Image Classification of Handwritten Symbols Using Convolution Neural Network

Kaushik B. Jadav, Ishita S. Trivedi, and Vipul Adil
k.jadav@ufl.edu, itrivedi@ufl.edu, vipuladil@ufl.edu

***Abstract*—This report focuses on the categorization of handwritten symbols using the convolution neural network (CNN), a well-known and widely used deep learning algorithm, on a dataset of 23502 symbol samples of various sorts. Deep learning has recently made a dramatic turn in the realm of AI, thanks to the expansion of Artificial Neural Networks (ANN). Due to its wide range of applications, deep learning (DL) is surprisingly used in a wide range of fields, including health, robotics, NLP, recommendation systems, drones, sports, election predictions and many more. In Deep learning, Convolution Neural Network (CNN) is at the center of considerable advances that integrate Artificial Neural Network (ANN) and exceptional deep learning techniques. For this project on classification of handwritten symbols, we utilized our custom handwritten symbols dataset and tried various experimental designs to come up with the best parameters. We did extensive pre-processing of data with normalization following it by training the data on CNN and classifying the symbols. We managed to achieve 93.3% accuracy on the dataset in our experiment.**

Keywords— Convolutional Neural Network (CNN), ReLu activation function, SoftMax, maxpooling2D, Conv2d, Adam.

I. Introduction

A. Overview

Neural networks are computer simulations of neurons in the human brain and nervous system. We can easily distinguish meaningful bits of knowledge such as symbols, numbers, forms, pictures, and languages. Each person has their own manner of writing symbols in real life, making the algorithm's identification of the symbols both interesting and difficult. The classification of handwritten symbols has a wide range of applications and has been used in a variety of situations, including mathematical equations. Another factor that makes neural networks more durable and efficient is feature engineering. Some of the methods used to categorize handwritten symbols include Convolutional Neural Networks, Support Vector Machines, Random Forests, and Logistic Regression, as well as K-Nearest Neighbors. This report was created using CNN, and the programming language Python was used to create it. The justification for utilizing this classifier is that when hyperparameter tuning is done with a large amount of data, it produces better results than many machine learning methods. In the course of the implementation of the CNN model using filters preserves the temporal and spatial dependencies in an image. The convolution operation facilitates the extraction of significant information from the data, such as edges. The pooling layer assists in the reduction of the size of features that were convolved. This minimizes the computational power needed in processing the data. We used Max Pooling for this project, which selects the maximum value from the kernel-covered part of the image and returns the most prominent features. Finally, the model predicts the class that the image belongs to from all the symbols available.

B. Literature Review

In today's world, Convolutional Neural Network has become one of the most conventional Network algorithms for training huge data sets. Understanding of the visual cortex of a partially anesthetized cat when Dr. Hubel and Dr. Wiesel were working on an area of Sensory Processing sparked the creation of CNN. CNN is a fantastic example of combining computer science and engineering ideas to fully utilize the inner workings of brain neurons and do a variety of tasks that were previously difficult to perform owing to model restrictions. Initially, CNN was primarily utilized for object identification tasks, but it is currently being employed in a variety of different applications including object tracking, pose estimation, text detection, recognition, scene labeling, and so on. Because it contains one input layer, one output layer, and at minimum one hidden layer, some of the properties of the Multilayer Perceptron Algorithm can be found in CNN.

To utilize CNN, you must first enter the dataset on which the neural network will be trained. Each piece of data in the dataset will be represented by an image, which will be pre-processed to ensure compatibility with the algorithm. Data pre-processing usually entails a variety of changes to the data, such as resizing of the input dataset, reshaping the database, normalizing the data, and adding filters (if necessary). Following the data input and preprocessing, the following step is to extract specific features from the input data that will be used to define the model. Convolutional layers, Activation unit/Function layers, and Pooling layers are used to extract features from input data. Multiple filters are applied to an image to extract distinct features in a convolutional layer, and the model learns the filters at the same time. Edge detection, unsharp masking (essentially subtracting the mean filters), picture smoothing, and other filters are examples of filters that could be applied to photos.A feature map is created for each individual filter applied once the various filters have been applied. After that, an activation function is applied to all of the feature maps that have been generated for all of the filters. It aids in determining whether or not specific features are present in the image. Depending on how thoroughly the CNN needs to be specified, we can increase the number of filter layers. More feature maps will be generated, which will aid in the development of a more comprehensive CNN. In comparison to the previous levels, as the layers advance, more complicated features are retrieved. The Pooling layer is the third layer utilized for feature extraction. The ideal parameters on the feature maps are selected using pooling layers, and

