handwriting recognition

Convolutional Neural Networks

AMIT KUMAR

amitkumar21@gmail.com

**Acknowledgment**

I would like to express my special thanks to Mr. Robert Guenther, who has been very supportive in providing me with the base Python Code for this project, along with his valuable time to undertake a project report review and provide his valuable insights. I am thankful to him.

Table of Contents

[Abstract 1](#_Toc102770892)

[Introduction 2](#_Toc102770893)

[Literature Review 4](#_Toc102770894)

[Data 7](#_Toc102770895)

[Data selection 7](#_Toc102770896)

[Pre-processing 8](#_Toc102770897)

[Findings 9](#_Toc102770898)

[Methods 13](#_Toc102770899)

[Results 18](#_Toc102770900)

[Analysis and Interpretation 18](#_Toc102770901)

[Conclusions 18](#_Toc102770902)

[Directions for Future Work 18](#_Toc102770903)

[References 19](#_Toc102770904)

**Table of Figures**

[Figure 1: Distribution of Numbers training dataset 10](#_Toc102770935)

[Figure 2: Distribution of Numbers testing dataset 10](#_Toc102770936)

[Figure 3: Distribution of Shapes dataset 11](#_Toc102770937)

[Figure 4: Distribution of lines 11](#_Toc102770938)

[Figure 5: Distribution of Training Letters 12](#_Toc102770939)

[Figure 6: Distribution of Test Letters 12](#_Toc102770940)

[Figure 7: Outline of CNN 14](#_Toc102770941)

# Abstract

Enabling a computer to develop an ability to classify handwritten digits, characters, shapes, and lines has been a popular dataset amongst students and researchers. The MNIST dataset is an old dataset and has become non-challenging, which led to the release of EMNIST. Most of the research conducted on the MNIST dataset concluded that Convolutional Neural Networks are the go-to architecture to achieve classification accuracy to near perfection (~99.5%). Research has also ascertained that pre-processing and data augmentation play an important role besides the model architecture in improving the accuracy of a deep learning model.

This study combines various datasets: MNIST dataset for digits, EMNIST for characters, geometric shapes, and line orientation data and develops a classification model using CNN to achieve an ambitious 95% accuracy. This study aims to achieve this by diving deep into the fundamentals of CNN architecture and by performing hyperparameter tuning to arrive at an optimal CNN architecture while considering the computational time requirements.

Keywords: Handwritten, Classification, Pre-processing, Convolutional Neural Network, Hyperparameter Tuning

# Introduction

The aspiration of humans to develop machines can be dated back to their evolution. This inspiration has led us to develop an intelligent thinking machine, which evolved for the first time in the 1950s, leading to the birth of Artificial Intelligence. Artificial Intelligence has gone through several phases but has brought about an era where we now have been able to develop self-driving cars. While as trivial as it may sound to produce such intelligent machines in today’s times, it has been a challenging and complex task to develop these “thinking machines.” Besides the mechanical driving ability, the primary objective of these self-driving cars is to have the ability to detect and classify objects around and develop the decision-making ability to respond to the complex environment, which is dynamic and constantly evolving. Developing such complex machines requires a robust fundamental understanding of first developing a capability to detect and classify various objects.

Detection and identification of the orientation of a line ability to distinguish between various geometric shapes, digits, and alphabets (uppercase or lowercase) is a trivial task for a human being. The ability to correctly classify these entities is an elementary objective introduced to a child as the first step of their structured learning. Once a child has learned to identify these objects, they can promptly and instinctively identify them regardless of their size, orientation, color, or even if they are partially occluded.

As the access to digital assets, i.e., laptops, tablets, and mobile phones, has become a common phenomenon to make notes or prepare reports, traditional pen and paper are more readily available and easy to use. There is still a large quantity of existing handwritten text, and much more being created. However, as we advance towards digitization, there is a vast need to convert this handwritten text into meaningful digital information for a computer. Many variances in people's handwriting complicate this effort of digitization.

