**HANDWRITTEN CHARACTER RECOGNITION USING CONVOLUTIONAL NEURAL NETWORKS (CNN)**

# 1. Introduction

Recognizing handwritten characters is difficult because of the variety of writing styles and the complexity of each individual's penmanship. The complexities of character recognition present a formidable obstacle to conventional processing techniques. Nevertheless, the latest developments in deep learning, particularly in convolutional neural networks (CNNs), have demonstrated incredible potential for multiple image identification tasks. This study intends to investigate the application of CNNs to developing models to recognize handwritten letters. The objective is to employ CNNs, which specialize in deriving valuable characteristics from images, in addressing the complex nature of handwritten character recognition.

## Background

The capacity to recognize handwritten characters is crucial in various disciplines, including document conversion to digital format, automated form processing, and text recognition. Such applications, beyond simple document recognition, necessitate a dependable technique. Recognizing handwritten characters manually is a laborious and error-prone exercise. Studies reveal that Convolutional Neural Networks (CNNs), a type of Artificial Neural Network, can provide an efficient method for autonomously recognizing handwritten characters. This research uses a CNN model to accomplish reliable and effective recognition of handwritten characters.

## Objectives

To accomplish the major objective of this research, which is to develop a model that is proficient in the automated identification of handwritten characters, the fundamental strategy that will be used is known as "harnessing the power of Convolutional Neural Networks," or "CNNs." The objective is to design a model that is precise and effective in correctly recognizing and deciphering handwritten characters. We hope to unleash the potential for substantial breakthroughs in automatic handwritten character recognition by using CNNs.

## Scope of the Project

Recognizing handwritten characters via manual techniques may be very time-consuming and error-prone, frequently resulting in inaccurate results. There is need for an automated program capable of identifying handwritten letters reliably and accurately. Developing a Convolutional Neural Network (CNN) model that is capable of adequately capturing and recognizing the complicated patterns and distinguishing characteristics found in handwritten characters is the primary purpose of this project. This project aims to simplify and improve the procedure of recognizing handwritten characters by drawing on the capabilities of deep learning.

# 2. Related Work

The history of handwritten character recognition technology is extensive and fascinating, beginning with the employment of handwriting reproduction tools like the pantograph in the middle of the 19th century. Significant advancements in the discipline, however, didn't occur until the 1960s with the introduction of computers and digital images. Pattern recognition and template matching algorithms served as the foundation for the earliest handwritten character recognition systems. The intrinsic diversity and complexity of handwriting hindered the recognition accuracy of these systems, which were dependent on character shape, size, and orientation analysis.

Artificial neural networks (ANNs) and other machine learning approaches completely changed the handwritten character recognition industry in the 1980s and 1990s. The recognition accuracy of these techniques could be improved over time by learning from examples. They faced a number of drawbacks, for example, the need for large training sets as well as the problem of having to deal with overlapping handwriting. Convolutional Neural Networks have proven to be highly efficient in image classification problems such as handwritten character recognition systems. They are able to not only handle extremely large datasets, but also learn complex patterns without the need for explicit feature extraction. However, handwritten character recognition systems face a number of restrictions regardless. These include the need for large training sets, the ambiguity of handwriting, image quality, and the inability to recognize overlapping characters. As a result, it is extremely difficult to develop a universal system that is able to recognize handwritten characters across multiple languages.

To improve the accuracy and adaptability of CNNs in image classification, various data preprocessing and augmentation techniques can be applied. Preprocessing the input image is a common step in traditional image processing techniques, which are then followed by feature extraction and classification. To enhance the quality of the input image, preprocessing operations may include strategies like binarization, noise reduction, and normalization. The size, shape, and orientation of individual characters are only a few examples of significant aspects that can be extracted from the input image and used for categorization. Techniques like template matching, k-nearest neighbor (k-NN), and decision trees are frequently used in classification approaches.

The recognition of handwritten characters has also been extensively aided by machine learning methods like ANNs and SVMs. With these techniques, a model is trained on a dataset of labeled input photos before being applied to the classification of new input images. Because they can distinguish various handwriting styles and variants and can be trained to recognize complicated patterns and characteristics in the input data, ANNs are particularly successful for image recognition applications. Another well-liked method is the use of SVMs, which can be applied to both classification and regression tasks and can handle high-dimensional data.

