Handwritten Digit Recognition Using CNNs

Rishab Lokray, rishab.lokray@ufl.edu *9357-3447* Siddharth Jain, siddhartjain@ufl.edu *7185-1164*

***Abstract*— There are multiple ways to implement character recognition, Firstly there are 2d Point cloud recognizers and CV techniques which match the 2d point clouds of the test data with the training data, Secondly there are Machine Learning algorithms that help recognize different images/shapes. With the advent of neural networks there has been a consistent improvement in machine learning algorithms that can be used for image classification and pattern recognition. We read papers on multiple classification techniques like baseline nearest neighbor classifier, radial basis function networks, multilayer perceptron and CNN. We settled on CNN as it has become a very powerful tool in image recognition/classification over the last few decades. In this paper we solve the task at hand of digit classification using a Convoluted Neural Network. User generated training data and the popular MNIST data set has been used to train a CNN model that can recognize digits {0,1,2,3,4,5,6,7,8,9}. This paper aims at developing a CNN to successfully recognize digits while experimenting with model parameters like image size, number of hidden layers, number of convolution layers, pooling layers, epochs, learning rates etc. We also try to discuss a solution to design a model for a hard test data set where we need to classify random images not belonging to the 0-9 set.**

***Index Terms*— Machine Learning, Neural Networks, CNN, ANN, Deep Learning.**

I. Introduction

The means to convert any physical/on paper text, either printed or handwritten into electronic data can be very helpful for many applications. Image recognition techniques have come a long way, from being able to detect only a few perfectly typed black and white characters, to being able to predict multiple patters with a very commendable accuracy. Earlier, the researchers were mainly limited by the computation power available at that time i.e. GPUs and storage. While in today's date everyone has direct access to super-fast processors and huge terabytes of data sets that help train machine learning algorithms. The principles behind digit recognition form the base for any character recognition OCR systems which can be used to digitize all types of important data and forms. After reading papers on different ways of carrying out the task of handwritten character recognition, we could narrow down to the two most general approaches. First being extracting features from images and using computer vision techniques to recognize characters. The disadvantages of this technique are that they are computationally expensive and cannot be generalized. The second and the preferred technique would be using Neural Networks. With the advancements in technology and easy access to processing power it makes sense to use powerful CNNs that can help generalize the model for varied datasets. As we need to classify inputs into 10 classes, we decided it is best to use a CNN as it learns complex patterns easily. Up next in the Approach & Implementation section we discuss how we designed the convoluted model and how we decided the model parameters: network architecture, batch size, regressors etc.

II. Approach & Implementation

A. Image preprocessing

As a part of the final project for the course Fundamentals of Machine Learning, all the students were asked to produce handwritten images of the digits 0-9. This formed a part of the training data set train.npy. However, this collection of 1200 images each with a size of 300x300(gray scale) had a lot of distorted images which had to be removed as they were negatively hampering our model. Many pictures also had the camscanner logo shadows that were adding a lot of noisy error.

The preprocessing steps were as follows; First load 300x300 data set provided in class, we crop the image by 45px on all sides as they did not hold any data, then we resize the image to 54x54 px. After that we convert the images to black and white by selecting a threshold of 0.59. Then we rotate the images by the angles in the range -45deg to +45 deg in increments of 5 degrees. After that we converted the images to 54x54x1 and normalize it by dividing by 255. We deleted about 104 images as they completely blacked out the training images. The final size is 21224x54x54x1. To make up for this deleted data we also added 1000 images of each digit from the MNIST data set. MNIST is a collection of standard 28x28 black and white digit images that had to be rescaled to 54x54x1. The total training set size was 31224x54x54x1.

We used Keras, an open source neural nets library written in python and capable of running on top of TensorFlow. Keras needs input images to be of a fixed size such that it can decide the input layer neurons. Hence, we had to bring all images to 54x54 pixels in grey scale. We chose the size 54x54 after testing various sizes which is discussed later in the paper.

The images obtained from MNIST data set were of size 28x28 and had to be made 54x54 by adding 0 padding to them. While the 300x300 images had to be scaled down to 54x54 and converted to b/w by normalizing by 255.The images are converted to 54x54x1 shape before fed to the model.

