

Chapter 1

INTRODUCTION

Deep learning, also known as **deep structured learning** or **hierarchical learning**, is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Learning can be supervised, semi-supervised or unsupervised. Deep learning architectures such as deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design and board game programs, where they have produced results comparable to and in some cases superior to human experts. Deep learning models are vaguely inspired by information processing and communication patterns in biological nervous systems yet have various differences from the structural and functional properties of biological brains, which make them incompatible with neuroscience evidences.

1.1 Definition

Deep learning is a class of machine learning algorithms that:

- use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners [1].
- learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.

1.2 Overview

In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, the raw input may be a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face.

Importantly, a deep learning process can learn which features to optimally place in which level on its own. (Of course, this does not completely obviate the need for hand-tuning; for example, varying numbers of layers and layer sizes can provide different degrees of abstraction). The "deep" in "deep learning" refers to the number of layers through which the data is transformed.

1.3 Artificial Neural Networks

Artificial neural networks (ANNs) or **connectionist systems** are computing systems inspired by the biological neural networks that constitute animal brains [10]. Such systems learn (progressively improve their ability) to do tasks by considering examples, generally without task-specific programming. For example, in image recognition, they might learn to identify images that contain cats by analyzing example images that have been manually labeled as "cat" or "no cat" and using the analytic results to identify cats in other images. They have found most use in applications difficult to express with a traditional computer algorithm using rule-based programming

An ANN is based on a collection of connected units called artificial neurons, (analogous to biological neurons in a biological brain). Each connection (synapse) between neurons can transmit a signal to another neuron. The receiving (postsynaptic) neuron can process the signal(s) and then signal downstream neurons connected to it. Neurons may have state, generally represented by real numbers, typically between 0 and 1. Neurons and synapses may also have a weight that varies as learning proceeds, which can increase or decrease the strength of the signal that it sends downstream. Typically, neurons are organized in layers. Different layers may perform different kinds of transformations on their inputs. Signals travel from the first (input), to the last (output) layer, possibly after traversing the layers multiple times.

The original goal of the neural network approach was to solve problems in the same way that a human brain would. Over time, attention focused on matching specific mental abilities, leading to deviations from biology such as backpropagation, or passing information in the reverse direction and adjusting the network to reflect that information. Neural networks have been used on a variety of tasks, including computer vision, speech recognition, machine translation, social network filtering, playing board and video games and medical diagnosis. As of 2017, neural networks typically have a few thousand to a few million units and millions of connections. Despite this number being several orders of magnitude less than the number of neurons on a human brain, these networks can perform many tasks at a level beyond that of humans (e.g., recognizing faces, playing "Go").

1.4 Deep Neural Networks

A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers [12]. The DNN finds the correct mathematical manipulation to turn the input into the output, whether it be a linear relationship or a non-linear relationship. The network moves through the layers calculating the probability of each output. For example, a DNN that is trained to recognize dog breeds will go over the given image and calculate the probability that the dog in the image is a certain breed. The user can review the results and select which probabilities the network should display (above a certain threshold, etc.) and return the proposed label. Each mathematical manipulation as such is considered a layer, and complex DNN have many layers, hence the name "deep" networks.

DNNs can model complex non-linear relationships. DNN architectures generate compositional models where the object is expressed as a layered composition of primitives. The extra layers enable composition of features from lower layers, potentially modeling complex data with fewer units than a similarly performing shallow network. Deep architectures include many variants of a few basic approaches. Each architecture has found success in specific domains. It is not always possible to compare the performance of multiple architectures, unless they have been evaluated on the same data sets.

DNNs are typically feedforward networks in which data flows from the input layer to the output layer without looping back. At first, the DNN creates a map of virtual neurons and assigns random numerical values, or "weights", to connections between them. The weights and inputs are multiplied and return an output between 0 and 1. If the network didn't accurately recognize a particular pattern, an algorithm would adjust the weights. That way the algorithm can make certain parameters more influential, until it determines the correct mathematical manipulation to fully process the data. Recurrent neural networks (RNNs), in which data can flow in any direction, are used for applications such as language modeling. Long short-term memory is particularly effective for this use. Convolutional deep neural networks (CNNs) are used in computer vision [2]. CNNs also have been applied to acoustic modeling for automatic speech recognition (ASR).

1.5 Applications

- **Automatic speech recognition**

Large-scale automatic speech recognition is the first and most convincing successful case of deep learning [13]. LSTM RNNs can learn "Very Deep Learning" tasks that involve multi-

second intervals containing speech events separated by thousands of discrete time steps, where one time step corresponds to about 10 ms. LSTM with forget gates is competitive with traditional speech recognizers on certain tasks.

The initial success in speech recognition was based on small-scale recognition tasks based on TIMIT. The data set contains 630 speakers from eight major dialects of American English, where each speaker reads 10 sentences. Its small size lets many configurations be tried. More importantly, the TIMIT task concerns phone-sequence recognition, which, unlike word-sequence recognition, allows weak phone bigram language models. This lets the strength of the acoustic modeling aspects of speech recognition be more easily analyzed. The error rates listed below, including these early results and measured as percent phone error rates (PER), have been summarized since 1991. All major commercial speech recognition systems (e.g., Microsoft Cortana, Xbox, Skype Translator, Amazon Alexa, Google Now, Apple Siri, Baidu and iFlyTek voice search, and a range of Nuance speech products, etc.) are based on deep learning.

- **Image recognition**

A common evaluation set for image classification is the MNIST database data set [14]. MNIST is composed of handwritten digits and includes 60,000 training examples and 10,000 test examples. As with TIMIT, its small size lets users test multiple configurations. A comprehensive list of results on this set is available. Deep learning-based image recognition has become "superhuman", producing more accurate results than human contestants. This first occurred in 2011.

- **Visual art processing**

Closely related to the progress that has been made in image recognition is the increasing application of deep learning techniques to various visual art tasks. DNNs have proven themselves capable, for example, of:

- a) identifying the style period of a given painting,
- b) "capturing" the style of a given painting and applying it in a visually pleasing manner to an arbitrary photograph, and
- c) generating striking imagery based on random visual input fields.