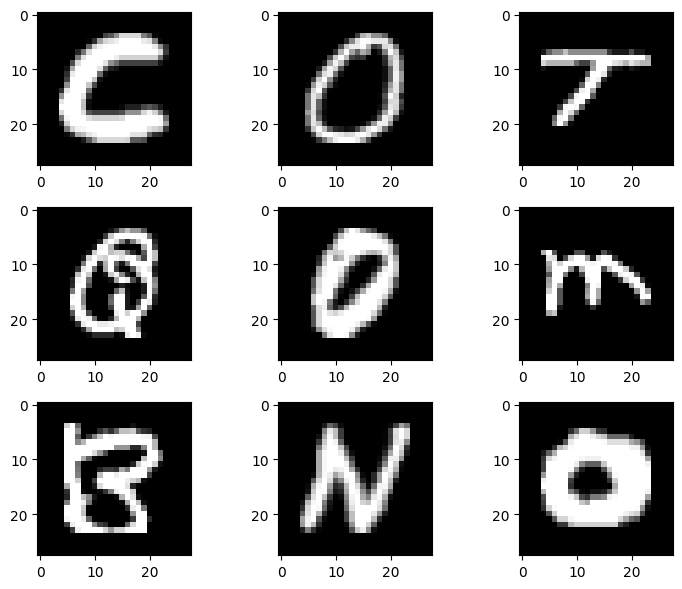
For the recognition of handwritten characters, deep learning models like CNNs and RNNs have gained popularity recently. These models can automatically extract pertinent characteristics from the raw input photos without the need for explicit feature extraction, and they can also learn hierarchical representations of the input data. In many benchmark datasets, including MNIST and the NIST Special Database 19, CNNs in particular have demonstrated state-of-the-art performance, outperforming more conventional techniques like template matching and feature extraction. Multiple CNNs or other models combined into ensemble models have also demonstrated enhanced performance. Depending on the particular task and language/script that needs to be identified, a broad range of datasets have been utilized in earlier studies on handwritten character recognition. Some researchers have made use of commonly available benchmark datasets, such MNIST and the NIST Special Database 19, for assessing the performance of various models. Some studies have even employed synthetic datasets to increase the number and variety of training data, while other studies have used proprietary datasets tailored to certain languages or scripts.

