

MINOR PROJECT REPORT

on

HANDWRITTEN DIGIT RECOGNIZER USING TENSORFLOW AND PYGAME

Submitted in partial fulfillment of requirements for the degree of

Bachelor of Technology (B. Tech)

in

Computer Science and Engineering & Electrical Engineering



Submitted by:

GOVIND BANURA (ET16BTHCS057)

PRACHI SUREKA (ET16BTHCS029)

AAYUSHI KUNDALIA (ET16BTHCS061)

AMIT KUMAR SINGH (ET16BTHCS067)

MANTU DEKA (ET16BTHEE022L)

School of Engineering and Technology

The Assam Kaziranga University

November 2019



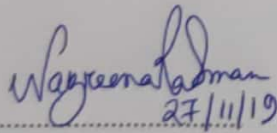
SCHOOL OF ENGINEERING AND TECHNOLOGY
THE ASSAM KAZIRANGA UNIVERSITY
JORHAT-785006:: ASSAM:: INDIA

CERTIFICATE

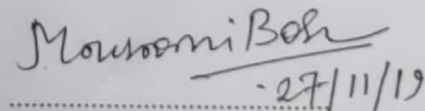
This is to certify that the project report entitled **HANDWRITTEN DIGIT RECOGNIZER USING TENSORFLOW AND PYGAME**, submitted to the Department of Computer Science and Engineering & Electrical Engineering, Kaziranga University, in partial fulfillment for the award of the degree of **Bachelor of Technology** in Computer Science and Engineering & Electrical Engineering, is a record of bonafide work carried out by **Mr. Govind Banura (ET16BTHCS057)**, **Miss Prachi Sureka (ET16BTHCS029)**, **Miss Aayushi Kundalia (ET16BTHCS061)**, **Mr. Amit Kumar Singh (ET16BTHCS067)** and **Mr. Mantu Deka (ET16BTHEE022L)** under my supervision and guidance.

All help received by them from various sources has been duly acknowledged.

No part of this report has been submitted elsewhere for the award of any other degree.


27/11/19

Internal Guide
**MS. NAZREENA
RAHMAN**
(Associate Professor of
CSE Department)


27/11/19

**MRS. MOUSOMI
BORAH**
(Head of the Department)

Dept. Of Comp. Science & Engineering
The Assam Kaziranga University
Assam - 785006

ABSTRACT

For humans, identifying numbers or items in a picture is extremely simple, but how do you train the machine to recognize these different things in images? Convolutional Neural Networks (CNN) can solve this problem. The goal of this project is to observe the accuracy of CNN to identify handwritten digits using various numbers of hidden layers and epochs. To make the user interface for outlining the numbers as an input and to predict the result, what exactly the numbers are. For this performance evaluation of CNN, we performed our experiment using the Modified National Institute of Standards and Technology (MNIST) dataset. Obviously, human beings can perceive that there is a hierarchy of conceptual structure in the image, but the machine does not, for example the trained neural network is inconvenient to deal with special changes in a position of numbers in digital pictures. Exactly put, no matter what the environment of the image (image background) is, it is unchallenging for human beings to judge whether there is such a figure in the image and it is unnecessary to repeat the learning training.

Acknowledgment

For the completion of our report we are indebted to plenty of people for the very sincere cooperation that they extended to us at various stages. This project has been completed under the supervision of our honorable guide Mr. Pankaj Chittora and Mr. Aaditya Maheswari who supported us and was with us from the beginning to the completion of this project.

We are also thankful to Ms. Nazreena Rahman for supporting us as an internal guide of the project.

Finally, we are very thankful to all staff members, volunteers and our classmates for their support and hard work for making this project a success

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. We understand that any violation of the above will be cause for disciplinary action by the University and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed. Project or Project code is not copied or downloaded from the Internet or any other resources.

(Signature)s

Govind Banura

Prachi Sureka

Aayushi Kundalia

H. K. Singh

Mantu Deka

(Name of the student)s Govind Banura

Prachi Sureka

Aayushi Kundalia

Amit Kumar Singh

Mantu Deka

Date: 27/11/19

CONTENTS

Abstract	III
Acknowledgment	IV
Declaration	V
List of Figures	VIII
1. Introduction	1
1.1 General Introduction	1
1.2 Machine Learning	1
1.3 Neural Network	2
1.4 Statement of Purpose	3
2. Requirement Analysis	4
2.1 Anaconda Navigator	4
2.2 Python 3.7 (64-bit)	4
2.3 TensorFlow	5
2.4 OpenCV	6
2.5 NumPy	6
2.6 Keras	6
2.7 PyGame	7
2.8 Sklearn	7
2.9 Pandas	8
2.10 SciPy	8
2.11 Matplotlib	8