- **Natural language processing**

Neural networks have been used for implementing language models since the early 2000s. LSTM helped to improve machine translation and language modeling. Deep neural architectures provide the best results for sentiment analysis, information retrieval, spoken language understanding, machine translation, writing style recognition, text classification and other [3]. Google Translate (GT) uses a large end-to-end long short-term memory network. Google Neural Machine Translation (GNMT) uses an example-based machine translation method in which the system "learns from millions of examples." It translates "whole sentences at a time, rather than pieces. Google Translate supports over one hundred languages. The network encodes the "semantics of the sentence rather than simply memorizing phrase-to-phrase translations". GT uses English as an intermediate between most language pairs.

- **Drug discovery and toxicology**

A large percentage of candidate drugs fail to win regulatory approval. These failures are caused by insufficient efficacy (on-target effect), undesired interactions (off-target effects), or unanticipated toxic effects. Research has explored use of deep learning to predict biomolecular target, off-target and toxic effects of environmental chemicals in nutrients, household products and drugs [15]. AtomNet is a deep learning system for structure-based rational drug design. AtomNet was used to predict novel candidate biomolecules for disease targets such as the Ebola virus and multiple sclerosis.

- **Customer relationship management**

Deep reinforcement learning has been used to approximate the value of possible direct marketing actions, defined in terms of RFM variables. The estimated value function was shown to have a natural interpretation as customer lifetime value.

- **Recommendation systems**

Recommendation systems have used deep learning to extract meaningful features for a latent factor model for content-based music recommendations. Multiview deep learning has been applied for learning user preferences from multiple domains. The model uses a hybrid collaborative and content-based approach and enhances recommendations in multiple tasks.

- **Bioinformatics**

An autoencoder ANN was used in bioinformatics, to predict gene ontology annotations and gene-function relationships. In medical informatics, deep learning was used to predict sleep quality based on data from wearables and predictions of health complications from electronic health record data. Deep learning has also showed efficacy in healthcare [4].

- **Mobile advertising**

Finding the appropriate mobile audience for mobile advertising is always challenging, since many data points must be considered and assimilated before a target segment can be created and used in ad serving by any ad server. Deep learning has been used to interpret large, many-dimensioned advertising datasets. Many data points are collected during the request/serve/click internet advertising cycle. This information can form the basis of machine learning to improve ad selection.

- **Image restoration**

Deep learning has been successfully applied to inverse problems such as denoising, super-resolution, and inpainting. These applications include learning methods such "Shrinkage Fields for Effective Image Restoration" which trains on an image dataset, and Deep Image Prior, which trains on the image that needs restoration.

- **Financial fraud detection**

Deep learning is being successfully applied to anti-money laundering. Deep anti-money laundering detection system can spot and recognize relationships and similarities between data and, further down the road, learn to detect anomalies or classify and predict specific events. The solution leverages both supervised learning techniques, such as the classification of suspicious transactions, and unsupervised learning, e.g. anomaly detection [5].

- **Commercial activity**








Facebook's AI lab performs tasks such as automatically tagging uploaded pictures with the names of the people in them. Google's DeepMind Technologies developed a system capable of learning how to play Atari video games using only pixels as data input. In 2015 they demonstrated their AlphaGo system, which learned the game of Go well enough to beat a professional Go player. Google Translate uses an LSTM to translate between more than 100 languages.



Chapter 2

LITERATURE REVIEW

After a thorough research, about the different frameworks available to support deep learning and the applications they have been designed to be used for, which is summarized in the table below, the Keras framework, with the Tensorflow framework as a backend, has been chosen [6][7].

Table 2.1 Deep learning systems

Name	Languages used	Application domain
	Python, C++, R, CUDA	Handwriting recognition, image classification
	C++ with a Python, MATLAB interface, CUDA	Image classification and segmentation
	Python, C++, CUDA	Natural language processing
	Lua, C and C++, CUDA	Image processing
	C++, CUDA	Networking, Data mining, Image processing.
	Python with R interface, CUDA	Image classification, text generation and summarization, tagging, and translation, along with speech and handwriting recognition
	C++ with Python, R, Julia, JavaScript, Scala, Go, Perl, MATLAB interface, CUDA	Imaging, handwriting/speech recognition, forecasting, and NLP.

Name	Languages used	Application domain
DEEPLEARNING4J	Java, C++ with Scala, C, Python, Clojure interface, CUDA	Network intrusion detection and cybersecurity, fraud detection for the financial sector, anomaly detection in industries such as manufacturing, recommender systems in e-commerce and advertising, image recognition, text mining, parts-of-speech tagging, and natural language processing.
	C++ with a Python interface, CUDA	Handling images, handwriting, and speech recognition
PaddlePaddle	Python	Calculating similarities between words, Understanding Sentiment (Predict the emotion expressed in a piece of text as positive or negative), Machine Translation (Translate between different languages)
	Python, CUDA	Framework for PaintsChainer, a service which does automatic colorization of black and white, line only, draft drawings with minimal user input, sentiment analysis, machine translation, speech recognition.

Chapter 3

ANALYSIS

The Number Recognition (NR) System is a web-based application. It provides a user-friendly, interactive Form-Based Interface (FBI) based on HTML5 form elements. The End-user forms to be requested by the browser and the trained DNN are on separate servers. All communication among the servers, uses a socket connection.

3.1 Purpose

NR System is used to obtain the number conveyed by the handwriting present in an uploaded image. The end-user will be able to update the dataset used by the initially trained DNN, in order to improve its learning and output accuracy in the future. This decreases the error rate of the output produced by the DNN. The Admin trains the DNN and maintains the initial datasets, on a server, separate from that storing the End-user forms.

3.2 Scope

NR System has a scope of two levels of users. At the first level, the Admin is allowed to maintain the DNN and the datasets used to train it, relevant and critical, for the application to run. He may also change the structure of the DNN, in order to achieve faster convergence.