B. Input Data

The most important difference between CNN and MLP is that we don't need to extract features for the image before feeding it to the model unlike MLP where we would have to convert the image to 2916x1 array where it loses spatial representation. To facilitate this, we use a convolutional filter at the input that helps reducing the number of connections to the neurons in the following layers.

C. Neural Network Architecture.

Convolutional neural networks are the current most efficient ML algorithms to classify images. The model learns the features hierarchically. The hidden layers usually consist of convolutional layers, pooling layers, fully connected layers and normalization layers. Below are the layers we added to our model:

a. Convolution layer: Convolution layer is used to multiply the image with a 2d filter. The convolution layer consists of a x set of independent filters, each filter is convolved with the image and we end up with x feature maps. We can have a series of convolution layers that will learn separate parts of the image. In this way CNNs facilitate what is called parameter sharing where weights are the shared by all neurons in a feature map. This helps reduce the number of parameters in the network and reduces the computations.
b. Pooling layer: Its job is to reduce the spatial size of the representation to reduce the number of parameters and computation in the network. Most commonly max pooling is used.
c. Fully connected layer: After the last pooling layer we have a regular NN layer that is connected to every output from the previous layer. They help detect different features in an image and help make classification boundaries.

Apart from changing the layers and neurons, we can also change the following parameters:

a. Learning rate: This defines the step size it takes to converge to the global minima to minimize the error.
b. Momentum: Used to change the learning rate If 2 or more steps are taken in same direction.
c. Activation function: Defines the output at each layer, can be anything ranging from sigmoid, tanh, relu, softmax. Etc.
d. Regularization: Used to add weight to the cost function to add sparsity.

The architecture we chose after the simulation results shown in the next section are as follows:

1. Convolution layers – 2 filters with sizes 64 (kernel size = 3x3) and 32 (kernel size = 5x5) and activation relu.
2. Pooling – 2 Maxpooling layers after each filter.(size = 2x2)
3. Dropouts – 2, One after each pooling layers.
4. Flatten – One layer before feeding the dense layer.
5. Dense – 4 layers with network sizes (200,128,84,10)
6. Activation function: Relu in all but output and softmax in output

III. TRAINING AND VALIDATION

As we did not have a separate testing set, we used K-fold Cross Validation to test the model parameters and train the network. We used 3-fold 5-fold and 7-folds. The accuracy was as best observed for a 5-fold validation.

IV. EXPERIMENTAL RESULTS

To evaluate the network, a thorough analysis was conducted wherein all the parameters, but one was kept constant. Multiple plots were jotted to determine the best accuracy possible. Below we tabulated training accuracy, validation accuracy and testing accuracy.

a. Varying Batch size and Epochs:

We also wanted to test how the model performed on varying batch sizes we kept every other model parameter constant. We can see that it works better for small epochs with smaller batch size. It will need longer epochs with bigger batch sizes.

Batch/Epoch	Training	Validation	Testing
32/10	98.58	97.27	97.6
100/10	98.2	97	97.1
200/15	98.4	97.5	97.3
250/15	98.5	96	96.7

b. Varying Learning Rate and Momentum:

Learning rate and momentum are the two most important parameters that can be tweaked. Therefore, the following observations were very crucial for the model to work and not over fit. We kept the batch size and epoch size const to 32 and 10, respectively. We use the solver SGD() in keras to change the lr. Image size is 54x54 and the batch size is 32 with epoch 10.

Lr/mom	Train	Valid	Test
0.03/0.5	98.7	96.7	96.9
0.1/0.5	98.8	97.3	97.5
0.3/0.5	10	9	10
0.8/0.6	9.2	9	11

c. Varying adaptive learning rate methods:

We can use many learning rate optimizers. The one we chose are Adam as it is a standard and provides the best result for our configurations

Method	Train	Valid	Test
Adadelta	98.2	97	96.7
Adam	98.8	97.8	97.8
Adagrad	97.2	96.8	97.1

rmsprop	99.1	96	95

d. Varying Learning rate with K Fold:

K fold was used to better train the model and choose a model that does not over fit the model started over fitting for folds over 10.

