English Handwritten Characters Recognition Using Convolution Neural Network

Mashuk Khan
Student ID : 2018-1-60-162
Department of CSE
East West Uniiversity
Dhaka, Bangladesh
2018-1-60-162@std.ewubd.edu

Noureen Alam Meem
Student ID : 2018-1-60-053
Department of CSE
East West Uniiversity
Dhaka, Bangladesh
2018-1-60-053@std.ewubd.edu

Fariha Fairuz
Student ID : 2018-1-60-080
Department of CSE
East West Uniiversity
Dhaka, Bangladesh
2018-1-60-080@std.ewubd.edu

***Abstract*—The paper will represent the best approach to obtain more than 90% accuracy in the field of Handwritten Character Recognition (HCR). Although plenty of research has been done in this field, the maximum accuracy is yet to be achieved. In this paper, Convolutional Neural Network and Tensorflow were used to recognize offline handwritten characters. For assigning the probabilities, Softmax Regression method is used as the value we will get is between 0 and 1 which will sum up to 1. The primary goal of this paper is to establish a software that can provide a very high accuracy rate and takes minimal time to complete the procedure with lower space complexity.**

***Index Terms*—Convolutional Neural Network, Handwritten Character Recognition, EMNIST dataset, Softmax, ReLU.**

I. Introduction

In the past few years deep learning-based algorithms have been performing remarkably in the research in solving a wide range of machine learning problems. One of the most prominent and frequently used techniques is convolutional neural network (CNN). CNN is a kind of neural network that can retrieve attributes from multidimensional inputs, convert those into features for solving a problem within the range of computer vision, a field of AI that allows a computer to derive relevant information from digital images, videos or any other form of visual inputs [1]. In this project we employed CNN for recognising English handwritten characters. The subsequent objective is to simulate the human trait of reading, recognizing into computers. Handwritten character recognition is one of the most compelling research in the field of pattern recognition and image processing [2]. This evidently contributes to the field of automation. Character recognition is of two sorts, online and offline. Online character recognition requires the input data to be captured in real time using a special pen together with an electronic surface. As the pen is moved across the surface, the two- dimensional coordinates of points are manipulated as a function of time and are stored order-wise. [**?**]. The EMNIST dataset has been extensively used for these models. The EMNIST (Extended Modified National Institute of Standards and Technology) database is a sizable database of handwritten english letters which is comprehensively used for training image processing models. It is also used as a testing and training dataset in Machine Learning [1], [3].

II. EMNIST

The NIST (National Institute of Standards and Technology) Special Database 19 [11] contains handwritten characters and digits accumulated from over 500 contributors. The dataset is a collection of binary scans of the collected handwriting samples. There is a noteworthy number of research tested over NIST and MNIST (Modified National Institute of Standards and Technology) and the tests were subjected to 1% error rate. In 2017, Cohen et al. [3] established the Extended MNIST database (EMNIST). EMNIST incorporated handwritten digits and letters. The authors established that, “the dataset labeling can be called into question”, alleging EMNIST as an uncomplicated and straightforward standard. A sample of EMNIST database is shown in Fig. 1.

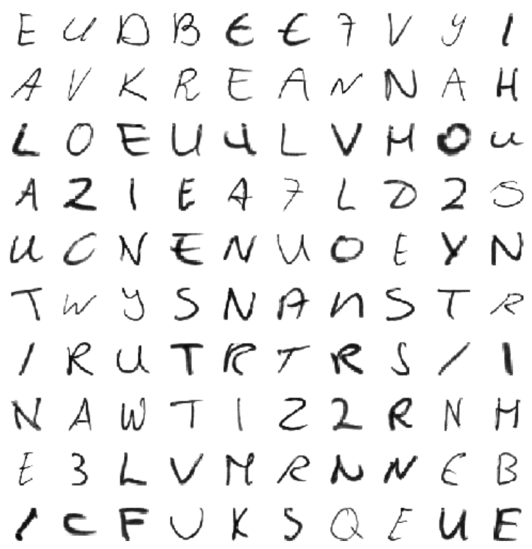


Fig. 1. EMNIST sample dataset

The source for compiling EMNIST database was NIST Special Database 19 (NIST SD 19) , containing NIST’s complete training materials for handwritten document and character recognition. This contains over 800,000 characters which have been manually checked and labelled from almost 3700 writers. It was accessible from the mid 1990s. But, it’s use remained very limited used due to the complications in obtaining and using it in modern computers. This was fixed in 2016 when the updated version of the database was published with a simpler format. The authors of EMNIST have intentionally made the database flexible with MNIST in terms of structure. Hence it can perform a similar processing. [1], [3].