At the second level, the end-user is allowed to upload images of handwritten numbers for classification. The End-user may enter the correct label in case of wrong output, to improve the accuracy of the NR System.

This application restricts its scope to being a web-based applications. It can further be extended to images of classes, other than numbers, like faces, animals, vehicles, etc.

3.3 Motivation

The main motivation behind the selection of this project was to design, develop and implement a software application which may be useful for Handwriting Recognition Systems (HRS). Further, it was the motivation to make the application user interface interactive and user-friendly and at the same time challenging to design and develop in PHP and HTML5.

3.4 Software Requirement Specifications

A **Software Requirements Specification** (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements and may include a set of use cases that describe user interactions that the software must provide.

3.4.1 Overall Description

The proposed system has been designed to support the users to have seem-less, interactive experience. The application is open to all users. Each user can interact with the application. The system supports the functionality of browsing the local file system, for the image to be uploaded.

3.4.2 Functionality

There are two user-access levels

- Admin
- End-user

3.4.2.1 Security Requirements

It is of utmost importance to ensure that there is protection against unauthorized access to corrupt the datasets. The type of image is validated by the server holding the requested user forms. The Admin on the other hand has privileged access in order to view both datasets and ensure seem less and efficient experience to all of its users.

3.4.2.2 Performance Requirement

The PCs used must be at least be INTEL CORE i3 machines so that they can give optimum performance of the product. In addition to these requirements, the system should also embrace the following requirements:-

- **Reliability:** The system should have little or no downtime.
- **Ease of Use:** The general and administrative views should be easy to use and intuitive.

3.4.2.3 Design Constraints

The designers must design the datasets is such a way that any change in the information of a client should be updated and saved effectively in the datasets on the corresponding server.

3.4.2.4 Hardware Requirement

- 64-bit processor
- Internet Connectivity 100 Mbps
- 8 GB memory

3.4.2.5 Software Requirement

1. Web browser
2. WAMP server
3. Operating System: Windows 10
4. Numpy library
5. OpenCV library
6. Tensorflow library
7. Keras library

3.4.2.6 Interface Requirement

The interface which is provided in this software allows users to obtain the number conveyed by the handwriting present in an uploaded image, and the administrator to train the DNN and track all the additions to the datasets. The user form designed should be very easy to use and user friendly.

- End-user can view the output of the DNN.
- The output is followed by, a form to enter the correct label if incorrect.
- End-user can also make use of the HTML button functionality to search for an image.

Communication between the servers is through a socket connection. WAMP server is communicated with, by the browser, with a HTTP GET request

Chapter 4

SYSTEM DESIGN

Systems design is the process of defining the architecture, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development. There is some overlap with the disciplines of systems analysis, systems architecture and systems engineering.

4.1 The MNIST Database

The **MNIST**(Modified National Institute of Standards and Technology) **database** is a large database of handwritten digits that is commonly used for training and testing various image processing systems in the field of machine learning. It was created by "re-mixing" the samples from NIST's original datasets. The creators felt that since NIST's training dataset was taken from American Census Bureau employees, while the testing dataset was taken from American high school students, it was not well-suited for machine learning experiments. Furthermore, the black and white images from NIST were normalized to fit into a 28x28 pixel bounding box and anti-aliased, which introduced grayscale levels.

The MNIST database contains 60,000 training images and 10,000 testing images. Half of the training set and half of the test set were taken from NIST's training dataset, while the other half of the training set and the other half of the test set were taken from NIST's testing dataset. There have been a number of scientific papers on attempts to achieve the lowest error rate; one paper, using a hierarchical system of convolutional neural networks, manages to get an error rate on the MNIST database of 0.23 percent. The original creators of the database keep a list of some of the methods tested on it. In their original paper, they use a support vector machine to get an error rate of 0.8 percent. An extended dataset similar to MNIST called EMNIST has been published in 2017, which contains 240,000 training images, and 40,000 testing images of handwritten digits. The set of images in the MNIST database 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.

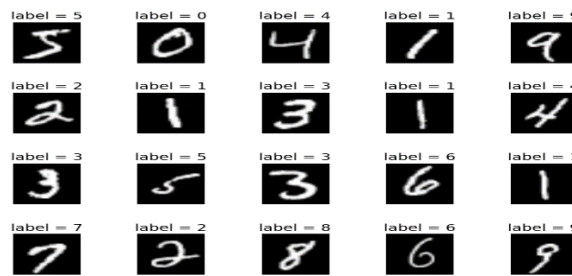


Figure 4.1 Sample images from MNIST test dataset.

4.2 The Tensorflow Framework

TensorFlow is an open-source software library for dataflow programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google, often replacing its closed-source predecessor, DistBelief. TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open source license on November 9, 2015.

4.3 The Keras Framework

Keras is an open source neural network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit or Theano. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer.

In 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library. Chollet explained that Keras was conceived to be an interface rather than a standalone machine-learning framework. It offers a higher-level, more intuitive set of abstractions that make it easy to develop deep learning models regardless of the computational backend used. Microsoft added a CNTK backend to Keras as well, available as of CNTK v2.0. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier. Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep learning

models on clusters of Graphics Processing Units (GPU). Keras has over 200,000 users as of November 2017. Keras was the 10th most cited tool in the KD Nuggets 2018 software poll and registered a 22% usage.

4.4 Training an Artificial Neural Network

Once a network has been structured for a particular application, that network is ready to be trained. To start this process the initial weights are chosen randomly. Then, the training, or learning, begins. There are two approaches to training - supervised and unsupervised. Supervised training involves a mechanism of providing the network with the desired output either by manually "grading" the network's performance or by providing the desired outputs with the inputs. Unsupervised training is where the network has to make sense of the inputs without outside help. The vast bulk of networks utilize supervised training. Unsupervised training is used to perform some initial characterization on inputs. However, in the full-blown sense of being truly self-learning, it is still just a shining promise that is not fully understood, does not completely work, and thus is relegated to the lab.