3. Workflow	9
3.1 CNN	9
3.2 OpenCV	11
3.3 Preparing Data	12
4. Implementation and Verification	14
4.1 Backend Algorithm	14
4.2 User Interface with PyGame	15
4.3 Interface Between UI and Model (Image Processing)	17
4.4 How to Use	18
5. Result Analysis	19
5.1 Model Accuracy	19
5.2 Integration and Final Working Model	19
6. Conclusion and Future Scope	20
References	

LIST OF FIGURES

Figure 1.3.1 A Basic Neural Network	3
Figure 3.1.1 CNN Architecture	9
Figure 3.1.2 Pooling Example	11
Figure 3.1.3 Fully Connected Model	11
Figure 3.3.1 Example of MNIST Dataset	13
Figure 3.3.2 Class Percentages in MNIST Dataset	13
Figure 4.4.1 GUI	18
Figure 5.2.1 GUI with the Result	19

Chapter 1

Introduction

1.1 General Introduction:

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. The image of the written text may be sensed “offline” from a piece of paper by optical scanning (optical character recognition) or intelligent word recognition. Alternatively, the movements of the pen tip may be sensed “on line”, for example by a pen-based computer screen surface, a generally easier task as there are more clues available. A handwriting recognition system handles formatting, performs correct segmentation into characters, and finds the most plausible words.

Handwritten digits recognition problem has been studied by researchers since 1998 with almost all the algorithms designed by then and even until now. The test error rate decreased from 12% in 1988 by the linear classifier to 0.23% in 2012 by convolutional nets, and these days more and more data scientists and machine learning experts are trying to develop and validate unsupervised learning methods such as auto-encoder and deep learning model. In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.

1.2 Machine Learning:

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solve. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.

Two of the most widely adopted machine learning methods are: -

- **Supervised learning** which trains algorithms based on example input and output data that is labeled by humans.
- **Unsupervised learning** provides the algorithm with no labeled data in order to allow it to find structure within its input data.

1.3 Neural Network:

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so, the network generates the best possible result without needing to redesign the output criteria. A neural network works similarly to the human brain's neural network. A "neuron" in a neural network is a mathematical function that collects and classifies information according to a specific architecture. It collects inputs from different nodes or any external sources and determines the output.

The neurons are arranged in three interconnected layers-

The input layer collects input patterns. It is even responsible for extracting patterns from the input dataset. Every input has a weight-related to it, allocated on the grounds of its importance relative to the other different inputs.

Hidden layers fine-tune the input weightings until the neural network's margin of error is minimal. It is hypothesized that hidden layers extrapolate salient features in the input data that have predictive power regarding the outputs. This describes feature extraction, which accomplishes a utility similar to statistical techniques such as principal component analysis.

The output layer has classifications or output signals (activation functions) to which input patterns may map. An activation function is used to specify the type of output to be obtained from a particular node.

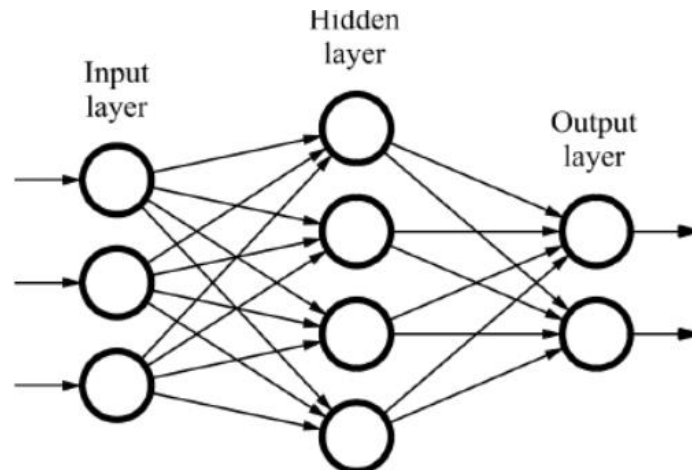


Figure 1.3.1: A basic neural network.

1.4 Statement of Purpose:

We aim at the construction of a classifier for recognition of handwritten digits using neural networks (most widely used machine learning methods).

There is a wide class of neural networks. The classifier is made based upon a convolutional neural network (CNN). CNN is part of deep, feed-forward artificial neural networks that can perform a variety of tasks with even better time and accuracy than other classifiers, in different applications of image and video recognition, recommender system and natural language processing.

The model is to be trained using the MNIST dataset and further testing of the classifier is done using live handwritten digits. Based on the predictions made by the classifier, the performance of the model is measured.

Every machine learning model has a chance of showing some percent of error in the results it shows. It is because a machine cannot be fully trained to think and behave like a human being.

The model which would be constructed for the digits may even show some fluctuations in the predictions.

Chapter 2

Requirement Analysis

Handwritten digit recognition technology has hit the market recently. A popular demonstration of the capability of deep learning techniques is object recognition in image data. This triggered a range of new ideas coming to creative minds.