these values are used as input for the subsequent layers. Polling layers cut down on the amount of computing resources needed to process the data. In theory, there are two types of pooling operations: average pooling and maximum pooling. The CNN design is divided into three phases: data pre-processing in the first, picture filtering with convolution layers and activation functions in the second, and generating a fully linked layer and employing the SoftMax classifier in the third. For predicting a multinomial probability distribution, the SoftMax classifier is required. Finally, the supplied symbol is assigned to the class with the greatest likelihood of representing it. The use of a confusion matrix to visualize the dataset's categorization results is a good idea. It will provide results for both the predicted and true labels in an easy-to-understand format. The accuracy of the CNN architecture can also be affected by the inclusion of hyper-parameters in our model. The goal of the project is to classify images successfully into their respective classes. In this project, we used 4 convolutional layers, ran the model for about 150 epochs with a batch size of 64 and managed to achieve an accuracy of about 93.3%.

II. Implementation

Several steps were followed for implementation namely:

1) Importing libraries and preprocessing of Images.

2) Splitting of Dataset into training and testing set to evaluate the model performance and create a model which works best on unseen/ blind dataset.

3) Normalizing the data to speed up the learning process and for faster convergence.

4) A convolution Neural Network model was created which is used as a classifier for classification of Images. To create a linear stack of convolutional, dropout, dense layers and Max polling, a sequential model was used. The activation functions used were ReLu and Softmax.

5) After various design experimentation, best parameters were choosen which yields higher accuracy and the loss and accuracy curves were plotted. The confusion matrix was also calculated as an evaluation metric for the model performance. It is attached at the end.

A model with four convolutional layers was used to learn the training data set. The model's first convolutional layer used a ReLU activation function and had a kernel size of (3,3). A Max-pooling layer was then applied to the feature map. The Max pooling layer reduces the amount of parameters to learn by downsampling the input, preventing overfitting. The pool was 2x2 in size. Another Convolution layer with a 3×3 kernel size was the following hidden layer. It was also done with the help of the ReLU Activation function. A Max-pooling layer with a pool size of 2x2 was added once more. Similarly, the table displays all of the convolutions layer's weights and biases. The next four layers are fully connected, with every neuron coupled to every connecting neuron. The model summary table displays the weights and bias of completely connected layers. With a SoftMax activation function, the model was compressed to a fully connected output layer of 25 neurons, which determines the symbols from classes $(0-24)$.

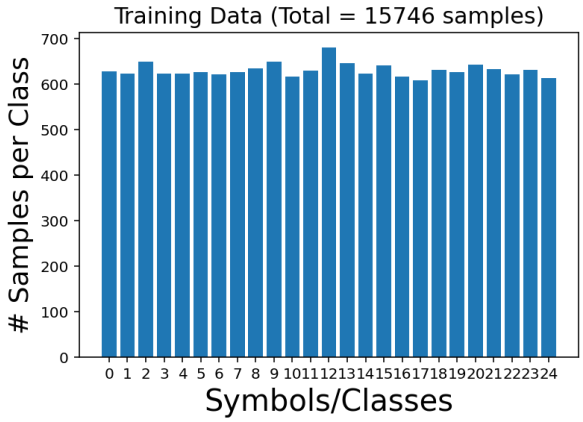

Fig. 1. Number of samples in Training set.