4.4.1 Supervised Training

In supervised training, both the inputs and the outputs are provided. The network then processes the inputs and compares its resulting outputs against the desired outputs. Errors are then propagated back through the system, causing the system to adjust the weights which control the network. This process occurs over and over as the weights are continually tweaked. The set of data which enables the training is called the "training set." During the training of a network the same set of data is processed many times as the connection weights are ever refined. An artificial neural network configures itself with the general statistical trends of the data. Later, it continues to "learn" about other aspects of the data which may be spurious from a general viewpoint. When the system has been correctly trained, and no further learning is needed, the weights can, if desired, be "frozen."

Networks also don't converge if there is not enough data to enable complete learning. Ideally, there should be enough data so that part of the data can be held back as a test. Many layered networks with multiple nodes are capable of memorizing data. To monitor the network to determine if the system is simply memorizing its data in some nonsignificant way, supervised training needs to hold back a set of data to be used to test the system after it has undergone its training. The designer then has to review the input and outputs, the number of layers, the number of elements per layer, the connections between the layers, the summation, transfer, and training

functions, and even the initial weights themselves. Those changes required to create a successful network constitute a process wherein the "art" of neural networking occurs. Another part of the designer's creativity governs the rules of training. There are many laws (algorithms) used to implement the adaptive feedback required to adjust the weights during training. The most common technique is backward-error propagation, more commonly known as back-propagation.

4.4.2 Unsupervised (Adaptive) Training

The other type of training is called unsupervised training. In unsupervised training, the network is provided with inputs but not with desired outputs. The system itself must then decide what features it will use to group the input data. This is often referred to as self-organization or adaption. At present time, the vast bulk of neural network work is in systems with supervised learning. Supervised learning is achieving results.

4.5 Datasets

In machine learning, algorithms that can learn from and make predictions on data work by making data-driven predictions or decisions, through building a mathematical model from input data. The data used to build the final model usually comes from multiple datasets. In particular, three data sets are commonly used in different stages of the creation of the model.

The model is initially fit on a **training dataset**, that is a set of examples used to fit the parameters (e.g. weights of connections between neurons in artificial neural networks) of the model. The model (e.g. a neural network) is trained on the training dataset using a supervised learning method (e.g. gradient descent). In practice, the training dataset often consist of pairs of an input vector and the corresponding *answer* vector or scalar, which is commonly denoted as the *target*. The current model is run with the training dataset and produces a result, which is then compared with the *target*, for each input vector in the training dataset. Based on the result of the comparison and the specific learning algorithm being used, the parameters of the model are adjusted. The model fitting can include both variable selection and parameter estimation.

The fitted model is used to predict the responses for the observations in a second dataset called the **validation dataset**. The validation dataset provides an unbiased evaluation of a model fit on the training dataset while tuning the model's hyperparameters (e.g. the number of hidden units in a neural network). Validation datasets can be used for regularization by early stopping: stop training when the error on the validation dataset increases, as this is a sign of overfitting to the training dataset.

Finally, the **test dataset** is a dataset used to provide an unbiased evaluation of a *final* model fit on the training dataset. A test dataset is a dataset that is independent of the training dataset, but that follows the same probability distribution as the training dataset. If a model fit to the training dataset also fits the test dataset well, minimal overfitting has taken place. A better fitting of the training dataset as opposed to the test dataset usually points to overfitting. A test set is therefore a set of examples used only to assess the performance (i.e. generalization) of a fully specified classifier.

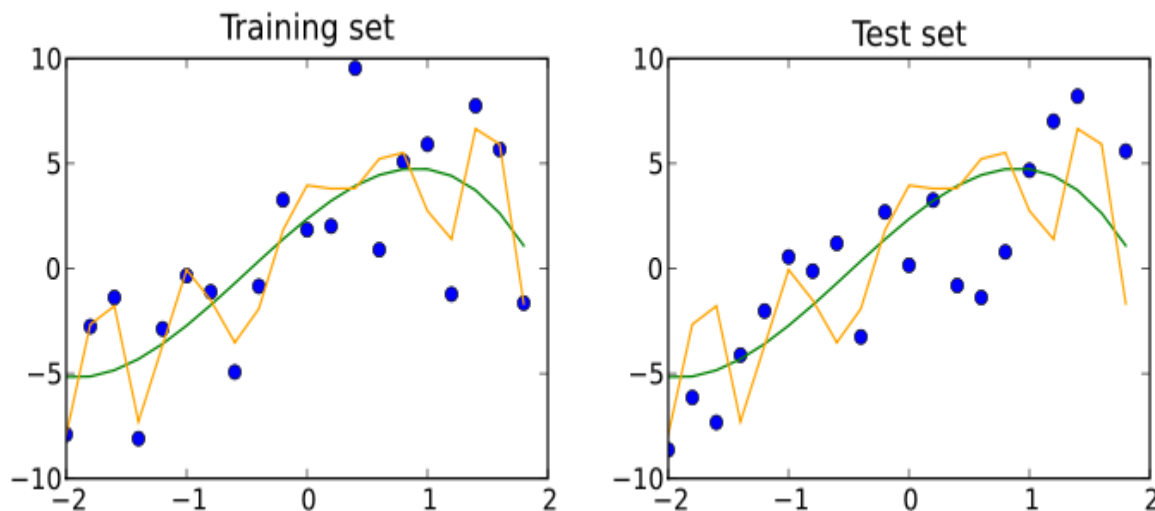


Figure 4.2 Typical graphs for training and test datasets

After training the DNN on MNIST dataset of GRAYSCALE images (white numbers on black background), I have created training and testing datasets of my own, containing images differing in the color of the pixels making the number or their backgrounds or both, as shown in the figures below.