III. The Proposed Recognition Algorithm

In this study, tests were performed on the EMNIST handwritten data set with CNN Algorithm using CSV file, filtered image files and binary format of EMNIST balanced dataset as inputs. The accuracy rates of the algorithms in all categories of inputs were compared. Our HCR system has these following steps (in order) : Image Acquisition, Preprocessing, Augmentation, One Hot Encoding, Splitting Dataset, Training and Testing, Prediction. [Figure2].

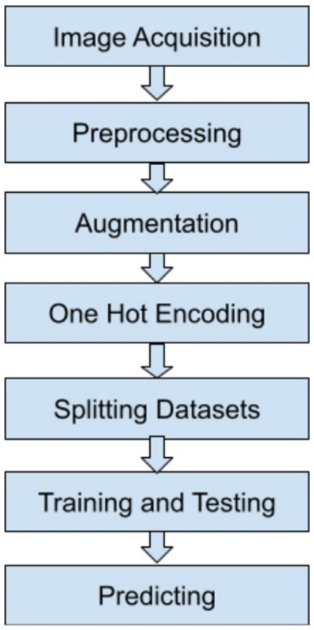


Fig. 2. HCR block diagram

A. Image Acquisition

1) CSV Input File: CSV input file was composed from images in EMNIST by_class unbalanced dataset. The image files were taken as an array of inputs to generate the CSV file as an output.

2) Image Input Files: In image acquisition, the recognition system generate a scanned image file as an input image. The image should have a specific format such as .jpg, .jpeg, .png, etc. Scanner, digital camera or any other digital input devices are used for image file acquisition depending on the user interface [4]. [Figure3].

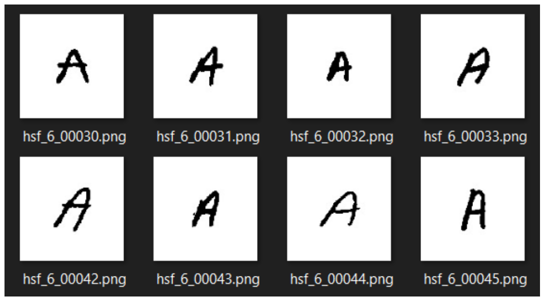


Fig. 3. Sample imput file

3) EMNIST Balanced Dataset in binary/idx Format: This variant on the dataset merges certain classes that look quite similar in respective uppercase and lowercase format, creating a 47-class classification task. The merged classes are for the letters C, I, J, K, L, M, O, P, S, U, V, W, X, Y and Z [5].

B. Preprocessing

All tests within the scope of this study were performed on the Visual Studio Code (v1.60.0), which is an integrated development environment used in programming and development. A Python library called Scikit Learn and Tensorflow was used in the study. Tensorflow was mainly used for the CNN model training and testing. It offers the opportunity to effectively implement many machine learning algorithms. It enables the rapid training and testing of machine learning algorithms with high level languages.

1) CSV Input File: Separate classes were generated for each characters. In the coding part, characters were labelled using string values. CSV file was reshaped into 28x28 pixels as two dimensional array. This two dimensional array was flattened and turned into single dimensional array of 784 indices.

2) Image Input Files: Input files were resized into 28x28 pixels. Initial depth of the files was 3 as they were considered as coloured image. The depth was converted to 1 to turn the coloured images into grayscale images.