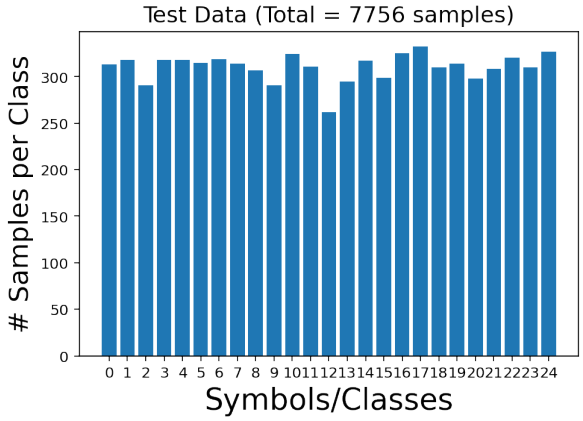

Fig. 2. Number of samples in Test set.

Model Summary		
Layers	Weights	Bias
Conv2d-1	(6, 1, 3, 3)	6
Conv2d-2	(16,6,3,3)	16
Conv2d-3	(32, 16, 3, 3)	32
Conv2d-4	(96, 32, 3, 3)	96
FC1	(84, 4704)	84
FC2	(84, 84)	84
FC3	(84, 84)	84
FC4	(25, 84)	25

III. Experimental Design

Our model is constructed using the purely custom written symbols from approximately 190 different individuals. The dataset contains 23502 samples, each of which is 150x150 pixels in size, with some samples having noise. The entire experiment is run on a Jupyter notebook running Python 3.8. With an initial learning rate of 0.001, an adaptive learning rate optimization technique known as Adam optimizer is applied. This is a somewhat slower initial rate that takes longer to converge to zero loss but achieves maximum accuracy. Deep learning networks, in general, necessitate a large dataset [1]. Our initial dataset has a large sample size, but the more examples there are, the higher the accuracy. During testing, we modified numerous factors to get the highest level of accuracy.

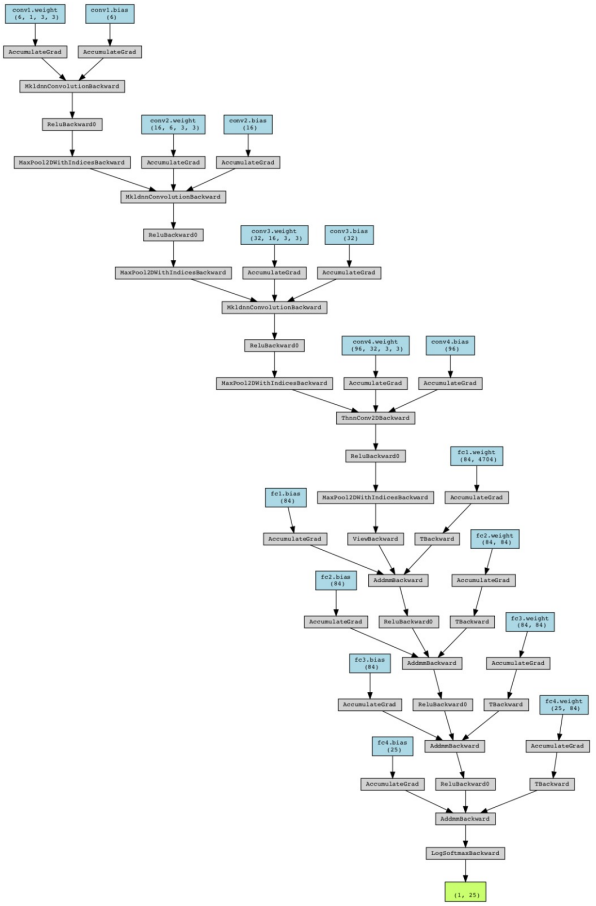

Fig. 3. Model Summary

The original dataset was divided into training and testing set initially to check the accuracy with different parameters. The dataset was divided with 15746 samples in training and 7756 samples in test set as shown in fig(1) and fig(2).

During experimentation, various parameters were tuned such as kernel size, batch size, learning rate, number of epochs of the CNN model to verify the accuracy. We also tried various optimization techniques for the CNN model like Stochastic gradient descent, Adam to see that Adam works best for us.