However, it takes a lot of work to turn this idea into a project. In fact, it requires a complete step-by-step project strategy starting from goal definition to publishing and maintenance. In this chapter, we will try to cover all the requirements which we will need further.

To make a chatbot, the requirements are:

- Anaconda Navigator
- Python 3.7 (64-bit)
- TensorFlow
- OpenCV
- NumPy
- Keras
- PyGame
- Sklearn
- Pandas
- Scipy
- Matplotlib

2.1 Anaconda Navigator:

Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. Package versions are managed by the package management system *conda*. The Anaconda distribution is used by over 15 million users and includes more than 1500 popular data-science packages suitable for Windows, Linux, and macOS.

We can download this from the official website of Anaconda Navigator. We will need Jupiter Notebook, which we can directly use through Anaconda Navigator

2.2 Python 3.7 (64-bit):

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. It is high-level built-in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program

maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

1. Open a browser window and navigate to the Download page for Windows at python.org.
2. Underneath the heading at the top that says **Python Releases for Windows**, click on the link for the **Latest Python 3 Release - Python 3.x.x**. (As of this writing, the latest in Python 3.7.x)
3. Scroll to the bottom and select **Windows x86-64 executable installer** for 64-bit. We strongly recommend 64-bit because TensorFlow doesn't support 32-bit.
4. Once you have chosen and downloaded an installer, simply run it by double-clicking on the downloaded file.

Important: You want to be sure to check the box that says **Add Python 3.x to PATH** as shown to ensure that the interpreter will be placed in your execution path.

5. Then just click **Install Now**. That should be all there is to it. A few minutes later you should have a working Python 3 installation on your system.

NOTE: If you are using Anaconda Navigator then you don't need to install python separately. Anaconda Navigator already contains the latest version of python in a 64-bit version.

2.3 TensorFlow:

To install this package with conda run one of the following:

```
conda install -c conda-forge tensorflow
```

```
conda install -c conda-forge/label/cf201901 tensorflow
```

To install this package with python, run the following:

```
pip install --user tensorflow
```

Created by the Google Brain team, TensorFlow is an open-source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor. It uses Python to provide a convenient front-end API for building applications with the framework while executing those applications in high-performance C++.

TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential

equation) based simulations. Best of all, TensorFlow supports production prediction at scale, with the same models used for training.

2.4 OpenCV:

To install this package with conda run one of the following:

```
conda install -c conda-forge opencv  
conda install -c conda-forge/label/gcc7 opencv  
conda install -c conda-forge/label/broken opencv  
conda install -c conda-forge/label/cf201901 opencv
```

To install this package with python, run the following:

```
pip install opencv-python
```

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. In simple language, it is library used for Image Processing. It is mainly used to do all the operations related to Images.

2.5 NumPy:

To install this package with conda run the following:

```
conda install -c conda-forge numpy
```

To install this package with python, run the following:

```
pip install numpy
```

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

2.6 Keras:

To install this package with conda run one of the following:

```
conda install -c conda-forge keras  
conda install -c conda-forge/label/cf201901 keras
```

To install this package with python, run the following:

```
pip install keras
```

Keras was created to be user-friendly, modular, easy to extend and to work with Python. The API was “designed for human beings, not machines,” and “follows best practices for reducing cognitive load.”

Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can combine to create new models. New modules are simple to add, as new classes and functions. Models are defined in Python code, not separate model configuration files.

The biggest reasons to use Keras stem from its guiding principles, primarily the one about being user-friendly. Beyond ease of learning and ease of model building, Keras offers the advantages of broad adoption, support for a wide range of production deployment options, integration with at least five back-end engines (TensorFlow, CNTK, Theano, MXNet, and PlaidML), and strong support for multiple GPUs and distributed training. Plus, Keras is backed by Google, Microsoft, Amazon, Apple, Nvidia, Uber, and others.

2.7 PyGame:

To install this package with conda run one of the following:

```
conda install -c cogsci pygame
```

To install this package with python, run the following:

```
pip install pygame
```

Pygame uses the Simple DirectMedia Layer (SDL) library,^[a] with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. This is based on the assumption that the most expensive functions inside games can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game.

