Classification of Handwritten Digits using Convolutional Neural Networks

Janshi S. Bhavsar, Venkata Sai Gireesh Chamarthi, Samuel Oyeleye, Paul Samuel Kommula,
Department of Electrical and Computer Engineering, University of Florida, Gainesville, USA
janshisunbhavsar@ufl.edu, vchamarthi@ufl.edu, samuel.oyeleye@ufl.edu, paulsamuekommula@ufl.edu

***Abstract*—This report concentrates on the classification of handwritten digits using a popular deep learning algorithm known as convolution neural network (CNN) on a dataset of 3360 custom digit samples. Lately, with the expansion of Artificial Neural Network (ANN), profound learning has brought a sensational turn in the field of AI by making it more artificially intelligent. Deep learning (DL) is surprisingly utilized in tremendous scopes of fields due to its diverse range of applications such as surveillance, health, medicine, robotics, sports, mechanical technology, drones, and so on In Deep learning, Convolution Neural Network (CNN) is at the focal point of terrific advances that blends Artificial Neural Organization (ANN) and exceptional deep learning methodologies. For this exhibition assessment of CNN, we played out our trial utilizing the custom handwritten digits dataset by doing extensive pre-processing with normalization followed by usage of CNN to classify the digits and achieved 93.3% accuracy for a real-world digit classification dataset.**

***Keywords*— Convolutional Neural Network (CNN), morphological transformation, Data Augmentation, ReLu activation function, SoftMax, maxpooling2D.**

I. INTRODUCTION

A. Overview

Neural networks are computer simulations of how neurons in the human brain function. Numbers, shapes, pictures, and languages are easy for us, as humans, to recognize as meaningful pieces of knowledge. We recognize the lines, loops, and other elements, as well as their structure, to recognize a number. In real-word scenarios each individual has their own style of writing digits or alphabets. The problem is interesting and makes it much more difficult for the algorithm to identify the digits. Handwritten digit classification has a variety of uses and has been used in a variety of settings, including digital signatures on papers and mathematical equations. Feature engineering is another aspect which makes the neural networks robust and more efficient. Convolutional Neural Networks, Support Vector Machines, Random Forests, and K-Nearest Neighbors, Logistic Regression are some of the approaches used to classify handwritten digits. CNN was used in the development of this report. TensorFlow Keras was used to implement CNN. The benefit of using this classifier is it can provide better outcomes than other machine learning algorithms if hyperparameter tuning is done with a good amount of data. During the actual implementation of the CNN model By using filters, it preserves the spatial and temporal dependencies in an image. The convolution operation enables us to extract essential information from the data, such as edges. Pooling layer aids in the reduction of the size of convolved features, minimizing the computational power needed to process the data. We used Max Pooling for this project, which returns the maximum value from the kernel-covered part of the image. One problem with this model is we cannot work directly with categorical data(0-9 digits).so we have used the one hot encoding technique. This data augmentation to improve the model's accuracy by increasing the variety of data used to train it. Finally, the model's output predicts the image class it belongs to among the digits.

B. Literature Review

Convolutional Neural Network has come to become one of the most widely used Network algorithms for training models. The idea for CNN was inspired by the understanding of the visual cortex of animals. Similar to Artificial Neural Networks (ANN) in general, CNN is an example of a brain-inspired initiative combined with principles gotten from computer science and engineering to take advantage of the inner workings of the brain neurons to perform a plethora of different tasks that was difficult to do due to limitations of current models at the time. [2]. CNN was initially used majorly for object recognition tasks, now it is being applied in other areas such as object tracking, text detection, pose estimation, recognition, scene labelling etc. Generally, CNN also shares some of the characteristics of the Multilayer Perceptron Algorithm since it has one input layer, one output layer and at least one hidden layer.

To use the CNN, one has to input the dataset to be worked on. Each of the dataset will be in the form of an image that will then be pre-processed to make it compatible with the CNN algorithm used. Pre-processing of data typically involves various modifications on the data, some of which include: normalizing the data, resizing the input dataset, re-shaping the dataset, applying filters (if necessary) and performing one-hot encoding in general. After inputting the data and preprocessing it, the next thing to do is extract some features from the input data that would

be needed for defining the model. To perform feature extraction of input data is done using three layers namely: Convolutional layers, Activation unit/Function, and the Pooling layers. In a convolutional layer, multiple filters are applied over an image to extract different features and at the same time, we are learning the filters applied (Some examples of filters that could be applied unto images include edge detection, unsharp masking – which is basically subtracting the mean filters, image smoothing,). After the different filters have been passed over the image, a feature map is formed for the individualized filter applied. All the feature maps generated for all filters are then passed through an activation function which helps in determining the presence of the specific feature within the image. After this process, depending on how well the CNN needs to be defined, we can increase the number of filtering layers which would lead to the creation of more feature maps which would help to create a deeper CNN. The third layer used for feature extraction is called Pooling layers. Pooling layers are used for selecting the optimum parameters on the feature maps and using these values as input in the subsequent layers. It is important to note that theoretically, we can do any operation using pooling layers, but practically, in order to find the outliers (i.e the feature locations), we would use max-pooling instead.