Tuning of Parameters			
Kernel Size	Batch Size	Learning Rate	Accuracy
(3,3)	32	0.001	90.3 %
(2,2)	32	0.001	87.7%
(2,2)	64	0.004	83.5%
(5,5)	32	0.001	77.8%
(5,5)	16	0.003	86.2%
(3,3)	128	0.004	92.9%
(3,3)	64	0.001	93.5%

Various different parameters settings resulted into different accuracies. The Adam optimizer with the kernel size of $(3, 3)$, batch size of 64 and learning rate of 0.001 yields the best accuracy of 93.5%. The learning rate affects how quickly the model adapts to the problem. Therefore, relatively a small learning rate helped our model converge quickly and with higher accuracy.

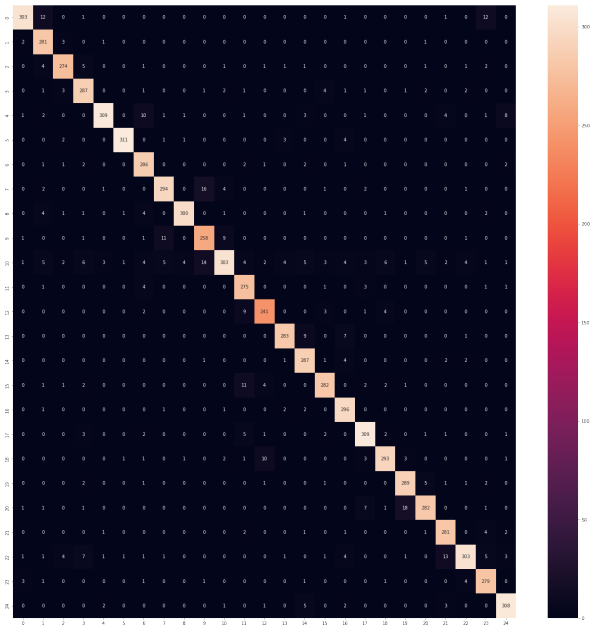

Fig. 4. Confusion matrix on Test set.

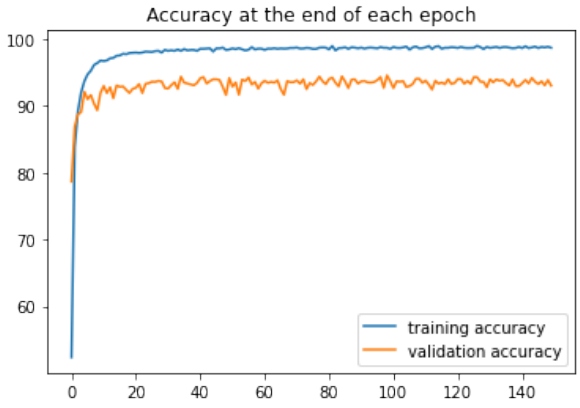

Fig. 5. Accuracy vs Epoch Curve.

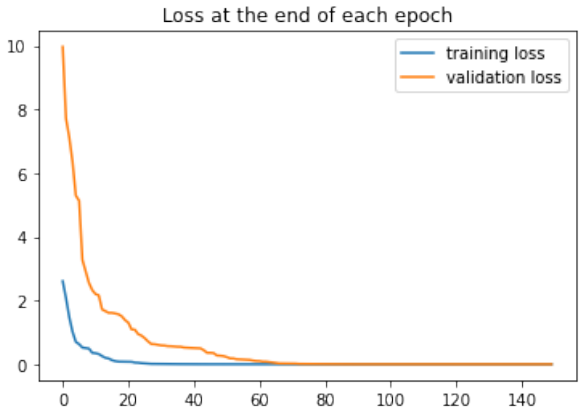

Fig. 6. Loss vs Epoch Curve.

IV. Conclusion

A method for classifying handwritten symbols using Convolutional Neural Networks was proposed in this report.This classifier was chosen because, when compared to base neural network design, CNN has shown to achieve superior accuracy on image classification projects since it preserves the spatial distances of the images. In order to achieve higher accuracy, we traded off time and used higher computation power. We ran our model for a greater number of epochs approximately 150 epochs and a slower learning rate of 0.001. The sole disadvantage of this experiment was that it took a significant amount of time to converge. We were able to achieve an accuracy of 99.1 percent on the training dataset and 93.3 percent on the validation dataset.

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

[5]https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

[6] https://medium.com/@gopalkalpande/biological-inspiration-of-convolutional-neural-network-cnn-9419668898ac