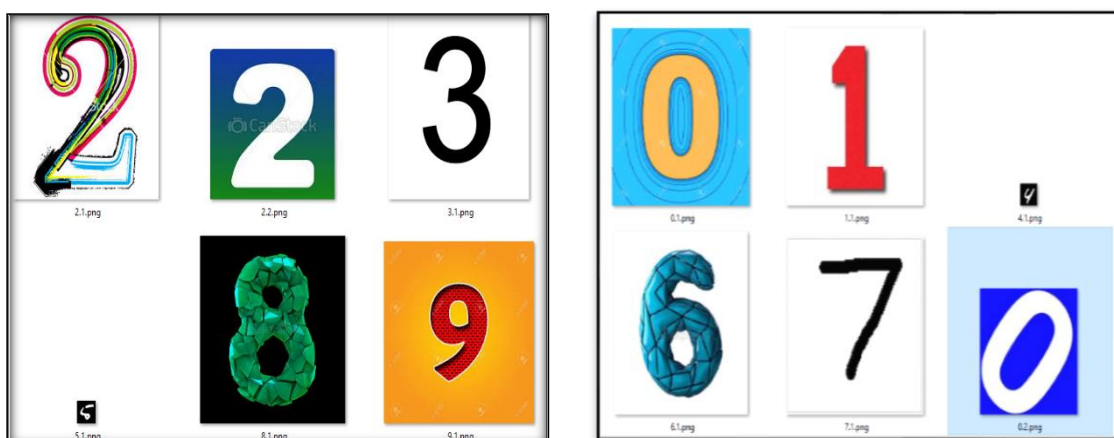


Figure 4.3 Modified Training and Test datasets

Chapter 5

DETAILED DESIGN

Detailed design of the system is the last design activity before implementation begins. The hardest design problems must be addressed by the detailed design or the design is not complete. The detailed design is still an abstraction as compared to source code but should be detailed enough to ensure that translation to source is a precise mapping instead of a rough interpretation.

5.1 Code for training the DNN

```
import keras
from keras.models import Sequential, model_from_json
from keras.layers import Dense
from keras.optimizers import RMSprop
import random
from random import shuffle
import numpy as np
import cv2
from keras.preprocessing import image
#*****
#Paths to folders training and testing datasets (images)
train_d='C:\\ProgramData\\training_images'
test_d='C:\\ProgramData\\testing_images'
#*****
# Obtain label from image file name
def one_hot_label(img):
    label=img.split('.')[0]
    return label
#*****
# Load training images in GRAYSCALE format and reshape into 28x28 2D array of pixels
def train_data_with_label():
    train_images=[]
    for i in os.listdir('C:\\ProgramData\\training_images'):
        path=os.path.join('C:\\ProgramData\\training_images',i)
        img = image.load_img(path=path,grayscale=True,target_size=(28,28))
        img =image.img_to_array(img)
    #*****
```

```

'''Normalize array of pixels, converting all pixel values to
absolute values, 0 (black) or 255 (white)'''
bc=0
wc=0
for x in img:
    for y in x:
        if(y<127):
            bc+=1
        else:
            wc+=1
        if(bc>wc):
            ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
        else:
            ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
    train_images.append([thresh1,one_hot_label(i)])
    shuffle(train_images)
return train_images
#*****

'''Load testing images in GRAYSCALE format and reshape
into 28x28 2D array of pixels'''
def test_data_with_label():
    test_images=[]
    for i in os.listdir('C:\\ProgramData\\testing_images'):
        path=os.path.join('C:\\ProgramData\\testing_images',i)
        img = image.load_img(path=path,grayscale=True,target_size=(28,28))
        img =image.img_to_array(img)
    '''Normalize array of pixels, converting all pixel values to
absolute values, 0 (black) or 255 (white) '''
    bc=0
    wc=0
    for x in img:
        for y in x:
            if(y<127):
                bc+=1
            else:
                wc+=1
            if(bc>wc):
                ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
            else:
                ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
        test_images.append([thresh1,one_hot_label(i)])
        shuffle(test_images)
    return test_images
#*****

train_images = train_data_with_label()
test_images = test_data_with_label()

```

```

# Obtain separate numpy array of 2D arrays for training images
tr_img_data=np.array([i[0] for i in train_images])
# Convert 2D array into 1D array
tr_img_data=tr_img_data.reshape(len(os.listdir(train_d)),784)
# Obtain separate numpy array of training image labels
tr_lbl_data = np.array([str(i[1]) for i in train_images])
# Convert labels to one-hot format
tr_lbl_data=keras.utils.to_categorical(tr_lbl_data,10)
tst_img_data=[]
# Obtain separate numpy array of 2D arrays for testing images
tst_img_data=np.array([i[0] for i in test_images])
# Convert 2D array into 1D array
tst_img_data=tst_img_data.reshape(len(os.listdir(test_d)),784)
# Obtain separate numpy array of testing image labels
tst_lbl_data = np.array([str(i[1]) for i in test_images])
# Convert labels to one-hot format
tst_lbl_data=keras.utils.to_categorical(tst_lbl_data,10)
*****
# Define structure of input, hidden and output layers of DNN
#The input layer of 784 features feeds into a ReLU layer of 512 nodes,
# which then goes into 10 #nodes with softmax applied
model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dense(10, activation='softmax'))
*****
# Compile the models setting values for required parameters
# Other choices for optimizers include Adagrad, SGD, Adam, Adamax, and Nadam
model.compile(loss='categorical_crossentropy',optimizer='rmsprop',
              metrics=['accuracy'])
*****
#Train the DNN on 40 iterations, 100 images at a time and use the testing images
#and labels # for validation of training.
history = model.fit(tr_img_data,tr_lbl_data,batch_size=100,epochs=40,verbose=0,
                    validation_data=(tst_img_data, tst_lbl_data))

# Store the loss and accuracy on completion of training
score=model.evaluate(tst_img_data, tst_lbl_data,verbose=2)
# Save model in JSON format to one file
with open('model_mnist.json', 'w') as f:
    f.write(model.to_json())
f.close()
# Save the model weights in HDF5 format to another file
model.save_weights("weights.h5")
*****
print("Model trained successfully.")
print("Saved model to disk")