As explained earlier, CNN architecture utilizes a multi-phase process that comprise of data pre-processing in the first phase, utilization of convolution layers, and the use of Activation Functions in the second phase (For the project, the RELU-Rectified Linear Unit activation function was used), and for the third phase, we build a fully connected layer and use the SoftMax classifier which is needed for predicting a multinomial probability distribution (Applied specifically to this project, the SoftMax classifier will give a probability of each digit in any of the 10 classes). The specified digit is then classified into the class that has the highest probability of representing it. A helpful way of visualizing the classification results of the dataset, is with the implementation of a confusion matrix which would have an axis for the predicted label and the true label. The use of hyper-parameters in our model can also affect the accuracy of the CNN architecture. For this project, using 2 pooling layers, 1 dropout layer, 4 convolutional layers, about 150 epochs, a batch size of 64 will increase the accuracy of our model to be about 93.3%. Other areas where we can apply digit classification (As done for this project) include: autonomous cars, handwriting recognition, License plate readers for security cameras and so much more.

II. IMPLEMENTATION

To recognize and successfully classify the custom made handwritten dataset, a 10 layer CNN was designed with 1 Input layer, 8 Hidden layers and 1 Output layer.

The below mentioned steps were followed for implementation:

1. Preprocessing of the input images.
2. Splitting the dataset into training and testing, the testing data is then used to evaluate the model performance.
3. Normalizing the data. It generally speeds up learning and leads to faster convergence.
4. Performing Data Augmentation on the images which will artificially expand the size of the training dataset by creating modified versions of the images in the dataset. The arguments that were provided for data augmentation were rotation range, width shift , height shift, shear range and zoom range.
5. A Convolutional Neural Network Model was created to be used as a classifier. The sequential model was used to create a linear stack of convolutional, Max Polling, dropout and Dense layers. ReLU and Softmax functions were used as activation functions.
6. The final training accuracy and validation accuracy were calculated after 150 epochs. The learning curves i.e., accuracy and loss curve were plotted. The confusion matrix was also plotted as an evaluation metric to check the performance of the model.

For identifying the digits which are not present in the training dataset, the SoftMax function was used for classification at the output layer. This gives output values in the form of probabilities. A threshold value of 0.09 was set as a baseline probability for the correct class classification. Whenever we have a new test point or sample, we can infer that it belongs to an unknown class (‘-1’) if the maximum probability is below the specified threshold.

The dataset consists of 4200 images of dimensions 300 x 300, out of which 3360 (80% of the dataset) has been used for training the model and 840 images were kept aside for testing on the trained model. Further these 3360 images have been split into Training set and Validation set (2688 and 672 respectively). Preprocessing is done on the dataset by reshaping and resizing the images to bring them to a format that can be fed to the CNN network. Further Morphological Transformation of Closing Operation from the CV2 module was done on the dataset. As the images in our dataset were highly disruptive and had noise, this morphological transformation was chosen. Closing Operation gets rid of noise in the foreground object (the digits) and small black points on the object.

The dimensions of the images in the dataset were resized to 30 x 30 x 1(greyscale images). This was done to reduce the computational cost of the model. One hot encoding is used to transform the vector of class integers into a binary matrix.

The training data set was trained with a model that consisted of 10 layers. The first layer of the model was a Convolution layer which made use of 32 filters with a kernel size of (3,3) using a ReLU activation function. The next hidden layer was another Convolution layer of 3 x 3 kernel size with 32 filters. It was also implemented using the ReLU Activation function. The feature map was then passed through a Max-pooling layer. The Max-

pooling layer down-samples the input, reducing the number of parameters to learn, thereby preventing overfitting. A pool size of 2x2 was used. The subsequent layers were two Convolution layers with 64 filters with kernel size of 3 x 3 and ReLU as the activation function. Again, a Max-pooling layer of pool-size 2x2 was added. It was then passed through a flatten layer, which is used to convert the 2-D feature map to a 1-D feature map. The feature map was then passed through a dense layer with 512 neurons after which it was sent to a dropout layer for regularization. The model was then compressed to a fully connected output layer of ten neurons with a SOFTMAX activation function, which determines the 0-9 digits.