This paper attempts to build a model to “educate” a computer to accomplish a similar task using EMNIST datasets and a dataset containing geometric shapes and lines. It is quite a challenging and complex task to enable a computer to detect and classify these essential elements with similar absolute accuracy. This paper attempts to imbibe the learnings from similar studies and develop a robust model using Convolutional Neural Networks (hereafter CNN) architecture.

# Literature Review

MNIST and EMNIST are popular datasets for learning and research for image classification problems in computer vision and neural networks. Due to the uncomplicated nature of the dataset, Deep learning models have achieved incredibly high accuracy for MNIST datasets. This section reviews some of the recommendations and findings from the relevant works.

Pre-processing of images plays an integral part in the accuracy of the model outcome (Baldominos, Saez, and Isasi 2019; Yahya, Tan, and Hu 2021). Before using the image dataset, it is crucial to understand converting the raw images to a lower resolution. It has been noted that there is an improvement in the error rate when the digits are centered by a bounding box with some padding rather than a center of mass to handle the variations in shape and size of characters more efficiently (Cohen et al. 2017). This approach also preserves the aspect ratio and prevents the digits and characters from touching the borders. Another aspect of pre-processing is the choice between normalized or binary pre-processing. In their study, Shamsuddin, Abdul-Rahman, and Mohamed 2018 conclude that the normalized dataset preserves more detailed information than the binary pre-processed image, and the representation of the images is also smoother. The study compared the two differently pre-processed images on the same model and observed a drastic difference in outcomes. The normalized dataset achieved 99.4% accuracy, while the same model achieved an accuracy of 90.1% for the binary pre-processed images. In their study, Yahya, Tan, and Hu 2021 recommend analyzing and correcting statistical errors and noise within the dataset. It is crucial to pre-process data to ensure the dataset is free from any redundant or irrelevant variables to the target variables, as it can lead to misleading results/classifications. The study also stresses the need to undertake the pre-processing as lack of this may negatively affect model performance accuracy. One of the recommendations is to compare the means and distribution of the numeric features of both train and test data. It is also essential to look at the distribution and mean values of the labels within the training set since an unbalanced training set can contribute to reducing the over-representation of the labels in the validation set.

Data augmentation is another aspect of enhancing the model performance when combined with a Convolutional Neural Network (Baldominos, Saez, and Isasi, 2019; Yahya, Tan, and Hu 2021). Data augmentation techniques such as rotation, horizontal shifting, and adjustments to zoom range have shown improvements in the model accuracy even if minimal (99.50% without augmentation vs. 99.98% with augmentation). Yahya, Tan, and Hu 2021 make an interesting observation in their study by adding an additive white Gaussian noise with sigma = 0.5 to evaluate the performance of their proposed algorithm on a noisy MNIST dataset. The study applied this noise to both training and test data with random noise and was still able to achieve a 99.40% accuracy.

Convolutional Neural Network (hereafter CNN) has been a method of choice for image classification even with minimal pre-processing when compared (Lecun et al. 1998). Feature extraction is a complicated task, and even with domain experts, human designers will be unable to capture all the relevant information in the input. The study showed that applying gradient-based learning to CNN allows substantial improvement in learning the appropriate features. Hassan, Abraham, and Ramanujan 2021 have outlined that CNN has been able to alleviate the shortcomings of the traditional machine learning approaches like support vector machines (SVM), random forests (RF), k nearest neighbors (kNN), and decision trees (DT). Cohen et al. 2017 study attempt to use an Extreme Learning Machine (ELM hereafter) neural network to perform classification using Linear and OPIUM-based classifiers on the EMNIST dataset. ELM can be used as a method of regression or classification for large and complex datasets. However, the study could only achieve a maximum mean accuracy of 78.02% with 10,000 hidden layer neuro, although the ELM NN model achieved higher accuracy on the MNIST dataset. The study concludes with an outcome that EMNIST is a more complicated dataset as compared to the MNIST dataset. Yahya, Tan, and Hu 2021 study recommend first calculating before using an Effective Receptive Field (ERF) that would allow selecting a typical filter size, which leads to improving the classification accuracy of the CNN. The study recommends starting the selection of the proposed filter with the calculated size to convolve the input image and obtain the feature map. The study then recommends repeating this process with the next layer, which will then provide a deeper feature map to a point where an output image with an effective receptive field with a size of 22 x 22 is achieved. Max pooling with a 2 x 2 filter and a stride = 2 has been used in the paper to increase the size of the receptive field. The model achieves a 99.98% recognition accuracy on the MNIST dataset.