3) EMNIST Balanced Dataset in binary/idx Format: Binary format files were altered by flipping and rotating to evolve them into error-free inputs.
We normalized the data dividing the pixel size by 255, the resulting number ranges from 0 to 1. This process is named as normalization. Normalization refers to re-scaling the original ranged real-valued numeric attributes within the 0 to 1 range. When the different features are on smaller scale, Machine Learning algorithms perform better and converge faster. Therefore, normalizing the data before training the Machine Learning model is very important. It makes the model training less sensitive to the scale of features.

C. Spliting Dataset

All the inputs were split into train sets and test sets. Each of the train set includes 80% of the total files. Test sets include 20% of the files.

D. Image Augmentation

Data augmentation is the technique of increasing the amount and diversity of data used for training a model. For reliable predictions, the ML algorithms often require a lot of training data, which is not always available. Therefore, the existing data is transformed in order to make a better generalized model. The most commonly used operations in data augmentation are-Scaling, Rotating, Zooming, Flipping, Cropping, Shearing, Changing the brightness level. While training a machine learning model, It acts as a regularizer and helps to prevent over-fitting. In our model, image augmentations were made such as enlarging, reducing the images rotating the images, adjusting height and width etc.

E. One Hot Encoding

The class values were passed as a parameter in one hot encoding function. One hot encoding is a process of converting categorical data variables into a form so that they can be provided to machine learning algorithms to do a better job in prediction. It is one of the most crucial parts of feature engineering for Machine Learning. Depending on the implementation process, Some Machine Learning algorithms can work directly with categorical data. However, most require any numeric value as input or output variables. With the help of one-hot, each categorical value is converted into a new categorical column and those columns are assigned with a binary value of 1 or 0. Each integer value is represented as a binary vector. The index value is marked with 1 and the rest of the values will be 0. By using one hot, a probability for the data can be easily determined.

F. Model

The model includes two activation functions : ReLU and Softmax functions and some other layers.

1) Rectified Linear Unit (ReLU) Function: ReLU is the mostly used activation function in Neural Networks. It is simpler and more cost effective than tanh function and sigmoid function as it contains simple calculations. It has comparatively less shortcoming as an activation function.

2) Maxpooling: Max Pooling is a convolution process that calculates the maximum value in each patch of each feature map. That means, it informs the Convolutional Neural Network that we will only carry forward the largest information available amplitude wise.

3) Softmax Function: The softmax function is used as an activation function to the very last layer of a neural network. The input values can be positive, negative, zero, or greater than one. However, the softmax function transforms them into values between 0 and 1 so that they can be interpreted as probabilities.

4) Convolution Layer: Convolutional layers are the layers where filters are applied to the original image, or to other feature maps in a deep CNN. This is where most of the user-specified parameters are in the network.

All of these repeat themselves in a pattern as illustrated in Fig. 4

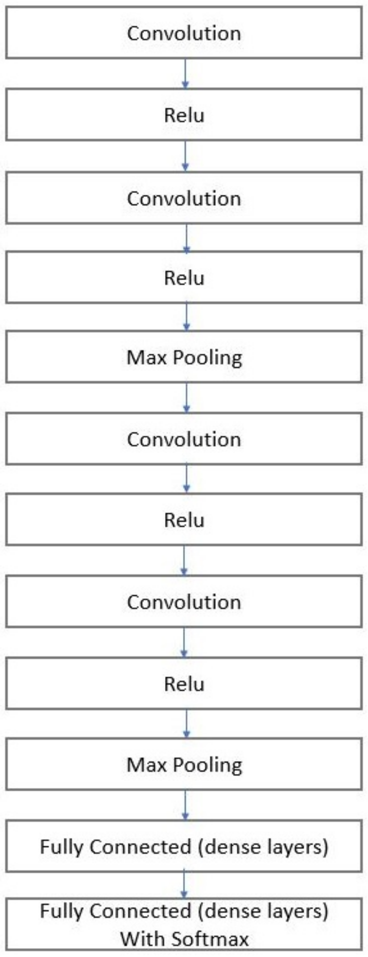


Fig. 4. CNN model layers

G. Accuracy

1) CSV Input File: CSV file had unbalanced number of data. [Figure5].

