layer of CNN. This is the part where the actual classification happens. All the matrix from the pooling layer is stacked up here and put into a single list. The values which are higher are the points of prediction for the given image.

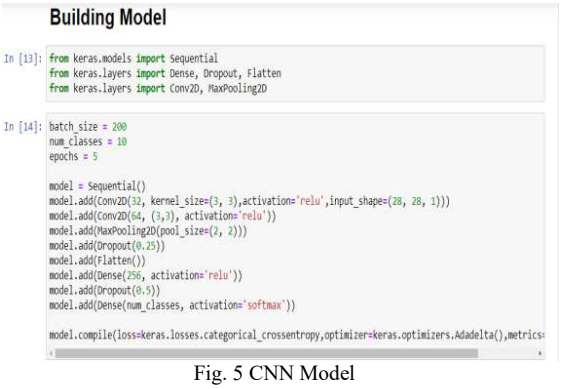


Fig. 5 CNN Model

5. Above is the visualization of the CNN model with all necessary libraries included. Sequential is used to keep all the processes in a sequence one after another. The code Dropout is used to dropout values from the dataset to reduces the over fitting.

C. Classification of Recognition results

So, after building the model it's time for evaluating the model. In this part of the project, we must check if the above built model is working to our expectation. But before that we first need to train the model on the evaluation dataset.

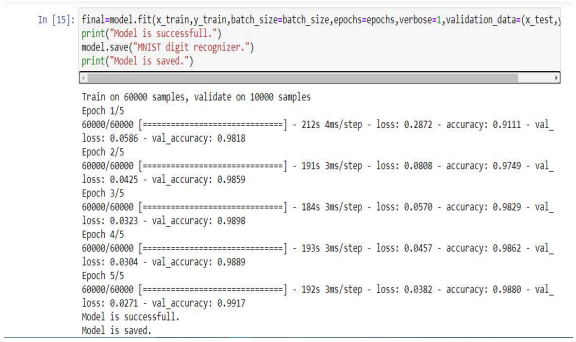


Fig. 6 Training and Testing of CNN Model

In the above image, we can clearly see the training and testing of the model. I have taken 5 epochs (iterations) so that my model doesn't take the same 5 values again and again and train on that. So, I have taken a batch of 200 with the nun class of 10 (because we have images from 0-9) and an epoch of 5. We are training our 60,000 images and validating on the 10,000 images [10]. As the process starts, we can see the results of validation accuracy and validation loss. With each epochs our validation accuracy increases. After the evaluation is done, we can see the result of our model on the test datasets.

IV. RESULTS

The test after effects of the MNIST transcribed digit dataset utilizing various boundaries of CNN models are recorded and broke down the discoveries of approval affirm the part of various engineering boundaries on the presentation of our acknowledgment framework. The preparation boundary utilized here has a learning pace of 0.02 and epochs of 5. The most elevated acknowledgment exactness accomplished is with CNN design having four layers, is 99.16% for the component.

Fig. 7 Evolution

The goal of the current work is to completely explore all the boundaries of CNN design that convey best acknowledgment exactness for a MNIST dataset. Generally speaking, it has been watched that the proposed model of

CNN design with three layers conveyed better acknowledgment exactness of 99.16% with the Adam streamlining agent.

V. PREDICTION

A. *From Test Dataset*

This is the final part of my project where we will check if our model is predicting the digits accurately and with what percentage it is correctly predicting it. So, what I have done here is I have extracted some of the images from the test dataset and kept it into a folder.

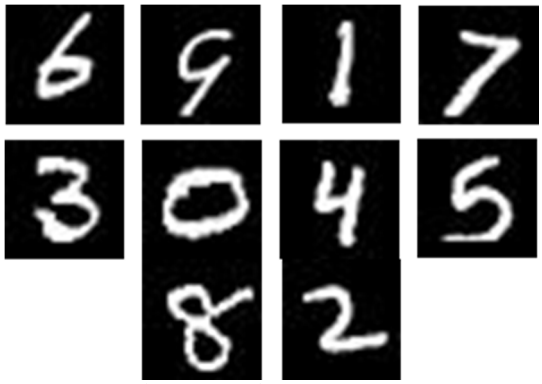


Fig. 8 Images from Test Dataset

Now my prediction code with take all the images from that folder and starting pre-processing the images and then will start to predict the images.

```
In [10]: im=[]
         image_path = "C:/Users/bhans/sample_image"
         for img in os.listdir(image_path):
            # load the image
            img = os.path.join(image_path, img)
            img= image.load_img(img, color_mode="grayscale", target_size=(28, 28))
            # convert to array
            img =image.img_to_array(img)
            img = img.reshape(1,28,28,1)
            # prepare pixel data
            img =img.astype('float32')
            img =img / 255.0
            im.append(img)
         im = np.vstack(im)

In [11]: model = load_model("MNIST digit recognizer.")
         filenames=[]
         for filename in os.listdir(image_path):
            classes = model.predict_classes(im , batch_size=10)
            filenames.append(filename)
         print(filenames)
         print(classes)

         ['A.jpg', 'I.jpg', 'M.jpg', 'N.jpg', 'R.jpg', 'S.jpg', 'T.jpg', 'U.jpg', 'V.jpg', 'Y.jpg']
         [7 2 5 8 0 4 1 9 6 3]
```

Fig. 9 Prediction

As we can clearly see that our model is working up to the mark with the prediction rate at 99% accurate. These were the images form the test dataset. All the images were predicted correctly from that folder. Now taking this model prediction to next stage. It's time to give images from user end side.