# Data

## Data selection

The data for this study comprises of following datasets:

1. Numbers: Handwritten digits (0 – 9) – Obtained from MNIST dataset
2. Shapes: Squares, Circles, and Triangles – Obtained from Kaggle dataset
3. Lines data: Horizontal, Vertical, Diagonal – Generated using code
4. Letters: Uppercase and Lowercase – Obtained from EMNIST dataset
5. **Numbers**: The MNIST dataset is constructed from the NIST Special Database 3 and Special Database 1. This database has binary images of handwritten digits. NIST's original black and white (bi-level) images have been size normalized to fit in a 20x20 pixel box while preserving their aspect ratio. The resulting images contain grey levels due to the anti-aliasing technique used by the normalization algorithm. The images have been centered in a 28x28 image by computing the center of mass of the pixels. Then the image is translated to a position at this point at the center of the 28x28 field.
6. **Shapes**: The shapes dataset is sourced from the Kaggle dataset. The dataset has been equally divided into 100 images each of squares, circles, and triangles. Dataset source states that each image is in \*.PNG format with an image size of 28 pixels x 28 pixels. The dataset is not split between training and test data. Moreover, due to the small number of images, the data needs to be augmented. This has been done in a two (2) step process: First, the shape has been rotated by an increment of 90 degrees (0, 90, 180, 270 degrees) to enhance the dataset. The output image is then placed in the top left corner and then is shifted down using a variable until it reaches the lower bound pixel 27. The image is then shifted to the top and then shifted right, and is again moved down in-line with the process in the previous step. The source data does not have labels assigned to them but are segregated by shape in three (3) different folders. The image is assigned the following labels: Square = 10, Circle = 11, Triangle = 12 for original data and at each data augmentation step. Each pixel has a value between 0 to 255 and shall be normalized from 0 to 1.
7. **Lines**: The line images are created using a generator and are characterized based on their slopes. Lines can be classified into three categories based on their orientation: Horizontal, Vertical, and Diagonal. The program will generate 7,500 lines for each of these categories and have been assigned the following labels: Horizontal = 30, Vertical = 40, Diagonal = 41. The images produced have a shape of 28 x 28 pixels. The other line generation characteristics are based on line thickness and a minimum length.
8. **Letters**: The letter examples are obtained from the EMNIST dataset, another dataset derived from the NIST Special Database 19, containing all 26 English alphabets. This dataset directly matches the image specifications and dataset structure of the MNIST dataset. However, the dataset is much more challenging regarding the interclass similarities and the larger number of output classes. The EMNIST data has both uppercase and lowercase alphabets and are available in an image size of 28 x 28 pixels.

## Pre-processing

A computer-based image in its basic form is a grid of numbers and each square in that grid is referred to as a pixel. Each pixel is assigned a number between 0 (white) to 255 (black) on a grayscale palette. These pixel values undergo a process of normalization to convert them to a range, which is between 0 (white) and 1 (black). Once we have normalized these numbers, we feed them into our model. There are several methods for pre-processing.

Normalization: The image data for this study has 42 categories with dimension size of 28 x 28 pixels in grayscale format. The grayscale images allow more detailed information to be preserved. The representative values of the pixels in an image contain an array of values from 0 to 255. The network activation can be slightly unstable since there will be more variation elements in the network input range. Therefore, to prevent a high activation of the learning models, the grayscale values are normalized using a min-max function with values ranging between zero (0) to one (1).

Data Augmentation is a process of artificially increasing the size of the training set by generating many realistic variants of each training instance. This will help in reducing overfitting, which is a regularization technique. Data augmentation also allows the model to be more tolerant to variations in the position of an object in an image.