2.8 Sklearn:

To install this package with conda run one of the following:

```
conda install -c anaconda scikit-learn
```

To install this package with python, run the following:

```
pip install scikit-learn
```

Scikit-learn (formerly **scikits.learn** and also known as **sklearn**) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

2.9 Pandas:

To install this package with conda run one of the following:

```
conda install -c anaconda pandas
```

To install this package with python, run the following:

```
pip install pandas
```

Pandas offer data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

2.10 Scipy:

To install this package with conda run one of the following:

```
conda install -c anaconda scipy
```

To install this package with python, run the following:

```
pip install scipy
```

SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

2.11 Matplotlib:

To install this package with conda run one of the following:

```
conda install -c conda-forge matplotlib
```

```
conda install -c conda-forge/label/testing matplotlib
```

```
conda install -c conda-forge/label/testing/gcc7 matplotlib
```

```
conda install -c conda-forge/label/gcc7 matplotlib
```

```
conda install -c conda-forge/label/broken matplotlib
```

```
conda install -c conda-forge/label/rc matplotlib
```

```
conda install -c conda-forge/label/cf201901 matplotlib
```

To install this package with python, run the following:

```
pip install matplotlib
```

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

Chapter 3

Workflow

3.1 CNN:

Convolutional Neural Networks have a different architecture than regular Neural Networks. Regular Neural Networks transform input by putting it through a series of hidden layers. Every layer is made up of a set of neurons, where each layer is fully connected to all neurons in the layer before. Finally, there is a last fully-connected layer — the output layer — that represent the predictions.

Convolutional Neural Networks are a bit different. First of all, the layers are organized in 3 dimensions: width, height, and depth. Further, the neurons in one layer do not connect to all the neurons in the next layer but only to a small region of it. Lastly, the final output will be reduced to a single vector of probability scores, organized along the depth dimension.

CNN is composed of two major parts:

- **Feature Extraction:**

In this part, the network will perform a series of convolutions and pooling operations during which the features are detected. If you had a picture of a zebra, this is the part where the network would recognize its stripes, two ears, and four legs.

- **Classification:**

Here, the fully connected layers will serve as a classifier on top of these extracted features. They will assign a probability for the object on the image being what the algorithm predicts it is.

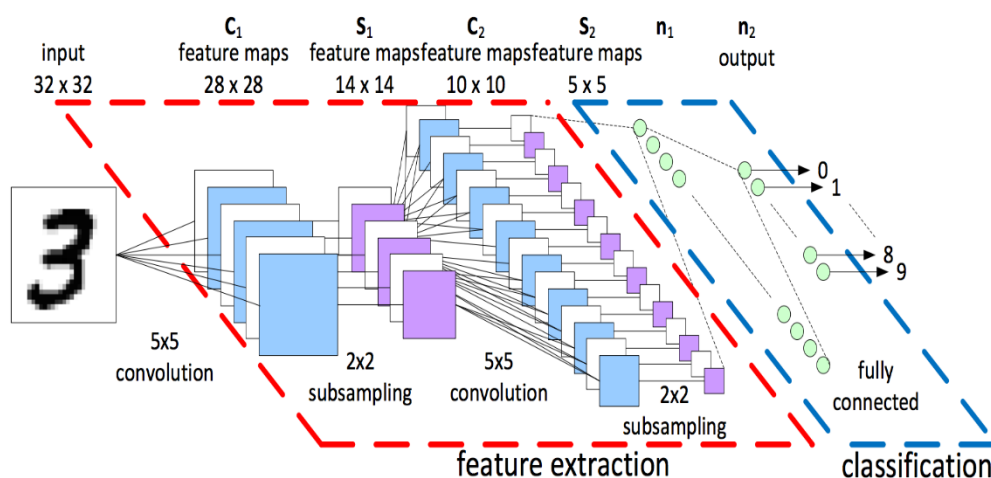


Figure 3.1.1: Convolutional Neural Networks architecture

There are squares and lines inside the red dotted region which we will break it down later. The green circles inside the blue dotted region named classification are the neural network or multi-layer perceptron which acts as a classifier. The inputs to this network come from the preceding part named feature extraction.

Now let us see about a bit of mathematics which is involved in the whole convolution process.

- Convolution layers consist of a set of learnable filters (a patch in the above image). Every filter has small width and height and the same depth as that of input volume (3 if the input layer is image input).
- For example, if we have to run convolution on an image with dimension $34 \times 34 \times 3$. Possible size of filters can be $a \times a \times 3$, where 'a' can be 3, 5, 7, etc but small as compared to image dimension.
- During forward pass, we slide each filter across the whole input volume step by step where each step is called stride (which can have value 2 or 3 or even 4 for high dimensional images) and compute the dot product between the weights of filters and patch from input volume.
- As we slide our filters, we'll get a 2-D output for each filter and we'll stack them together and as a result, we'll get output volume having a depth equal to the number of filters. The network will learn all the filters.

A CNN is a sequence of layers, and every layer transforms one volume to another through differentiable functions.

Types of layers:

Let us take an example by running a CNN on of image of dimension $32 \times 32 \times 3$.