```

5.2 Code for testing the DNN on uploaded image

```
import keras
from keras.models import Sequential, model_from_json
from keras.layers import Dense
from keras.optimizers import RMSprop
from keras.preprocessing import image
import cv2
#*****
# Load model from disk
with open('model_mnist.json', 'r') as f:
    model = model_from_json(f.read())
f.close()
#*****
#Load model weights from disk
model.load_weights("weights.h5")
print("Loaded model from disk")
#*****

#Load uploaded image in GRAYSCALE format and reshape into 28x28 2D array of pixels
img = image.load_img(path='x.png', grayscale=True, target_size=(28,28))
img.convert('L')
img = image.img_to_array(img)
# Normalize array of pixels, converting all pixel values to absolute values, 0 (black)
bc=0
wc=0
for x in img:
    for y in x:
        if(y<127):
            bc+=1
        else:
            wc+=1
if(bc>wc):
    ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
else:
    ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
#*****
# Convert 2D array into 1D array
test_img = thresh1.reshape((1,784))
#*****
# Obtain label on uploaded image from trained model
y_pred = model.predict_classes(test_img)
#*****
# Store label in shared resource (text file)
f=open('ans.txt','w')
f.write(('It is the number: %d' % y_pred[0]))
f.close()
```

5.3 Code for adding wrongly classified image to datasets of DNN

```
# Read file with correct label for wrongly classified uploaded image
f=open('label.txt')
l=f.read()
f.close()
# Read file with wrongly classified uploaded image
f=open('x.png','rb')
img=f.read()
f.close()
#*****
# Add image and label (as part of image file name) to training dataset of DNN
train_d='C:\\ProgramData\\training_images'
x=len(os.listdir(train_d))
y=str(l)+'.'+str((x+1))+'.png'
path=os.path.join(train_d,y)
f=open(path,'wb')

f.write(img)
f.close()
#*****
# Add image and label (as part of image file name) to testing dataset of DNN
test_d='C:\\ProgramData\\testing_images'
x=len(os.listdir(test_d))
y=str(l)+'.'+str((x+1))+'.png'
path=os.path.join(test_d,y)
f=open(path,'wb')
f.write(img)
f.close()
```

Chapter 6

IMPLEMENTATION

Implementation is the process of defining how the system should be built, ensuring that it is operational and meets quality standards. It is a systematic and structured approach for effectively integrating a software-based service or component into the requirements of end users.

6.1 Selection of the programming language

6.1.1 Frontend – PHP and WAMP server

PHP, which stands for **Hypertext Preprocessor**, is a server-side scripting language designed for Web development, but also used as a general-purpose programming language [9]. PHP code may be embedded into HTML code, or it can be used in combination with various web template systems, web content management systems, and web frameworks.

PHP code is usually processed by a PHP interpreter implemented as a module in the web server or as a Common Gateway Interface (CGI) executable. The web server combines the results of the interpreted and executed PHP code, which may be any type of data, including images, with the generated web page. PHP code may also be executed with a command-line interface (CLI) and can be used to implement standalone graphical applications.

WampServer refers to a software stack for the Microsoft Windows operating system, created by Romain Bourdon and consisting of the Apache web server, OpenSSL for SSL support, MySQL database and PHP programming language.

The **Apache HTTP Server**, colloquially called **Apache**, is a free and open-source cross-platform web server, released under the terms of Apache License 2.0. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. The Apache HTTP Server is cross-platform with Version 2.0 improved support for non-Unix operating systems such as Windows.

6.1.2 Backend – Python

Python is an interpreted high-level programming language for general-purpose programming and has a design philosophy that emphasizes code readability, notably using significant whitespace [8]. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution. Rather than having all of its functionality built into its core, Python was designed to be highly extensible.

6.2 Socket Programming

A **network socket** is an internal endpoint for sending or receiving data within a node on a computer network. Concretely, it is a representation of this endpoint in networking software (protocol stack), such as an entry in a table (listing communication protocol, destination, status, etc.), and is a form of system resource. The term *socket* is communication between two nodes through a channel being visualized as a cable with two connectors plugging into sockets at each node. Similarly, the term *port* is used for *external* endpoints at a node, and the term *socket* is also used for an internal endpoint of local inter-process communication (IPC) (not over a network).

A process can refer to a socket using a *socket descriptor*, a type of handle. A process first requests that the protocol stack create a socket, and the stack returns a descriptor to the process so it can identify the socket. The process then passes the descriptor back to the protocol stack when it wishes to send or receive data using this socket. Unlike ports, sockets are specific to one node; they are local resources and cannot be referred to directly by other nodes. Further, sockets are not necessarily associated with a persistent connection (channel) for communication between two nodes, nor is there necessarily some single other endpoint. However, in practice for internet communication, sockets are generally used to connect to a specific endpoint and often with a persistent connection.

Sockets are assumed to be associated with a specific **socket address**, namely the IP address and a port number for the local node, and there is a corresponding socket address at the foreign node (other node), which itself has an associated socket, used by the foreign process. Associating

a socket with a socket address is called *binding*. While a local process can communicate with a foreign process by sending or receiving data to or from a foreign *socket address*, it does not have access to the foreign *socket* itself, nor can it use the foreign *socket descriptor*, as these are both internal to the foreign node.

The application programming interface (API) that programs use to communicate with the protocol stack, using network sockets, is called a **socket API**. Development of application programs that utilize this API is called socket programming or network programming. Sockets have functions to read, write, open, and close. In inter-process communication, each end generally has its own socket, but these may use different APIs, abstracted by the network protocol. In the standard Internet protocols TCP and UDP, a **socket address** is the combination of an IP address and a port number. Sockets need not have a source address, for example, for only sending data, but if a program *binds* a socket to a source address, the socket can be used to receive data sent to that address. Based on this address, Internet sockets deliver incoming data packets to the appropriate application process.

6.2.1 Socket implementation using Python's socket library

- `socket()` creates a socket.
- `bind()` binds the socket to a specific IP address and port number.
- `accept()` is used to accept a connection from a client.
- `connect()` is used to establish a connection with a server.
- `send()` is used to send data into the socket.
- `recv()` is used to receive data sent into the socket.
- `listen()` is used to listen on the medium for incoming connections.
- `settimeout()` is used to set the maximum timeout delay, in seconds.
- `close()` is used to close the socket.