## Findings

After importing each dataset, the first step is to perform an Exploratory Data Analysis on each input dataset to understand its basic characteristics. A category wise count of each class is presented below:

Digits: The digits dataset is split into training and testing datasets. The dataset has 60,000 training images and 10,000 test images. A bar graph showing a split of each of these categories is presented *in Figure 1*. It is observed that the training dataset does not have a balanced distribution. The count of number two (2) images is the highest, with number five (5) having the least number of images.

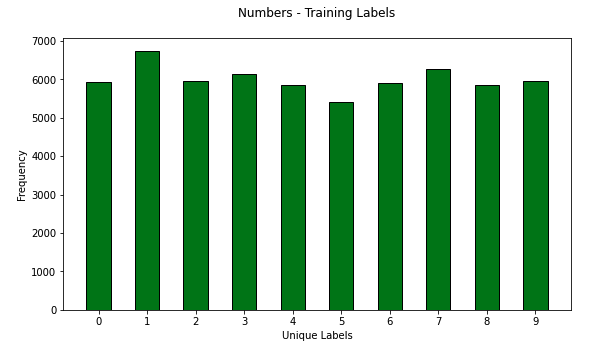


Figure 1: Distribution of Numbers training dataset

The distribution of the test dataset in *Figure 2* is similar to the training dataset, with the highest number of images for the number two (2) and the lowest for the number five (5).

Chart

Description automatically generated

Figure 2: Distribution of Numbers testing dataset

Shapes: Post data augmentation, the shapes dataset now has 18,924 images of the three shapes. Its distribution is shown in *Figure 3*. The below distribution indicates that the circles have the highest number of images, followed by squares and triangles.

Chart, bar chart

Description automatically generated

Figure 3: Distribution of Shapes dataset

Lines: The program generates 22,500 images, with each line having an individual count of 7,500 images. Therefore, this is a balanced distribution ().

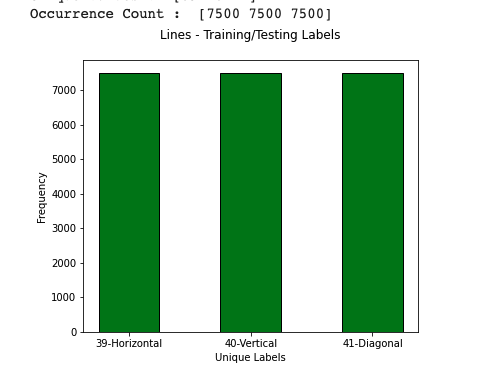


Figure 4: Distribution of lines

Letters: The dataset contains 145,600 images across 26 balanced classes. The training set has 124,800 images, while the test set has 20,800 images. *Figure 5* and *Figure 6* shows that both the training and test dataset are balanced. All the alphabets