- 1 **Input Layer:** This layer holds the raw input of image with width 32, height 32 and depth 3.
- 2 **Convolution Layer:** This layer computes the output volume by the computing dot product between all filters and image patches. Suppose we use total 12 filters for this layer we'll get output volume of dimension $32 \times 32 \times 12$.
- 3 **Activation Function Layer:** This layer will apply the element-wise activation function to the output of convolution layer. Some common activation functions are RELU: $\max(0, x)$, Sigmoid: $1/(1+e^{-x})$, Tanh, Leaky RELU, etc. The volume remains unchanged hence output volume will have dimension $32 \times 32 \times 12$.
- 4 **Pool Layer:** This layer is periodically inserted in the comments and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting. Two common types of pooling layers are **max pooling** and **average pooling**. If we use a max pool with 2×2 filters and stride 2, the resultant volume will be of dimension $16 \times 16 \times 12$.

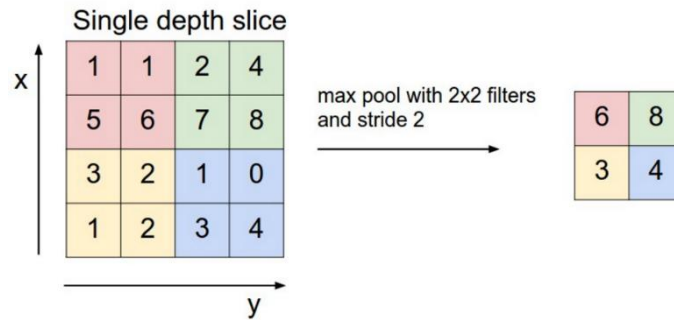


Figure 3.1.2: Pooling example

- 5 **Fully-Connected Layer:** This layer is a regular neural network layer that takes input from the previous layer and computes the class scores and outputs the 1-D array of size equal to the number of classes.

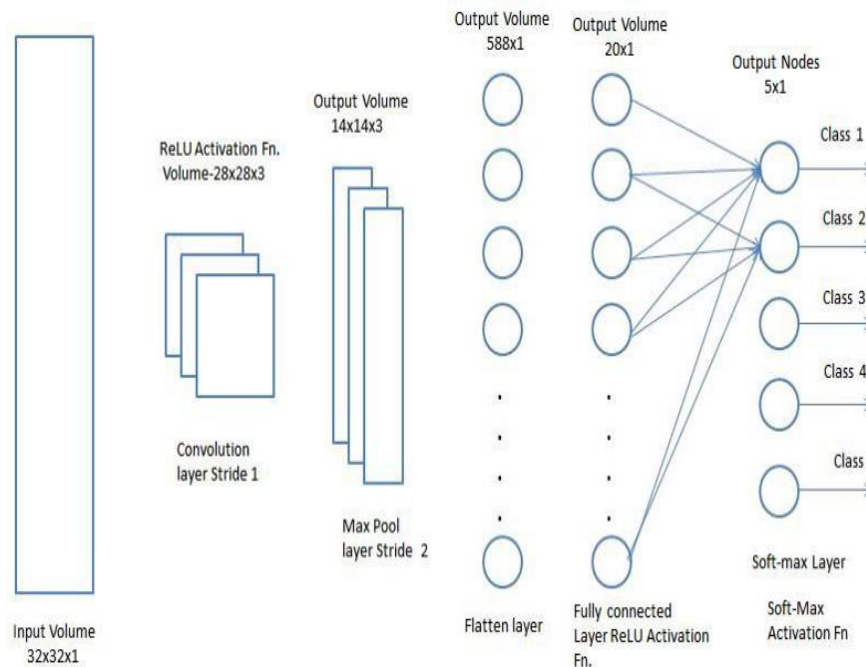


Figure 3.1.3: Fully connected model

3.2 OpenCV:

OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing, video capture, and analysis including features like face detection and object detection.

Computer Vision can be defined as a discipline that explains how to reconstruct, interrupt and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modeling and replicating human vision using computer software and hardware.

Using OpenCV library, we can –

- Read and write images
- Capture and save videos
- Process images (filter, transform)
- Perform feature detection
- Detect specific objects such as faces, eyes, cars, in the videos or images.
- Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

The following are the main library modules of the OpenCV library.

- **Core Functionality:** This module covers the basic data structures such as Scalar, Point, Range, etc., that are used to build OpenCV applications. In addition to these, it also includes the multidimensional array Mat, which is used to store the images.
- **Image Processing:** This module covers various image processing operations such as image filtering, geometrical image transformations, color space conversion, histograms, etc.
- **Video:** This module covers the video analysis concepts such as motion estimation, background subtraction, and object tracking.
- **The video I/O:** This module explains the video capturing and video codecs using OpenCV library.
- **calib3d:** This module includes algorithms regarding basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence and elements of 3D reconstruction.
- **features2d:** This module includes the concepts of feature detection and description.
- **Objdetect:** This module includes the detection of objects and instances of the predefined classes such as faces, eyes, mugs, people, cars, etc.
- **Highgui:** This is an easy-to-use interface with simple UI capabilities.