# 3. Dataset

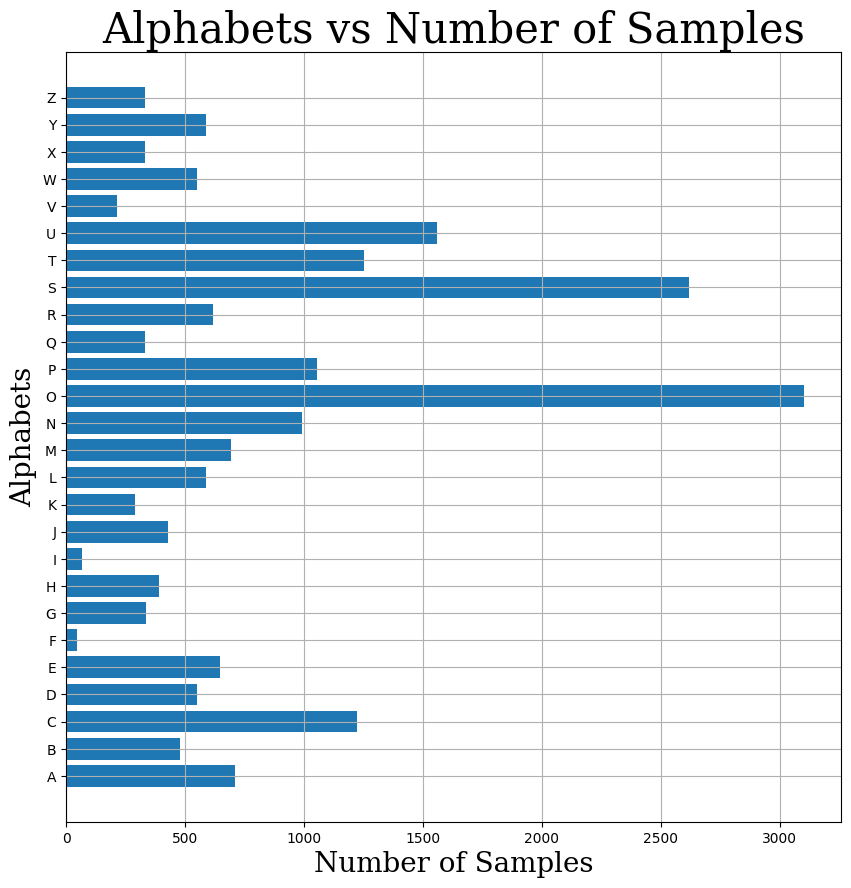
## 3.1 Overview of the Dataset

The dataset was obtained from the following [URL](https://www.kaggle.com/datasets/sachinpatel21/az-handwritten-alphabets-in-csv-format/download?datasetVersionNumber=5). It contains 372450 images of handwritten characters represented as arrays of shape (785,) where the actual image has a shape of (784,) and the extra column represents the label for that image. The dataset is labeled with categories of characters corresponding to the associated IDs (the labels are encoded as integers from 0 through 25 representing the letters A to Z).

## 3.2 Sample Images



## 3.3 Distribution of data from a sample obtained from the dataset



# 4. Methodology

## 4.1 Data Preprocessing

The following data preprocessing techniques were applied to the dataset:

1. **Data Sampling**

The dataset originally contained 372450 rows of data. However, a random sample of 20000 rows was obtained from the dataset so as to make the training process more manageable and faster. This was made possible by use the sample function from Pandas without replacement, which eliminates the chances of duplicates.

1. **Splitting Features and Target**

In Machine Learning, it is a common practice to separate the independent variables (features) from the dependent variable (target). The target (which is the first column in the dataset) is stored in a variable y, whereas the features (the rest of the columns excluding the first column) are stored in a variable named X.

1. **Data Conversion**

The data was then converted from a Pandas DataFrame to Numpy arrays since popular frameworks in Machine Learning expect data in the form of arrays. I used the np.array() function to achieve this.

1. **Features Reshaping**

In the original dataset, images are represented as 1D arrays of shape (784,). Convolutional Neural Networks, however, expect images as 2D arrays where each image is represented as a matrix with height, width, and channels (for colored images). As such, the images are reshaped to be compatible with the CNNs.

1. **Splitting into Train and Test Sets**

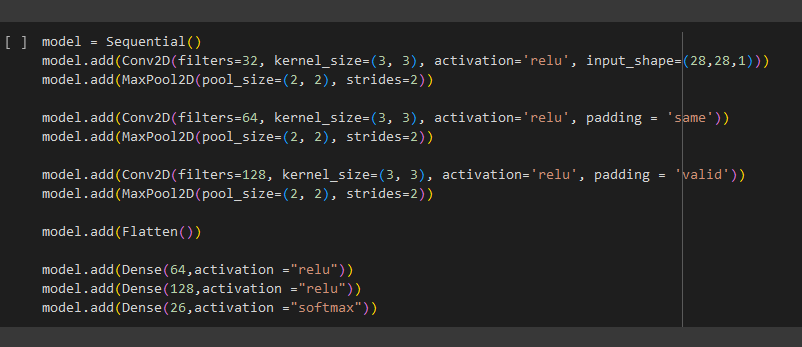
It is good practice to evaluate the performance of a Machine Learning model on unseen data. This is accomplished by splitting the dataset into the training and testing sets by the help of the train\_test\_split function from scikit-learn.

1. **One-Hot Encoding the Target**

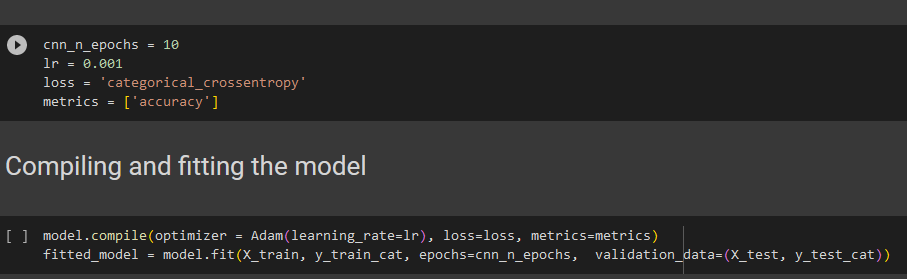
It is common to represent the target as binary vectors, especially when dealing with multiclass classification in Machine Learning. This process is called encoding and it involves transforming all the elements in a binary vector to zero except for the element corresponding to the class label. I used Keras’ to\_categorical function to one-hot encode the target variable. The parameter num\_classes=26 specifies the number of classes (letters A to Z) in the dataset.