Figure 5: Distribution of Training Letters

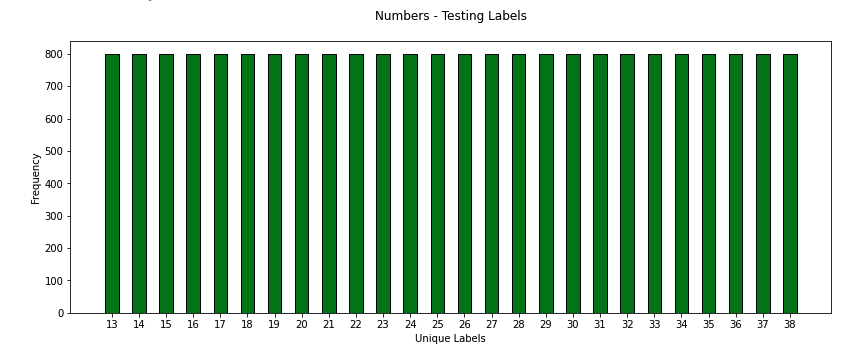
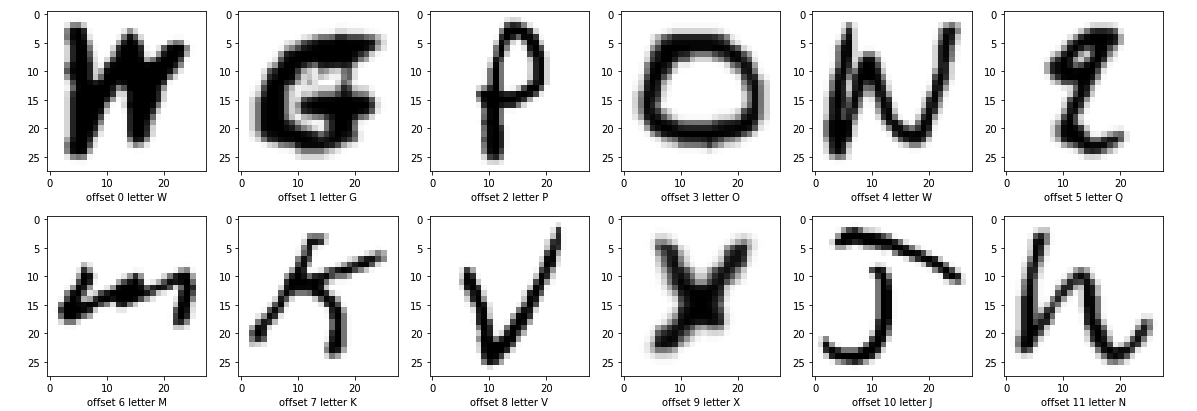


Figure 6: Distribution of Test Letters

have been labeled as numbers from 13 to 38. A display of a few handwritten digits shows that these letters are a combination of uppercase and lowercase.



# Methods

After the Exploratory Data Analysis and pre-processing of the images based on an approach described above, all datasets are merged. The train and test splits are created from this combined dataset using a 70:30 split, allocated to Train and Test datasets. The next step is constructing a deep learning model using a Sequential model. This study uses sequential type, which is quite popular as it can be easily developed by simply adding a layer on top of another layer. The which has the following steps

1. Defining the model
2. Compiling the model
3. Fit the model
4. Make predictions

This study uses Convolutional Neural Network architecture to build the model. The Convolutional Neural Network method is a powerful way to make predictions for unknown mapping relationships in data. The literature research section of this study concludes that CNN has been a preferred approach in image classification. A CNN architecture has two main parts:

1. Feature extraction: In this process, a convolution tool separates and identifies the various features of an image for analysis.
2. Classification: In this part, based on the features from the previous step, the fully connected layers utilize the output from the convolution process and predict the image class.

Diagram

Description automatically generated

Figure 7: Outline of CNN

Source: (Balaji 2020)

Within CNN, convolution is a mathematical process that combines the input data with a convolutional filter or kernel and produces a featured map. The convolution acts as a kernel passing over the image. By passing the kernel over the image from top left to bottom right, one-pixel step across at a time, the kernel multiplies its values with the values of the input image. These values are then passed on to a receptive field, where a hidden layer activation function shall be applied, another mathematical function. This study uses a Rectified Linear Unit or ReLU as the first selection for this model. This function converts all the negative values to zero and leaves the positive values intact. However, as the study progresses, this activation function's performance will be evaluated for potential improvement and consider other variants of ReLU activation functions to tweak model architecture.

**Pooling Layer**: This layer aims to reduce the computational load, memory usage, and the number of parameters that will limit the risk of overfitting. This layer selects the maximum element from the region of the feature map covered by the filter. It gives an output, which would be a feature map containing the most prominent features of the previous feature map. This study uses the Maxpooling layer, the most commonly used pooling layer type.

**Fully Connected Layers:**

**Flatten layer** is where the connection between the Convolution and Dense layer occurs. In this layer, the output from the convolution part of the CNN is converted into a 1-dimensional array. This layer will convert the output layer from NN into a long, single-dimension string of numbers for 784 (28 x 28), which will be fed into the Dense layer.

**A dense layer** is a classifier, which is the last stage of a CNN architecture. This layer uses the 1-dimensional array from the flatten layer and applies an output layer activation function. As the dataset is a multiclass problem, this study uses Softmax as the output activation function, which provides an array with one slot corresponding to each class. Each of these values corresponds to a probability score for each class that sums up to one. The prediction will be based on the class with the highest probability. Since multiple classes are involved, the output layer will have the same number of neurons as the number of input classes.