3.3 Preparing Data:

In this project, we use the MNIST dataset. The MNIST dataset is an acronym that stands for the Modified National Institute of Standards and Technology dataset. The MNIST is a dataset developed by LeCun, Cortes, and Burges for evaluating machine learning models on the handwritten digit classification problem. It has been widely used in research and to design novel handwritten digit recognition systems. The MNIST dataset contains 60,000 training cases and 10,000 test cases of handwritten digits (0 to 9). Each digit is normalized and centered in a gray-scale (0 - 255) image with size 28×28 . Each image consists of 784 pixels that represent the features of the digits. Some examples from the MNIST dataset are shown in Figure 3.3.1. The MNIST dataset is balanced over the ten classes (0 - 9). Figure 3.3.2 shows the percentage of each class in the MNIST dataset.



Figure 3.3.1 Examples of MNIST dataset.

Value	Count	Percent	
1	6742	11.237%	<div></div>
7	6265	10.442%	<div></div>
3	6131	10.218%	<div></div>
2	5958	9.93%	<div></div>
9	5949	9.915%	<div></div>
0	5923	9.872%	<div></div>
6	5918	9.863%	<div></div>
8	5851	9.752%	<div></div>
4	5842	9.737%	<div></div>
5	5421	9.035%	<div></div>

Figure 3.3.2 Class percentages in MNIST dataset.

Chapter 4

Implementation and Verification

4.1 Backend Algorithm

We will basically create an algorithm on CNN using Keras Functional API.

4.1.1 Preparing the Data

We will import TensorFlow and Keras. From Keras, we import various modules that help in building NN layers. Here we are using the MNIST dataset. Filename: (model.py)

```
1  import numpy as np
2  import cv2
3  import matplotlib.pyplot as plt
4  from keras.models import Sequential
5  from keras.layers.core import Dense, Dropout, Activation, Flatten
6  from keras.layers import Conv2D, MaxPool2D
7  from keras import optimizers
8  from keras.datasets import mnist
9  from keras.utils import to_categorical
10 import keras
11
12 # ### loading mist hand written dataset
13 (X_train, y_train), (X_test, y_test) = mnist.load_data()
14 print(X_train.shape, y_train.shape, X_test.shape, y_test.shape)
15
16 # ## Applying threshold for removing noise
17 _,X_train_th = cv2.threshold(X_train,127,255,cv2.THRESH_BINARY)
18 _,X_test_th = cv2.threshold(X_test,127,255,cv2.THRESH_BINARY)
19
20 # ### Reshaping
21 X_train = X_train_th.reshape(-1,28,28,1)
22 X_test = X_test_th.reshape(-1,28,28,1)
23
24 # ### Creating categorical output from 0 to 9
25 y_train = to_categorical(y_train, num_classes = 10)
26 y_test = to_categorical(y_test, num_classes = 10)
```

4.1.2 Defining and Training the models

Here, the model will be defined and trained on the MNIST dataset. After training, it is saved for future work.

```

35 # # Creating CNN model
36 input_shape = (28,28,1)
37 number_of_classes = 10
38
39 model = Sequential()
40 model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',
41               input_shape=input_shape))
42 model.add(Conv2D(64, (3, 3), activation='relu'))
43 model.add(MaxPool2D(pool_size=(2, 2)))
44 model.add(Dropout(0.25))
45 model.add(Flatten())
46 model.add(Dense(128, activation='relu'))
47 model.add(Dropout(0.5))
48 model.add(Dense(number_of_classes, activation='softmax'))
49
50 model.compile(loss=keras.losses.categorical_crossentropy,
51             optimizer=keras.optimizers.Adadelta(),metrics=['accuracy'])
52 model.summary()
53
54 history = model.fit(X_train, y_train,epochs=5, shuffle=True,
55                   batch_size = 200,validation_data= (X_test, y_test))
56
57 model.save('digit_classifier2.h5')

```

4.2 User Interface with PyGame

To make it user-friendly, we build a small app using PyGame in python. by which the user can interface with this model. (Filename: app.py)

```

1  import pygame
2  from process_image import get_output_image
3
4  # pre defined colors, pen radius and font color
5  black = [0, 0, 0]
6  white = [255, 255, 255]
7  red = [255, 0, 0]
8  green = [0, 255, 0]
9  draw_on = False
10 last_pos = (0, 0)
11 color = (255, 128, 0)
12 radius = 7
13 font_size = 500
14
15 #image size
16 width = 640
17 height = 640
18
19 # initializing screen
20 screen = pygame.display.set_mode((width*2, height))
21 screen.fill(white)
22 pygame.font.init()

