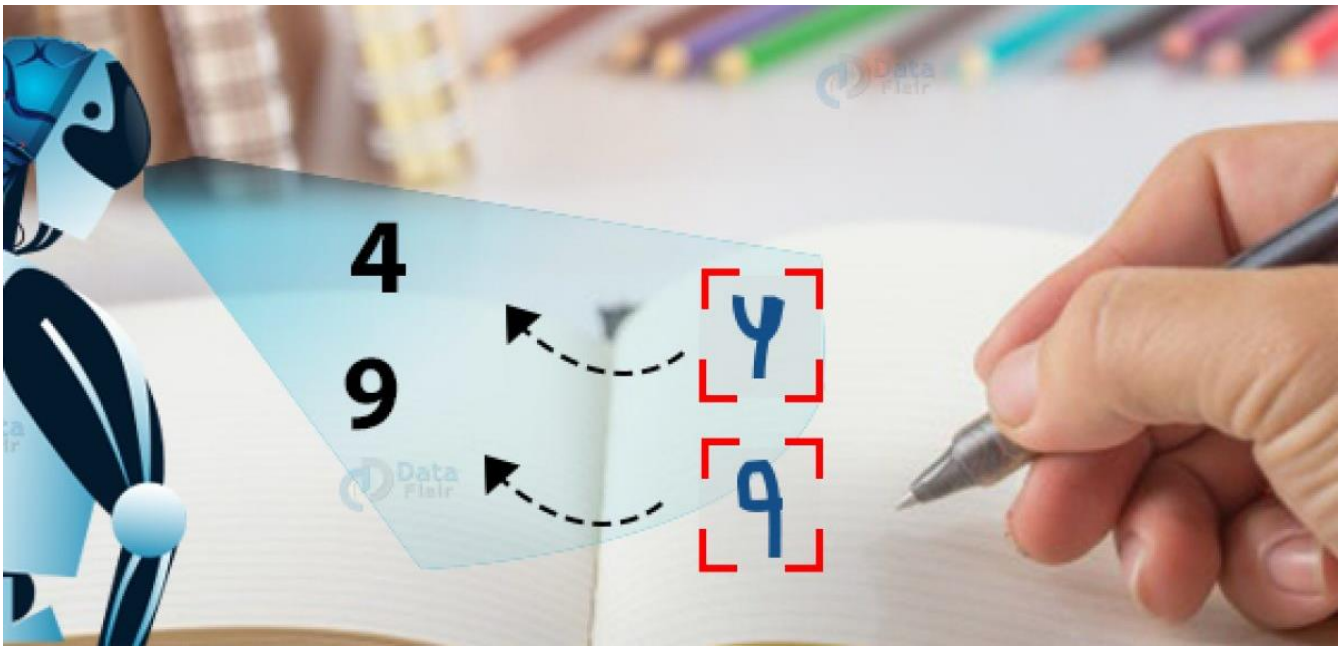
Handwritten Digit Recognition using Convolutional Neural Networks

Lebanese University
Faculty of Engineering
4th year



Presented to:
Mohamad Aoude

Presented by:
Ragheed Ismail 5652
Hasan Ibrahim 5446

Abstract

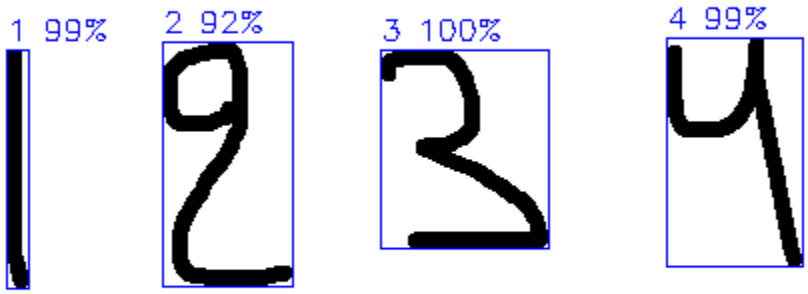
The reliance of humans over machines has never been so high such that from object classification in photographs to adding sound to silent movies everything can be performed with the help of deep learning and machine learning algorithms. Likewise, Handwritten text recognition is one of the significant areas of research and development with a streaming number of possibilities that could be attained. Handwriting recognition (HWR), also known as Handwritten Text Recognition (HTR), is the ability of a computer to receive and interpret intelligible handwritten input from sources such as paper documents, photographs, touch-screens and other devices. Apparently, this paper illustrates handwritten digit recognition with the help of MNIST datasets using Convolution Neural Network (CNN) models. The main objective of this paper is to create a graphical user interface (GUI) and use it to detect number written inside of it.

Table of Contents

Introduction

Handwritten digit recognition is the ability of a computer to recognize the human handwritten digits from different sources like images, papers, touch screens, etc. and classify them into 10 predefined classes (0-9). This has been a topic of boundless-research in the field of deep learning. Digit recognition has many applications like number plate recognition, postal mail sorting, bank check processing, etc. In Handwritten digit recognition, everyone faces many challenges because of different styles of writing of different peoples as it is not an Optical character recognition.

Every day, developers are working hard on machines to make them more smart and intelligent by using machine learning and deep learning techniques so that they can perform tasks similar to humans. With the help of these techniques human effort can be reduced in recognizing, learning, predictions and many other areas.



Our project focuses on the MNIST data set and using Convolutional Neural Networks (CNN) in order to train our program to read handwritten numbers.