**Training model**: The model shall be trained for various epochs and batch sizes. A validation set shall be created from the training dataset. The performance of the model shall be assessed for its accuracy by reviewing the accuracy and loss functions for the training and validation data. The model can be tweaked for various hyperparameters either manually or using the Randomsearch algorithm for the following:

1. Kernel size
2. Padding
3. Strides
4. Learning rate
5. Number of epochs
6. Batch size
7. Activation function
8. Number of neurons

The second step is to tune the number of layers since fewer layers can lead to underfitting, while too many may lead to an overfitting model. This shall be carefully assessed based on the model performance. The model can also be tweaked by reviewing the visualization of the outputs of any Keras layer using the Keract library. The transformation of the input to the output prediction and the bottlenecks can be viewed using this library, allowing the model to improve.

**Model Compilation**: This is the final step toward creating a model. After compilation is done, we will move on to the training phase.

**Loss Function**: This function needs to be defined to find the error in the learning process for identifying the difference between the predicted value by the model and the actual value. For this study, we will use sparse categorical cross-entropy.

**Optimizer**: This process will optimize the input weights by comparing the prediction and the loss function, and we will use the Adam optimizer in this study.

**Metrics**: This is used to evaluate the performance of our model by reviewing the ratio of the "number of correct predictions" and "total number of predictions ."This study uses Accuracy as the Metric for our model.

The model performance shall be evaluated on the test dataset. The misclassification will be reviewed using a Heatmap / Clustermap to identify apparent patterns and tweak the model accordingly.

# Results

# Analysis and Interpretation

# Conclusions

# Directions for Future Work

# References

Baldominos, Alejandro, Yago Saez, and Pedro Isasi. “A Survey of Handwritten

Character Recognition with MNIST and EMNIST.” Applied Sciences 9, no. 15 (2019): 3169–. <https://doi.org/10.3390/app9153169>.

Cohen, Gregory, Saeed Afshar, Jonathan Tapson, and André van Schaik. “EMNIST:

An Extension of MNIST to Handwritten Letters.” arXiv:1702.05373v2 [cs.CV] (17 Feb 2017), <https://doi.org/10.48550/arXiv.1702.05373>

Lecun, Y, L Bottou, Y Bengio, and P Haffner. “Gradient-Based Learning Applied to

Document Recognition.” Proceedings of the IEEE 86, no. 11 (1998): 2278–2324. <https://doi.org/10.1109/5.726791>.

Hassan, Nahil Ahmed, Abhigith Neil Abraham, and Ajeesh Ramanujan. 2021. “Deep

Learning for Character Recognition” in Prakash, Kolla Bhanu., Ramani. Kannan, S.Albert. Alexander, and G. R. Kanagachidambaresan. 2021. Advanced Deep Learning for Engineers and Scientists : A Practical Approach. 1st ed. 2021. Cham: Springer International Publishing. <https://doi.org/10.1007/978-3-030-66519-7>.

Shamsuddin, Mohd Razif, Shuzlina Abdul-Rahman, and Azlinah Mohamed.

“Exploratory Analysis of MNIST Handwritten Digit for Machine Learning Modelling.” In Soft Computing in Data Science, 134–45. Singapore: Springer Singapore, 2018. <https://doi.org/10.1007/978-981-13-3441-2_11>.

Shinde, Akshada, Adwait Bhave, and Y. V. Haribhakta, “Handwriting Recognition on

Filled-in Forms Using CNN.” In Computer Vision and Image Processing, 88–98. Singapore: Springer Singapore, 2021. <https://doi.org/10.1007/978-981-16-1086-8_9>.

Yahya, Ali Abdullah, Jieqing Tan, and Min Hu. “A Novel Handwritten Digit

Classification System Based on Convolutional Neural Network Approach.” Sensors (Basel, Switzerland) 21, no. 18 (2021): 6273–. <https://doi.org/10.3390/s21186273>.