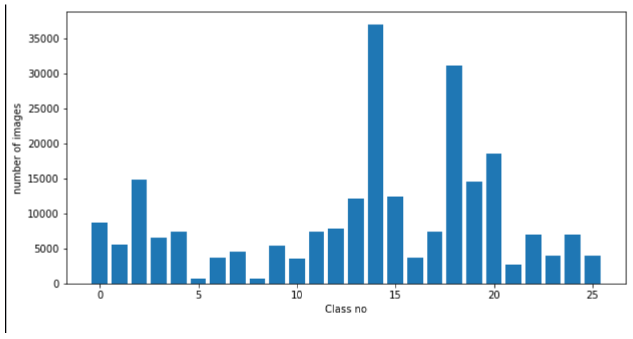
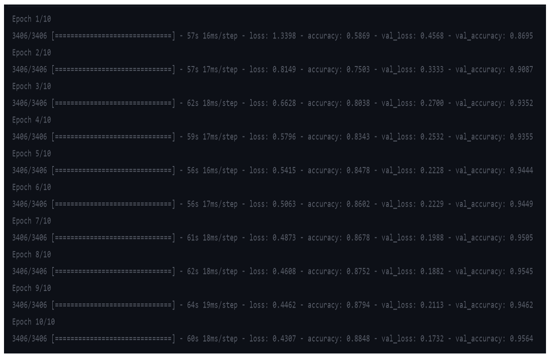


Fig. 5.

Training accuracy was 88.48% and test accuracy was 95.64% [Figure6]. But it was unable to meet the required prediction accuracy.That is why we did not use image files for predicting.



```
Epoch 1/10
3406/3406 [==============================] - 57s 16ms/step - loss: 1.3398 - accuracy: 0.5869 - val_loss: 0.4568 - val_accuracy: 0.8695
Epoch 2/10
3406/3406 [==============================] - 57s 17ms/step - loss: 0.8149 - accuracy: 0.7503 - val_loss: 0.3333 - val_accuracy: 0.9087
Epoch 3/10
3406/3406 [==============================] - 62s 18ms/step - loss: 0.6628 - accuracy: 0.8038 - val_loss: 0.2700 - val_accuracy: 0.9352
Epoch 4/10
3406/3406 [==============================] - 59s 17ms/step - loss: 0.5796 - accuracy: 0.8343 - val_loss: 0.2532 - val_accuracy: 0.9355
Epoch 5/10
3406/3406 [==============================] - 56s 16ms/step - loss: 0.5415 - accuracy: 0.8478 - val_loss: 0.2228 - val_accuracy: 0.9444
Epoch 6/10
3406/3406 [==============================] - 56s 17ms/step - loss: 0.5063 - accuracy: 0.8602 - val_loss: 0.2229 - val_accuracy: 0.9449
Epoch 7/10
3406/3406 [==============================] - 61s 18ms/step - loss: 0.4873 - accuracy: 0.8678 - val_loss: 0.1988 - val_accuracy: 0.9505
Epoch 8/10
3406/3406 [==============================] - 62s 18ms/step - loss: 0.4608 - accuracy: 0.8752 - val_loss: 0.1882 - val_accuracy: 0.9545
Epoch 9/10
3406/3406 [==============================] - 64s 19ms/step - loss: 0.4462 - accuracy: 0.8794 - val_loss: 0.2113 - val_accuracy: 0.9462
Epoch 10/10
3406/3406 [==============================] - 60s 18ms/step - loss: 0.4307 - accuracy: 0.8848 - val_loss: 0.1732 - val_accuracy: 0.9564
```

Fig. 6. Training and test accuracy for CSV file

2) Image Input Files: Training dataset for image files were balanced i.e., the classes had 2000 images each. [Figure7].