B. *Images from User's side*

We have seen that the model is performing well with the images from the test dataset and now it's time to take the prediction model to next stage. We will be adding images from user's end and will check if the model is predicting those images or not.

What I have done here is created some digit images on MS PAINT. I have created images of digits 0-9 on MS PAINT. Keeping the background black and digit with white to keep the model working. Surprisingly, I was shocked with the results. The accuracy was up to the mark. The model predicted all the images correctly with an accuracy of 100%.

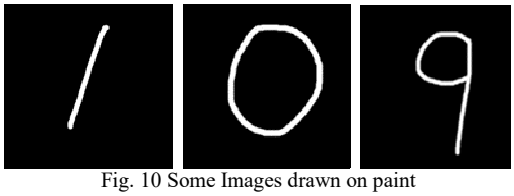


Fig. 10 Some Images drawn on paint

These are the images created on MS PAINTS. Now I will put all the images in the folder and start predicting the images. Let's see what the model predicts.

```
In [13]: model = load_model("MNIST digit recognizer.")
         filenamess=[]
         for filenamee in os.listdir(image_paths):
            classes = model.predict_classes(imq , batch_size=10)
            filenamess.append(filenamee)
         print(filenamess)
         print(classes)

         ['0.jpg', '1.jpg', '2.jpg', '3.jpg', '4.jpg', '5.jpg', '6.jpg', '7.jpg', '8.jp
         g', '9.jpg']
         [0 1 2 3 4 5 6 7 8 9]
```

Fig. 11 Prediction of User's images

As we can see in the fig 9, prediction of the images from the user end is predicted 100% accurately.

VI. CONCLUSION

In this work, with the point of improving the exhibition of transcribed digit acknowledgment, we assessed variations of a convolutional neural organization to keep away from complex pre-preparing, exorbitant component extraction and a perplexing troupe (classifier blend) approach of a conventional acknowledgment framework. Through broad assessment utilizing a MNIST dataset, the current work recommends the job of different hyper-boundaries. We additionally confirmed that tweaking of hyper-boundaries is fundamental in improving the presentation of CNN engineering. We accomplished acknowledgment pace of 99.89% with the Adam analyzer for the MNIST information base, which is superior to all recently revealed outcomes.

The impact of expanding the quantity of convolutional layers in CNN design on the presentation of transcribed digit acknowledgment is unmistakably introduced through the tests. The oddity of the current work is that it altogether explores all the boundaries of CNN engineering that convey best acknowledgment precision for a MNIST dataset. Companion scientists couldn't coordinate this precision utilizing an unadulterated CNN model. A few analysts utilized gathering CNN network models for the equivalent dataset to improve their acknowledgment precision at the expense of expanded computational expense and high testing multifaceted nature yet with practically identical exactness as accomplished in the present work.

In future, various designs of CNN, in particular, cross breed CNN, viz., CNN-RNN and CNN-HMM models, and space explicit acknowledgment frameworks, can be researched. Developmental calculations can be investigated for streamlining CNN learning boundaries, to be specific, the quantity of layers, learning rate and portion sizes of convolutional channels.

REFERENCES

[1] Niu, X.X.; Suen, C.Y. A novel hybrid CNN–SVM classifier for recognizing handwritten digits. Pattern Recognit. 2012, 45, 1318–1325.

[2] Long, M.; Yan, Z. Detecting iris liveness with batch normalized convolutional neural network. Compute, Mater. Contin. 2019, 58, 493–504.

[3] Y. LeCun et al., "Backpropagation applied to handwritten zip code recognition," Neural computation, vol. 1, no. 4, pp. 541-551, 1989.

[4] Sueiras, J.; Ruiz, V.; Sanchez, A.; Velez, J.F. Offline continuous handwriting recognition using sequence to sequence neural networks. Neurocomputing. 2018, 289, 119–128.

[5] Wells, Lee & Chen, Shengfeng&Almamlook, Rabia&Gu, Yuwen. (2018). Offline Handwritten Digits Recognition Using Machine learning.

[6] Burel, G., Pottier, I., &Catros, J. Y. (1992, June). Recognition of handwritten digits by image processing and neural network. In Neural Networks, 1992. IJCNN, International Joint Conference on (Vol. 3, pp. 666-671) IEEE.

[7] Salvador España-Boquera, Maria J. C. B., Jorge G. M. and Francisco Z. M., “Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 33, No. 4, April 2014.

[8] Ahmed, M., Rasool, A. G., Afzal, H., & Siddiqi, I. (2017). Improving handwriting-based gender classification using ensemble classifiers. Expert Systems with Applications, 85, 158-168.

[9] Sadri, J., Suen, C. Y., & Bui, T. D. (2007). A genetic framework using contextual knowledge for segmentation and recognition of handwritten numeral strings. Pattern Recognition, 40(3), 898-919.

[10] Sarkhel, R., Das, N., Das, A., Kundu, M., &Nasipuri, M. (2017). A multi-scale deep quad tree based feature extraction method for the recognition of isolated handwritten characters of popular Indic scripts. Pattern Recognition, 71, 78-93.