III. EXPERIMENT

Our model is constructed using the purely custom written digits (0-9) from approximately 60 different writers. The dataset contains 3360 samples each of 300x300 sizes and the majority of the samples have various sorts of noise. These images were not even close to the MNIST digit images. So, this model will be best work for the real-world handwritten digits. This whole experimentation is carried out on a Google colab with a python 3.8 Google compute engine backend which is an open-source platform. We have performed a lot of preprocessing to make the images look like a raster image. An adaptive learning rate optimization algorithm commonly known as Adam optimizer is used with an initial learning rate of 0.001. This is a slightly slower starting rate which may take longer time to converge to minimal loss but reaches the peak accuracy.

Deep learning networks as a rule requires a huge dataset [1]. Our original dataset has very less sample size, so data augmentation is used to modify images thereby increasing the sample size. We have tweaked various parameters of augmentation to yield best accuracy possible.

Rotation range	Width shift	Height shift	Kernel size	Batch size	Accuracy
10	0.2	0.2	(3,3)	32	90.7%
10	0.2	0.2	(2,2)	32	84.7%
5	0.2	0.2	(2,2)	64	83.5%
15	0.2	0.2	(2,2)	16	86.2%
90	0	0.4	(5,5)	32	77.3%
10	0.1	0.1	(3,3)	32	87%
10	0.1	0.1	(3,3)	128	91.9%
8	0.08	0.08	(3,3)	64	93.3%

Table 1. Fine-tuning of various parameters in the model

Table 1 depicts the tuning of various parameters like width shift, height shift, rotation range, with varying batch and kernel size of the CNN model. Various settings resulted in a spectrum of accuracy ranges. Batch size of 64 with kernel size of (3,3) resulted in best accuracy of 93.3%.

We used the stochastic gradient descent algorithm to train the network using a batch size of 64. The categorical cross entropy is utilized to determine the difference between the true and predicted data values. Since the learning rate affected how quickly a model is adapted to the problem, going for a small learning rate of 0.001 enabled the model to converge more quickly and effectively (with high accuracy) without seeing unnecessary spikes when plotting the learning curves, the only downside being that it would take longer for the results to converge. Attached below is the Loss curve and the Accuracy curve for the training data.

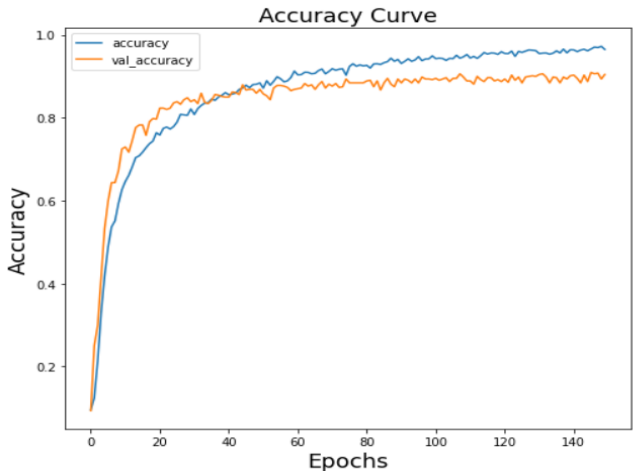

Fig.1 Accuracy vs Epochs curve

The accuracy curve shows the performance of our model over time, which signifies that the model is improving with experience. It can be observed that the curve grows at the beginning, but then plateaus, which signifies that the model is unable to learn anymore. For our dataset, most of the learning occurred after about 50 epochs, after which the learning rate decreased significantly to give us an accuracy of 93.3%

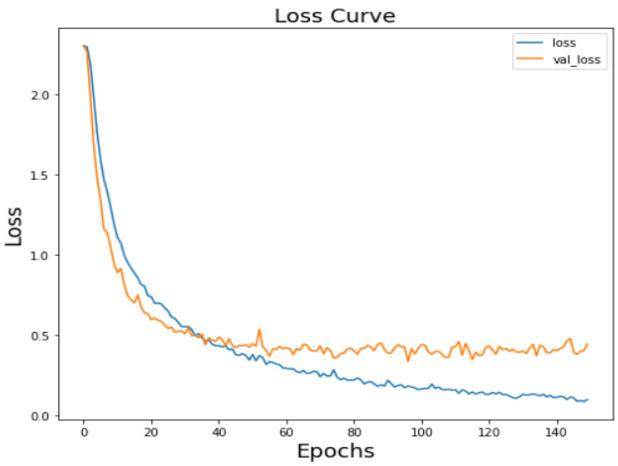

Fig.2 Loss vs Epochs curve

The training loss (shown in blue) shows how well the model was able to fit the training data, while the validation loss (orange color) show how well the model is able to fit the new data.