6.2.2 Socket implementation in PHP

- `socket_create()` creates a socket.
- `socket_bind()` binds the socket to a specific IP address and port number.
- `socket_accept()` is used to accept a connection from a client.
- `socket_connect()` is used to establish a connection with a server.
- `socket_write()` is used to send data into the socket.
- `socket_read()` is used to receive data sent into the socket.

- `socket_listen()` is used to listen on the medium for incoming connections.
- `set_time_limit()` is used to set the maximum timeout delay, in seconds.
- `socket_close()` is used to close the socket.

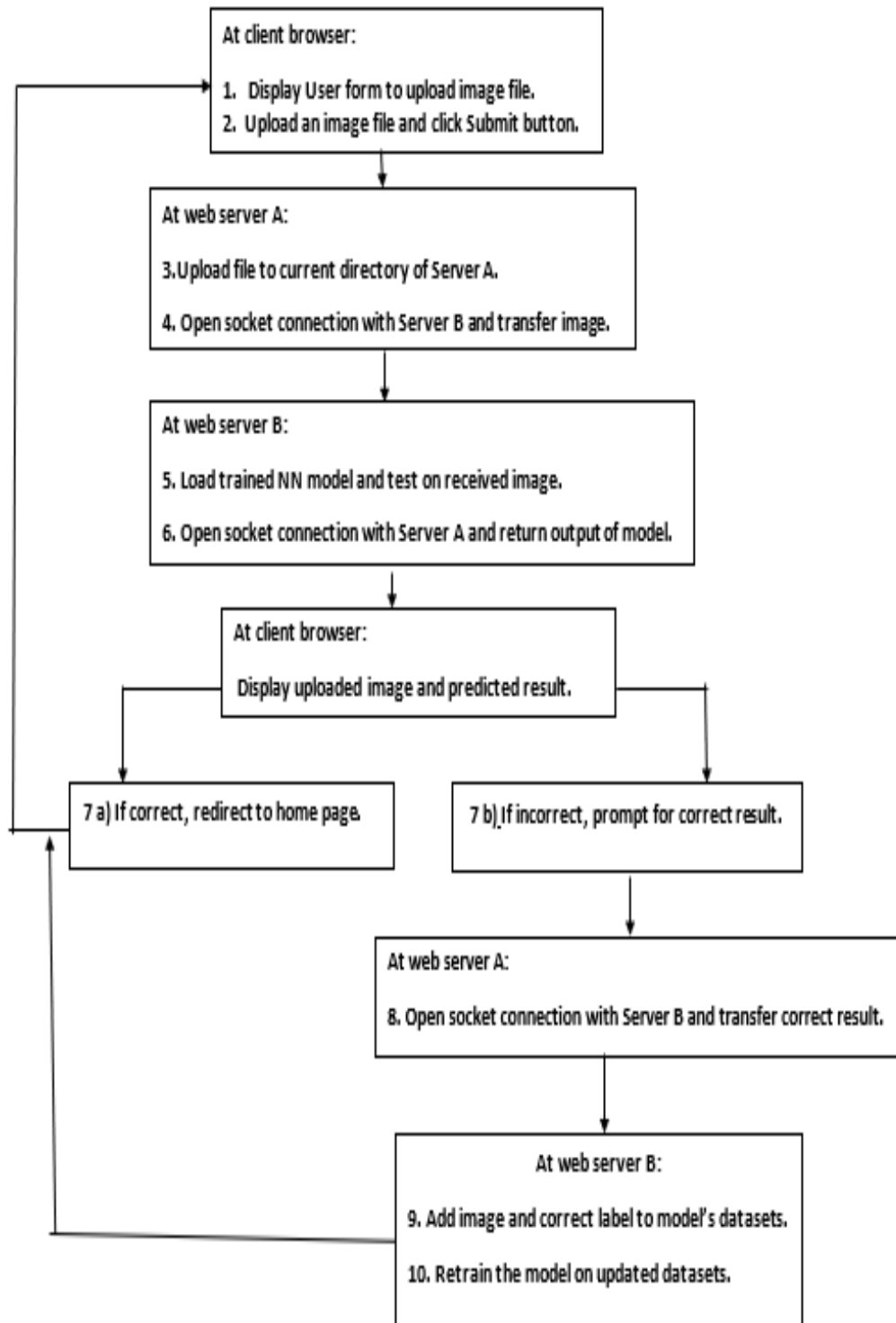


Figure 6.1 System flowchart

6.3 Programming Coding Guidelines

Following Code guidelines are important to programmers for a number of reasons:

- 80% of the lifetime cost of a piece of software goes to maintenance.
- Hardly any software is maintained for its whole life by the original author.
- Code conventions improve the readability of the software, allowing engineers to understand new code more quickly and thoroughly.
- If you ship your source code as a product, you need to make sure it is as well packaged and clean as any other product you create.

6.3.1 Python and PHP Coding Guidelines

6.3.1.1 Naming Conventions

- Each identifier name follows the conventions for naming identifiers in Python or PHP as required.
- The names given to all variables give the meaning of the entity they represent.
- The names containing more than one word use an underscore to separate the two names. For eg, train_images etc.

6.3.1.2 Button click coding

- As soon as the button is clicked a set of code are executed before forwarding the user to the next module.

6.3.2 Discussion Of Code Segments

6.3.2.1 Indentation

In Python, indentation is used to identify the scope of statement blocks. It is 4 spaces by convention.

6.3.2.2 Including Braces

In PHP, braces should **always** be included when writing code using if, for, while etc. blocks. There are **no exceptions** to this rule, even if the braces could be omitted. Leaving out braces makes code harder to maintain in the future and can also cause bugs that are very difficult to track down. Braces should always be placed on a line on their own; again there are no exceptions to this rule. Braces

should also align properly (use tabs to achieve this) so a closing brace is always in the same column as the corresponding opening brace.

6.3.2.3 Spaces between Tokens

There should always be one space on either side of a token in expressions, statements etc. The only exceptions are commas (which should have one space after, but none before), and in PHP, semi-colons (which should not have spaces on either side if they are at the end of a line, and one space after otherwise). Methods should follow the rules laid out already, i.e. no spaces between the method name and the opening bracket and no space between the brackets and the arguments, but one space between each argument.