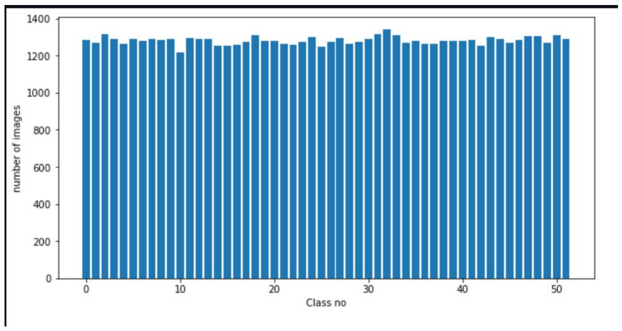
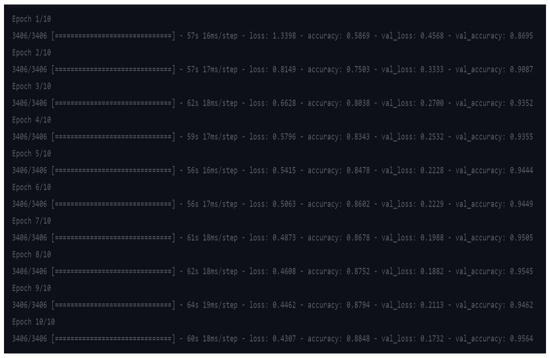


Fig. 7.

But the training accuracy was only 1.9%. Test accuracy was 2%. [Figure8].



```
Epoch 1/10
3406/3406 [==============================] - 57s 16ms/step - loss: 1.3398 - accuracy: 0.5869 - val_loss: 0.4568 - val_accuracy: 0.8695
Epoch 2/10
3406/3406 [==============================] - 57s 17ms/step - loss: 0.8149 - accuracy: 0.7503 - val_loss: 0.3333 - val_accuracy: 0.9087
Epoch 3/10
3406/3406 [==============================] - 62s 18ms/step - loss: 0.6628 - accuracy: 0.8038 - val_loss: 0.2700 - val_accuracy: 0.9352
Epoch 4/10
3406/3406 [==============================] - 59s 17ms/step - loss: 0.5796 - accuracy: 0.8343 - val_loss: 0.2532 - val_accuracy: 0.9355
Epoch 5/10
3406/3406 [==============================] - 56s 16ms/step - loss: 0.5415 - accuracy: 0.8478 - val_loss: 0.2228 - val_accuracy: 0.9444
Epoch 6/10
3406/3406 [==============================] - 56s 17ms/step - loss: 0.5063 - accuracy: 0.8602 - val_loss: 0.2229 - val_accuracy: 0.9449
Epoch 7/10
3406/3406 [==============================] - 61s 18ms/step - loss: 0.4873 - accuracy: 0.8678 - val_loss: 0.1988 - val_accuracy: 0.9505
Epoch 8/10
3406/3406 [==============================] - 62s 18ms/step - loss: 0.4608 - accuracy: 0.8752 - val_loss: 0.1882 - val_accuracy: 0.9545
Epoch 9/10
3406/3406 [==============================] - 64s 19ms/step - loss: 0.4462 - accuracy: 0.8794 - val_loss: 0.2113 - val_accuracy: 0.9462
Epoch 10/10
3406/3406 [==============================] - 60s 18ms/step - loss: 0.4307 - accuracy: 0.8848 - val_loss: 0.1732 - val_accuracy: 0.9564
```

Fig. 8. Training and test accuracy for Image files

That is why we did not use image files for predicting.

3) EMNIST Balanced Dataset in binary/idx Format: Training dataset for image files were balanced [Figure9].
But the training accuracy was 51.44%. Test accuracy was 70.38%. [Figure10].
We only used EMNIST Balanced Dataset in binary/idx Format for prediction as we obtained maximum percentage of accuracy which also met the requirement.

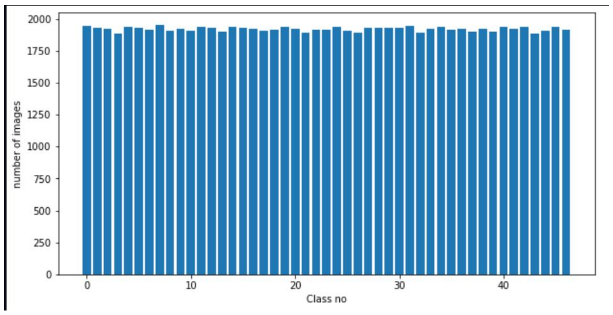
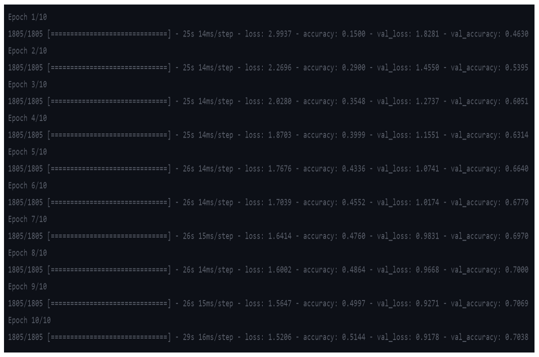