```

```

26 def show_output_image(img):
27     surf = pygame.pixelcopy.make_surface(img)
28     surf = pygame.transform.rotate(surf, -270)
29     surf = pygame.transform.flip(surf, 0, 1)
30     screen.blit(surf, (width+2, 0))
31
32 def crope(original):
33     cropped = pygame.Surface((width-5, height-5))
34     cropped.blit(original, (0, 0), (0, 0, width-5, height-5))
35     return cropped
36
37 def roundline(srf, color, start, end, radius=1):
38     dx = end[0] - start[0]
39     dy = end[1] - start[1]
40     distance = max(abs(dx), abs(dy))
41     for i in range(distance):
42         x = int(start[0] + float(i) / distance * dx)
43         y = int(start[1] + float(i) / distance * dy)
44         pygame.draw.circle(srf, color, (x, y), radius)
45
46 def draw_partition_line():
47     pygame.draw.line(screen, black, [width, 0], [width,height ], 8)

```

```

50 try:
51     while True:
52         # get all events
53         e = pygame.event.wait()
54         draw_partition_line()
55         # clear screen after right click
56         if(e.type == pygame.MOUSEBUTTONDOWN and e.button == 3):
57             screen.fill(white)
58         # quit
59         if e.type == pygame.QUIT:
60             raise StopIteration
61         # start drawing after left click
62         if(e.type == pygame.MOUSEBUTTONDOWN and e.button != 3):
63             color = black
64             pygame.draw.circle(screen, color, e.pos, radius)
65             draw_on = True
66         # stop drawing after releasing left click
67         if e.type == pygame.MOUSEBUTTONUP and e.button != 3:
68             draw_on = False
69             fname = "out.png"
70             img = crope(screen)
71             pygame.image.save(img, fname)
72             output_img = get_output_image(fname)
73             show_output_image(output_img)
74         # start drawing line on screen if draw is true
75         if e.type == pygame.MOUSEMOTION:
76             if draw_on:
77                 pygame.draw.circle(screen, color, e.pos, radius)
78                 roundline(screen, color, e.pos, last_pos, radius)
79                 last_pos = e.pos
80             pygame.display.flip()
81     except StopIteration:
82         pass
83
84     pygame.quit()

```


4.3 Interface Between UI and Model (Image Processing)

In this file, UI captures the input from the user and saves it as image named **out.png**. After predicted by the model, this image is again shown on the right panel of the UI with the result. (Filename: process_image.py)

```
1  import cv2
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from scipy import ndimage
5  import math
6  from keras.models import load_model
7
8  # loading pre trained model
9  model = load_model('cnn_model/digit_classifier.h5')
10
11 def predict_digit(img):
12     test_image = img.reshape(-1,28,28,1)
13     return np.argmax(model.predict(test_image))
14
15 #pitting label
16 def put_label(t_img,label,x,y):
17     font = cv2.FONT_HERSHEY_SIMPLEX
18     l_x = int(x) - 10
19     l_y = int(y) + 10
20     cv2.rectangle(t_img,(l_x,l_y+5),(l_x+35,l_y-35),(0,255,0),-1)
21     cv2.putText(t_img,str(label),(l_x,l_y), font,1.5,(255,0,0),1,cv2.LINE_AA)
22     return t_img
```

```
24 # refining each digit
25 def image_refiner(gray):
26     org_size = 22
27     img_size = 28
28     rows,cols = gray.shape
29     if rows > cols:
30         factor = org_size/rows
31         rows = org_size
32         cols = int(round(cols*factor))
33     else:
34         factor = org_size/cols
35         cols = org_size
36         rows = int(round(rows*factor))
37     gray = cv2.resize(gray, (cols, rows))
38
39     #get padding
40     colsPadding = (int(math.ceil((img_size-cols)/2.0)),int(math.floor((img_size-cols)/2.0)))
41     rowsPadding = (int(math.ceil((img_size-rows)/2.0)),int(math.floor((img_size-rows)/2.0)))
42
43     #apply apdding
44     gray = np.lib.pad(gray,(rowsPadding,colsPadding),'constant')
45     return gray
```

```

48 def get_output_image(path):
49     img = cv2.imread(path,2)
50     img_org = cv2.imread(path)
51     ret,thresh = cv2.threshold(img,127,255,0)
52     contours,hierarchy = cv2.findContours(thresh, cv2.RETR_CCOMP,
53                                           cv2.CHAIN_APPROX_SIMPLE)
54     for j,cnt in enumerate(contours):
55         epsilon = 0.01*cv2.arcLength(cnt,True)
56         approx = cv2.approxPolyDP(cnt,epsilon,True)
57         hull = cv2.convexHull(cnt)
58         k = cv2.isContourConvex(cnt)
59         x,y,w,h = cv2.boundingRect(cnt)
60         if(hierarchy[0][j][3]!=-1 and w>10 and h>10):
61             #putting boundary on each digit
62             cv2.rectangle(img_org,(x,y),(x+w,y+h),(0,255,0),2)
63             #cropping each image and process
64             roi = img[y:y+h, x:x+w]
65             roi = cv2.bitwise_not(roi)
66             roi = image_refiner(roi)
67             th,fnl = cv2.threshold(roi,127,255,cv2.THRESH_BINARY)
68             pred = predict_digit(roi) # getting prediction of cropped image
69             print(pred)
70             (x,y),radius = cv2.minEnclosingCircle(cnt) # placing label on each digit
71             img_org = put_label(img_org,pred,x,y)
72     return img_org