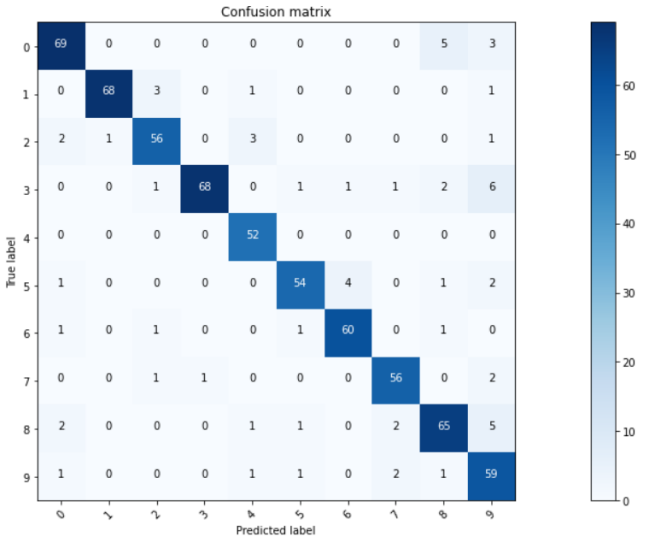

Fig. 3 Confusion matrix

Confusion matrix from figure 3 explains the correctly classified validation images along the diagonal with blue color blocks. The lighter blue color blocks show the misclassified images.

Some of the mis-classified digit samples are attached below. As we can see from the images, most of the images that were mis-classified have similar pattern/ shape to the correct labels. The misclassification could be due to insufficient cleaning of the sample data (i.e. there is still significant noise in the data that affected the accuracy of our model).

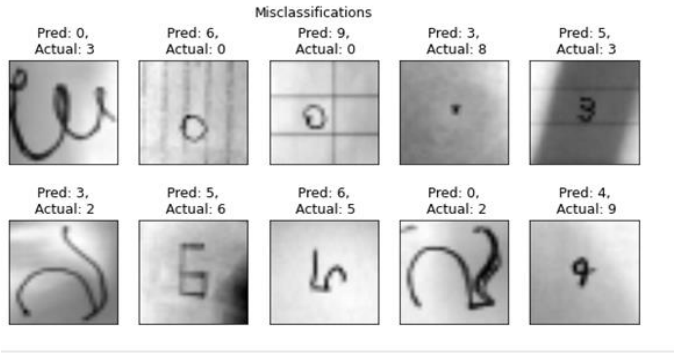

Fig.4 . samples of Mis-classified Dataset

We also experimented with a number of classifiers namely SVM with RBF kernel, KNN, Naives Bayes Classifier and Logistic Discrimination Classifier. All these classifiers performed poorly in comparison to CNN. All of them giving accuracy less than 80% . In addition, when comparing between the Multi-Layer Perceptron (MLP) and CNN we know that MLP does not account for spatial information while CNN algorithm is able to capture spatial information in a global or local view by either using large kernel matrices or smaller kernel matrices. The high accuracy seen in image results analyzed using CNN could be attributed to the fact that CNN preserves the spatial distances of the images.

Other improvement suggestions include applying network pruning techniques such as Boosting (a type of pruning technique that is going to weigh the samples based on their difficulty, and the difficulty is defined based on the mis-classified error), and Bagging (also known as Bootstrap Aggregating, where we train a classifier, and then combine these classifiers with the majority rule for us to find out the decision surface).

IV. CONCLUSION

In this paper, a method used to classify handwritten images using Convolutional Neural Networks was proposed. This classifier was chosen because CNN has been known to achieve high accuracy on image classification projects since it preserves the spatial distances of the images compared to base neural network architecture. To achieve a higher accuracy, the tradeoff between computational cost was considered. So, the experiment was carried out on a higher number of epochs and lower learning rate. A method to classify images, which are not present in the training data as unknown images that are ultimately assigned to label '-1' was also implemented. Accuracy of 99.1% on the training and an accuracy of 93.3% on the validation dataset was achieved.

V. REFERENCES

[1] C. Wigington, S. Stewart, B. Davis, B. Barrett, B. Price and S. Cohen, "Data Augmentation for Recognition of Handwritten Words and Lines Using a CNN-LSTM Network," 2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), 2017, pp. 639-645, doi: 10.1109/ICDAR.2017.110.

[2] A. M. Al-Saffar, H. Tao and M. A. Talab, "Review of deep convolutional neural network in image classification," 2017 International Conference on Radar, Antenna, Microwave, Electronics, and Telecommunications (ICRAMET), 2017, pp. 26-31, doi: 10.1109/ICRAMET.2017.8253139.

[3] Fathma Siddique, Shadman Sakib, and Md. Abu Bakr Siddique, "Recognition Of Handwritten Digit using Convolutional Neural Network in Python with Tensorflow and comparison of Performance for Various Hidden Layers", presented at the 5th Int. Conf. Advances in Electrical Eng., Dhaka, Bangladesh, Sept. 26-28, 2019.

[4] Nimisha Jain et al., "Handwritten digit recognition using convolutional neural network," IJIACS, vol. 6, issue 5, May 2017.