Fig. 9.



```
Epoch 1/10
1805/1805 [==============================] - 25s 14ms/step - loss: 2.9937 - accuracy: 0.1500 - val_loss: 1.8281 - val_accuracy: 0.4630
Epoch 2/10
1805/1805 [==============================] - 25s 14ms/step - loss: 2.2696 - accuracy: 0.2900 - val_loss: 1.4550 - val_accuracy: 0.5395
Epoch 3/10
1805/1805 [==============================] - 25s 14ms/step - loss: 2.0280 - accuracy: 0.3548 - val_loss: 1.2737 - val_accuracy: 0.6051
Epoch 4/10
1805/1805 [==============================] - 25s 14ms/step - loss: 1.8703 - accuracy: 0.3999 - val_loss: 1.1551 - val_accuracy: 0.6314
Epoch 5/10
1805/1805 [==============================] - 26s 14ms/step - loss: 1.7676 - accuracy: 0.4336 - val_loss: 1.0741 - val_accuracy: 0.6640
Epoch 6/10
1805/1805 [==============================] - 26s 14ms/step - loss: 1.7039 - accuracy: 0.4552 - val_loss: 1.0174 - val_accuracy: 0.6770
Epoch 7/10
1805/1805 [==============================] - 26s 15ms/step - loss: 1.6414 - accuracy: 0.4760 - val_loss: 0.9831 - val_accuracy: 0.6970
Epoch 8/10
1805/1805 [==============================] - 26s 14ms/step - loss: 1.6002 - accuracy: 0.4864 - val_loss: 0.9668 - val_accuracy: 0.7000
Epoch 9/10
1805/1805 [==============================] - 26s 15ms/step - loss: 1.5647 - accuracy: 0.4997 - val_loss: 0.9271 - val_accuracy: 0.7069
Epoch 10/10
1805/1805 [==============================] - 29s 16ms/step - loss: 1.5206 - accuracy: 0.5144 - val_loss: 0.9178 - val_accuracy: 0.7038
```

Fig. 10. Training and test accuracy for Image files

H. Comparison Among Dataset Accuracies

The reason why we obtained different accuracies beacuse, (1)The number of letters in each classes were unbalanced in all of the datasets and (2) Image file didn't contain adequate amount of files for each characters and there were many corrupted files in those.

IV. Conclusion

The handwritten character recognition system described in this project will find promising applications in handwritten name recognition, digitizing documents, document reading, conversion of any handwritten document into structural text form. These set of algorithms can be easily modified and upgraded according to future requirements.

References

[1] A. Baldominos, Y. Saez, and P. Isasi, "A survey of handwritten character recognition with MNIST and EMNIST," *Applied Sciences (Switzerland)*, vol. 9, no. 15, 2019.
[2] Y. Lu, "Handwritten capital letter recognition based on OpenCV," *MATEC Web of Conferences*, vol. 277, p. 02030, 2019.
[3] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, "EMNIST: Extending MNIST to handwritten letters," *Proceedings of the International Joint Conference on Neural Networks*, vol. 2017-May, pp. 2921–2926, 2017.
[4] M. K. Sahu and D. N. K. Dewangan, "A Survey on Handwritten Character Recognition," *Iarjset*, vol. 4, no. 1, pp. 89–91, 2017.
[5] N. Gutierrez, "Unsupervised Learning to Distinguish Letters and Numbers Using Spike-Timing Dependent Plasticity," pp. 1–9.