```

4.4 How to use

- Download or clone this repository (<https://github.com/govindbanura/Hand-Written-Digit-Recognizer-using-Tensorflow-and-Pygame.git>).
- Extract to some location
- First, run **app.py** from the terminal.
- Make sure that you have installed all the required libraries.
- Now, the Pygame window will open. Write any digit in the left panel with the left mouse button.
- To clear the left panel, use the right mouse button.



Fig:4.4.1 GUI

Chapter 5

RESULT ANALYSIS

5.1 Model Accuracy

On testing the model with the different images, we came to know that the model gives the correct answer for 97.23% of the images.

5.2 Integration And Final Working Model

After Properly Integrating python script with GUI these are the results of what we got,

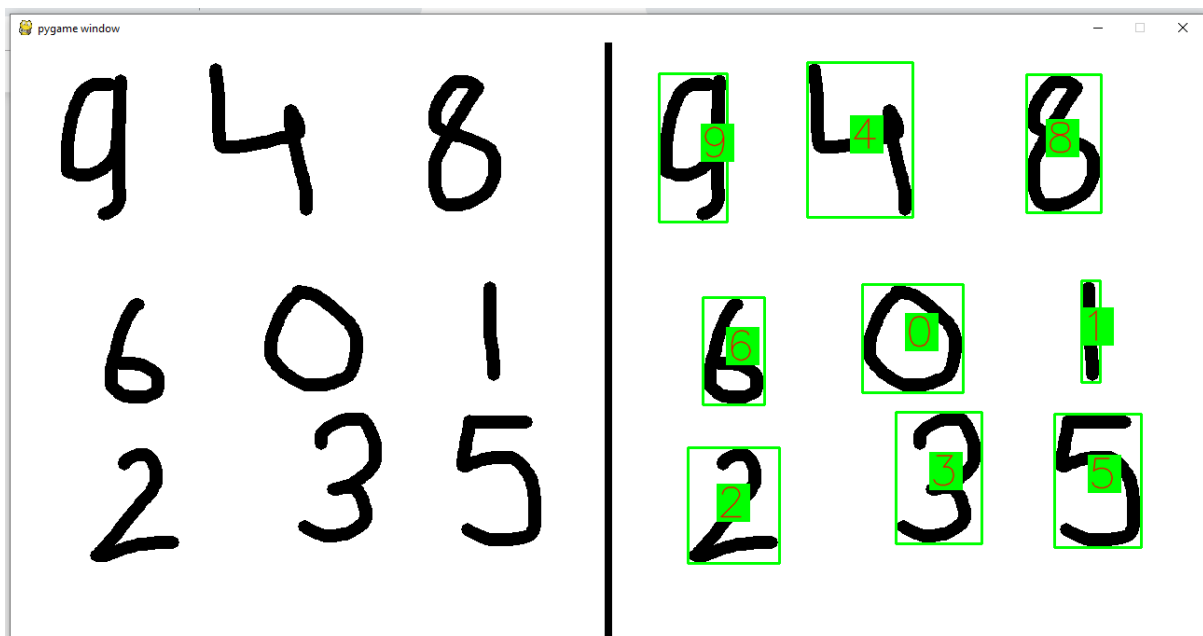


Fig 5.2.1: GUI with the result

Chapter 6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

We have successfully downloaded the data and able to predict the result. In this project, the Handwritten Digit Recognition using Deep learning methods has been implemented. The most widely used Machine learning algorithm for image classification i.e. CNN has been trained by identifying pictures in MNIST handwritten digital databases to predict exactly what the numbers in the pictures are. Utilizing this deep learning technique, a high amount of accuracy can be obtained. Compared to other research methods, this method works better than other models in image processing by more than 99%. Using Keras as backend and Tensorflow as the software, a CNN model is able to give an accuracy of about 97.23%.

6.2 Future Work

- We can make a UI, where we can upload the pictures of handwritten numbers.
- We can build an android application to recognize digits.
- We can use a handwritten alphabets dataset to train the model. It will help in recognizing addresses etc.

References

- <https://medium.com/>
- <https://github.com/>
- <https://stackoverflow.com/>
- <https://www.wikipedia.org/>