## 4.2 Model Architecture Design

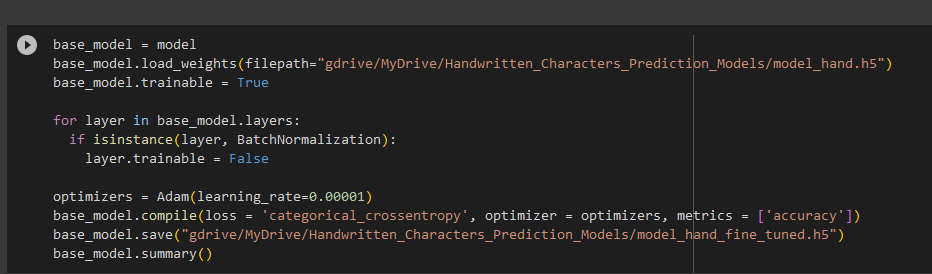
The model was created using the Keras Sequential API, which allows building models layer-by-layer. The architecture of the CNN consists of three convolutional layers, each followed by a max-pooling layer to reduce spatial dimensions and capture important features. ReLU (Rectified Linear Unit) is used as the activation function in the convolutional layers to introduce non-linearity. The last layer is a dense layer with 26 units and a softmax activation function, representing the 26 classes (letters A to Z) for multiclass classification.



The model is then compiled using the Adam optimizer, categorical cross-entropy loss function, and accuracy as the evaluation metric. It is trained on the training data with a specified number of epochs. The training process involves forward and backward passes to update the model's weights using the backpropagation algorithm.



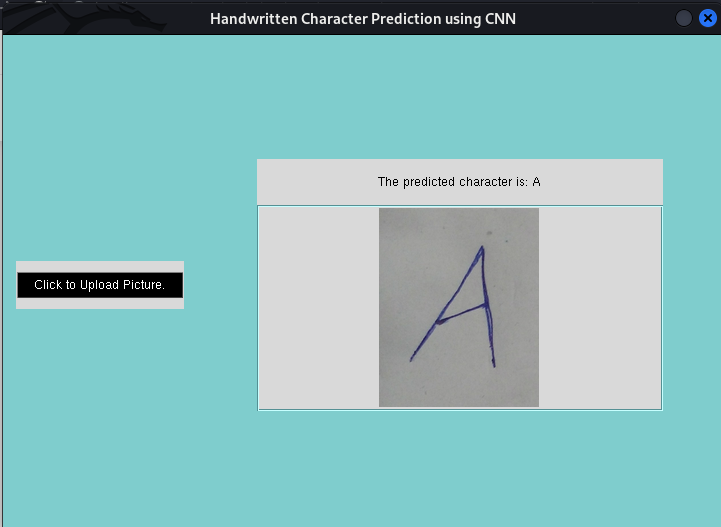
After the initial training, the model undergoes fine-tuning. Some layers (BatchNormalization layers) are frozen, preventing their weights from being updated during fine-tuning, while the rest of the layers are trainable. This is to fine-tune the model on specific layers with a lower learning rate, allowing it to adapt to the data better.



The accuracy and loss curves of the initial training and fine-tuning are plotted to visualize the model's performance over the epochs. Finally, the model is evaluated on the test data, and the accuracy of the model is calculated by comparing the predicted labels with the actual labels.