Methodology

I. Dataset

Handwritten character recognition is an expansive research area that already contains detailed ways of implementation which include major learning datasets, popular algorithms, features scaling & feature extraction methods. MNIST dataset (Modified National Institute of Standards and Technology database) is the subset of the NIST dataset which is a combination of two of NIST's databases: Special Database 1 and Special Database 3. Special Database 1 and Special Database 3 consist of digits written by high school students and employees of the United States Census Bureau, respectively. MNIST contains a total of 70,000 handwritten digit images (60,000 - training set & 10,000 - test set) in 28x28 pixel bounding box and anti-aliased. All these images have corresponding Y values which apprises what the digit is.

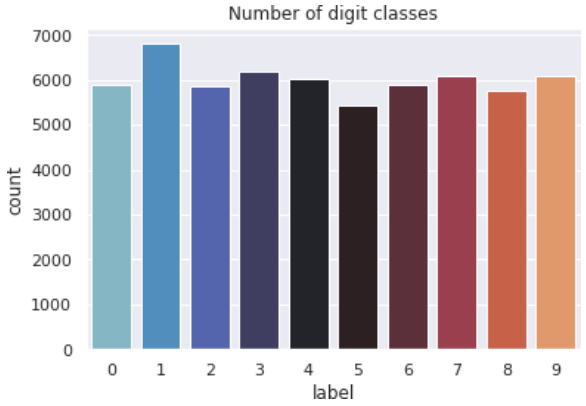


Figure 1. Bar graph illustrating the MNIST handwritten digit training dataset (Label vs Total number of training samples)

Figure 2. Plotting of some random MNIST Handwritten digit

II. Convolutional Neural Network (CNN)

CNN is a deep learning algorithm that is widely used for image recognition and classification. It is a class of deep neural networks that require minimum pre-processing. It inputs the image in the form of small chunks rather than inputting a single pixel at a time, so the network can detect uncertain patterns (edges) in the image more efficiently. CNN contains 3 layers namely, an input layer, an output layer, and multiple hidden layers which include Convolutional layers, pooling layers (Max and Average pooling), Fully connected layers (FC), and normalization layers. CNN uses a filter (kernel) which is an array of weights to extract features from the input image. CNN employs different activation functions at each layer to add some non-linearity. Further, we observe the height and width decrease while the number of channels increases. Finally, the generated column matrix is used to predict the output.

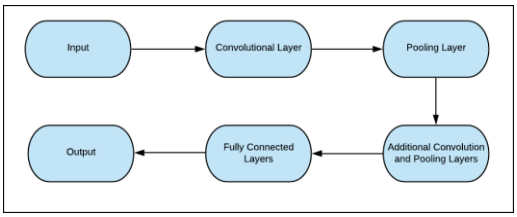


Figure 3. This figure shows the architectural design of CNN layers in the form of a Flow chart

Implementation

In order to implement CNN, we opted to use python programming language since it includes the MNIST dataset in its libraries and a graphical user interface can be programmed to test our results with it.
Our Python project requires five main libraries to function.
These libraries are:

- numpy
- tensorflow
- keras
- pillow
- openvc

They can be manually installed or by using the "pip install" command in cmd.

The Programing steps are as follows:

I. Import the Libraries and Load the Dataset

Import all the modules that we are going to need for training our model. The Keras library already contains some datasets and MNIST is one of them. So we can easily import the dataset and start working with it. The mnist.load_data() method returns us the training data, its labels and also the testing data and its labels.

II. Preprocess the Data:

The image data cannot be fed directly into the model so we need to perform some operations and process the data to make it ready for our neural network. The dimension of the training data is (60000,28,28). The CNN model will require one more dimension so we reshape the matrix to shape (60000,28,28,1).

III. Create the Model

A CNN model generally consists of convolutional and pooling layers. It works better for data that are represented as grid structures, this is the reason why CNN works well for image classification problems. The dropout layer is used to deactivate some of the neurons and while training, it reduces offer fitting of the model. We will then compile the model with the Adadelta optimizer.

IV. Train the Model

The model.fit() function of Keras will start the training of the model. It takes the training data, validation data, epochs, and batch size.
It takes some time to train the model. After training, we save the weights and model definition in the 'final_model.h5' file.

V. Evaluate the model

There are 10,000 images in the dataset which will be used to evaluate how good the model works. The testing data was not involved in the training of the data therefore, it is new data for our model. The MNIST dataset is well balanced so we can get around 99% accuracy with a loss less than 0.4.

VI. Create GUI to Predict Digits

The aim is to build an interactive window to draw digits on canvas and with a button, we can recognize the digit. The Tkinter library comes in the Python standard library.
A function Recognize_Digit() that takes the image as input and then uses the trained model to predict the digit.
A function clear_widget() that resets the screen to blank.
A function activate_event(event) that is used to detect when we press on Mouse Button 1.
A function draw_lines(event) that draws lines where we hold Mouse Button 1.

Conclusion:

This Project revolves around a GUI that is programmed and coded to recognize handwritten digits with accuracy that varies between 92% and 100% and that is by using the Convolutional Neural Network (CNN). This can be handy in many fields and applications such as online handwriting recognition on computer tablets, recognize zip codes on mail for postal mail sorting, processing bank check amounts, numeric entries in forms filled up by hand (for example - tax forms) and so on. There were different challenges faced while attempting to solve this problem. The handwritten digits are not always of the same size, thickness, or orientation and position relative to the margins.

References

- https://en.wikipedia.org/wiki/Handwriting_recognition
- https://www.quora.com/What-can-a-digit-recognizer-be-used-for
- https://medium.com/dataseries/basic-overview-of- convolutional-neural-network-cnn-4fcc7dbb4f17
- https://github.com/dixitritik17/Handwritten-Digit-Recognition
- https://medium.com/analytics-vidhya/handwritten-digit-recognition-gui-app-46e3d7b37287