K fold	LR	Accuracy
3	0.1	95.16
3	0.3	16.7
5	0.1	95.62
5	0.3	17.1
10	0.1	96.16

e. Varying image size:

We observed that as we used MNIST data that is of 28x28 we could not downscale our 300x300 image to 28x28 as it lost too many pixels and became noisy. We tested many image sizes and settled for 54x54 as it is smaller and has less features so that it does not take up much processing power unlike a 100x100 image and it also facilitates the construction of a simpler model that reduces the chances of overfitting.

Size	Train	Valid	Test
28x28	85	66	66
54x54	98	98	98
100x100	97	98	98

V. HARD DATA SET

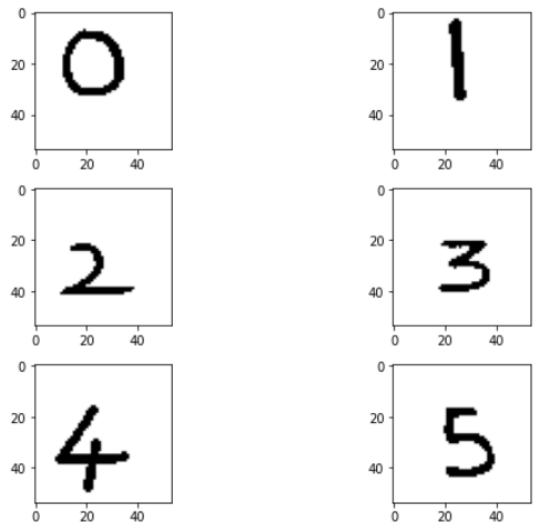
Hard data set was made of one extra class such that the classifier was made to detect false inputs and correctly classify them into the class -1 while it also correctly predicted the values of all other digits 0-9.

To help us do this we used the activation function 'Softmax' in the output layer that gives us an array of probabilities that each class belongs to. We chose a threshold for our data set and allotted an image to a class only if its probability was over that threshold. And if there was no probability over that value, we assigned it to the class -1.

VI. TESTING ON SMALL DATA SET AND HARD DATA

A. Small data set

The model was trained and tested on a set of 5 handwritten images {0,1,2,3,4,5}. We got an accuracy of 100% for it.



```
x_test Class shape: (6, 54, 54, 1)
Number of images in x_test Class 6
Loaded model from disk
Predicted Labels :  [0 1 2 3 4 5]
Original Labels :  ['0' '1' '2' '3' '4' '5']
Accuracy : 1.0
```

B. Conclusion

In this paper we have discussed the design and implementation of a Convolutional Neural Network using Python libraries in TensorFlow and keras. A model with 2 filter and 4 dense layers with Adam optimizer and softmax activation function for output layer was designed to correctly recognize handwritten digits in 10 classes. It is designed to preprocess the input data to 54x54 size and B&W in nature.
We could obtain an accuracy of 98% for training and 97.5% for testing data.

We can conclude that CNNs are one of the best classifiers for image recognition because of their multiple advantages.

C. Referencing a Figure or Table Within Your Paper

When referencing your figures and tables within your paper, use the abbreviation "Fig." even at the beginning of a sentence. Do not abbreviate "Table." Tables should be numbered with Roman Numerals.

REFERENCES

[1] L. Chen, S. Wang, W. Fan, J. Sun and S. Naoi, "Beyond human recognition: A CNN-based framework for handwritten character recognition," 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), Kuala Lumpur, 2015, pp. 695-699.

[2] C. Wu, W. Fan, Y. He, J. Sun and S. Naoi, "Handwritten Character Recognition by Alternately Trained Relaxation Convolutional Neural Network," *2014 14th International Conference on Frontiers in Handwriting Recognition*, Heraklion, 2014, pp. 291-296.

[3] A. Yuan, G. Bai, L. Jiao and Y. Liu, "Offline handwritten English character recognition based on convolutional neural network," 2012 10th IAPR International Workshop on Document Analysis Systems, Gold Cost, QLD, 2012, pp. 125-129.