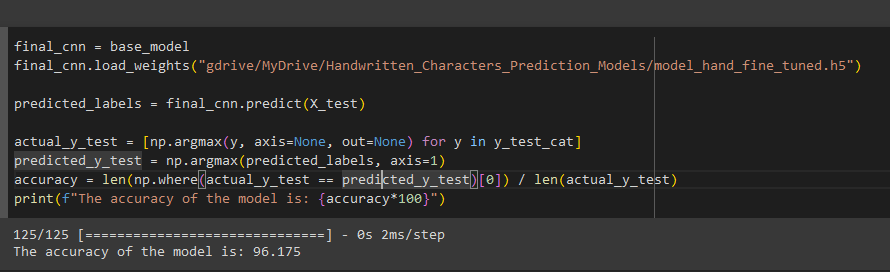
## 4.3 Graphical User Interface

As part of the research project, I built a Graphical User Interface (GUI). The GUI allows users to interact with the model at a higher level abstracting the inner details of how the model works. It allows for users to upload an image of a handwritten character and have the model predict what class it belongs to. The GUI will be built using Tkinter.

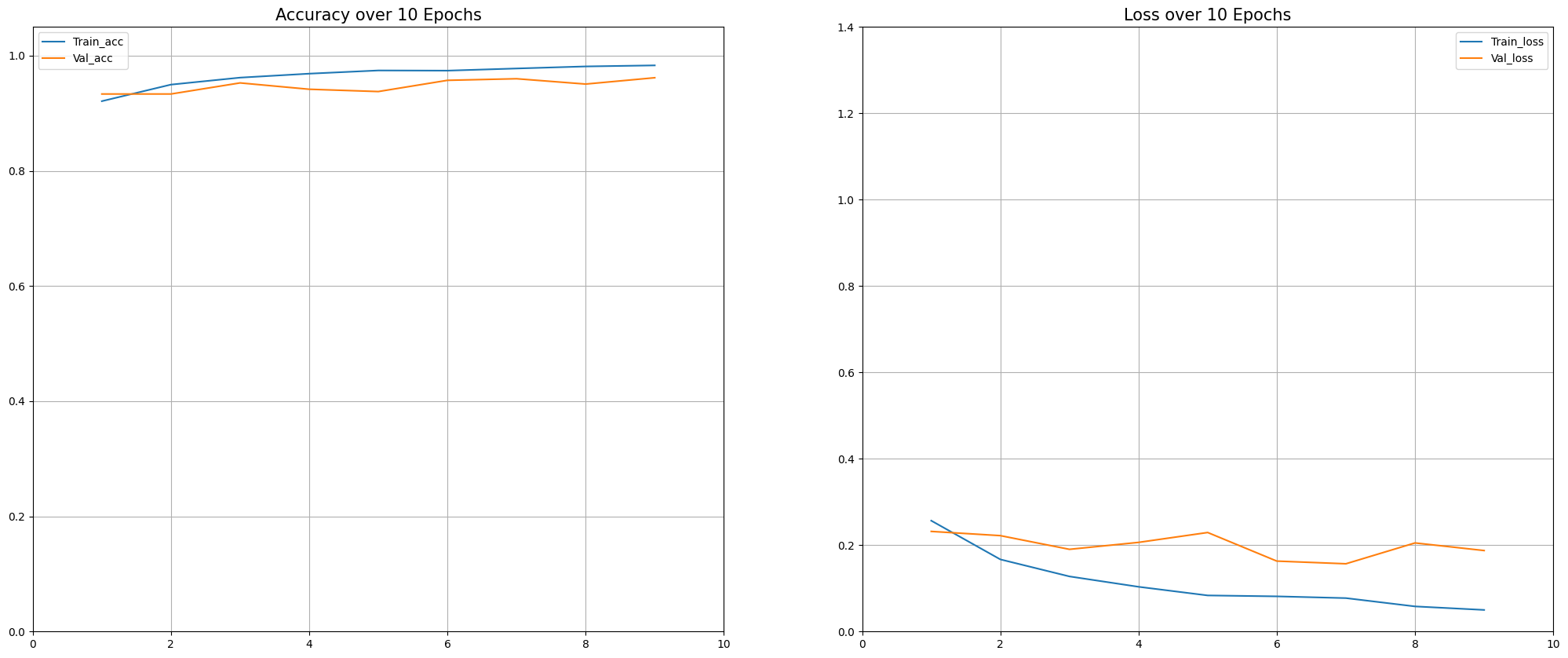


# 5. Results

The trained model achieved an accuracy of 96.175 on the validation set.



**Accuracy and Loss Curve on initial model**



**Accuracy and Loss Curve of fine tuned model**