6.3.2.4 Control Structures

These include if, if –else, for, while etc. control statements should have one space between the control keyword and the colon. Having them increases readability and decreases the likelihood of logic errors being introduced when new lines are added.

6.3.2.5 Method Calls

Methods should be called with no spaces between the method name, the opening parenthesis, and the first parameter; spaces between commas and each parameter, and no space between the last parameter, the closing parenthesis, which in PHP, is followed by the semicolon.

6.3.2.6 Comments

Complete inline documentation comment blocks (docblocks) must be provided. Non-documentation comments are strongly encouraged. C style comments (`/* */`) and standard C++ comments (`//`) are both fine. In Python, (`#`) denotes single-line comments and (`'''`) denotes multiple-line comments.

Chapter 7

TESTING

Software testing is conducted to provide stakeholders with information about the quality of the software product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Techniques include the process of executing a program or application with the intent of finding software bugs, and verifying that the software product is fit for use.

7.1 Unit Testing

7.1.1 Unit Test Case 1

This test case checks if the DNN training module executed by the Admin at the backend functions properly, as described in Table 7.1. The trained model and its weights are then saved to separate files on disk, as shown in Figure 7.1.

Table 7.1 Unit Test Case for DNN training module Check

Sl No. of test case :	1
Name of test :	Check test
Item / Feature being tested :	DNN Training module
Sample Input :	The command 'python mytrain1.py'
Expected output :	Messages 'Model trained successfully' and 'Saved model to disk'
Actual output :	Messages 'Model trained successfully' and 'Saved model to disk'
Remarks :	Test succeeded

```

C:\ProgramData>python mytrain1.py
Using TensorFlow backend.
2018-08-01 20:30:05.433784: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
Model trained successfully.
Saved model to disk

```

Figure 7.1 Unit Test Case for DNN training module Check

7.2 Integration Testing

7.2.1 Integration Test Case 1

This test case checks what happens when there is a timeout in the socket connection between the servers, even when the image file has been uploaded successfully to WAMP server, as described in Table 7.2, and shown in Figure 7.2 and 7.3.

Table 7.2 Integration test case for Socket connection Check

SI No. of test case:	1
Name of test:	Check test
Item / Feature being tested:	Socket connection
Sample Input:	Attempt to upload image by WAMP server.
Expected output:	Messages 'Your file has been uploaded' and 'Could not connect to server'.
Actual output:	Messages 'Your file has been uploaded' and 'Could not connect to server'.
Remarks:	Test succeeded

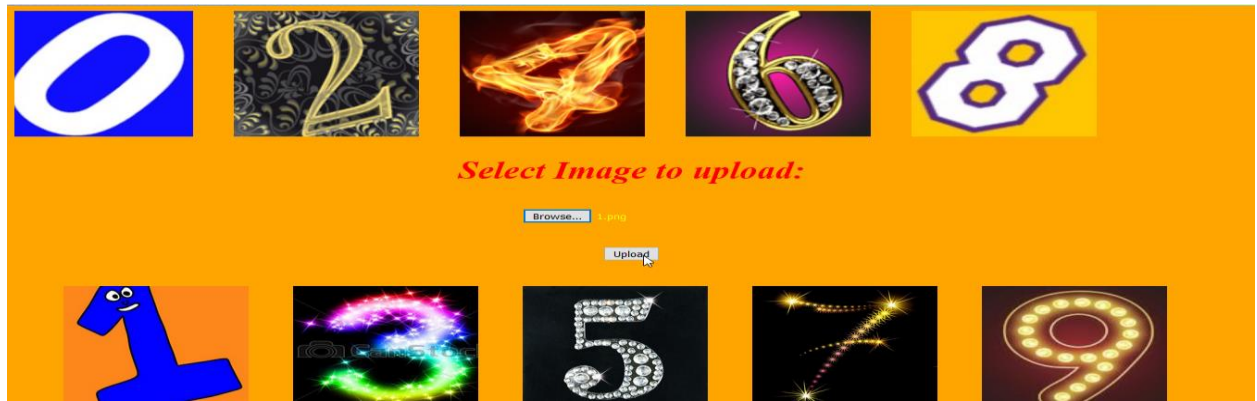


Figure 7.2 Integration test case for Socket connection Check

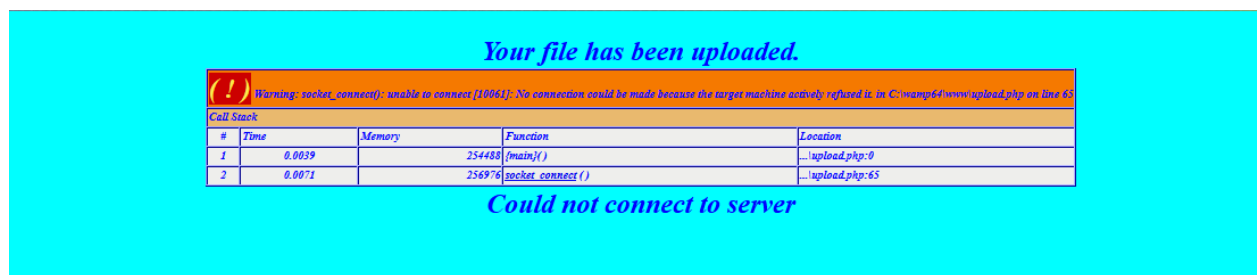


Figure 7.3 Integration test case for Socket connection Error

7.2.2 Integration Test Case 2

This test case checks if the DNN training module gets executed at the backend when the output of DNN is incorrect and end-user has entered the correct label for uploaded image, as described in Table 7.3. The re-trained model and its weights are then saved to separate files on disk, as shown in Figure 7.4 and 7.5.

Table 7.3 Integration test case for Call to DNN training module Check

SI No. of test case:	2
Name of test:	Check test
Item / Feature being tested:	Call to DNN training module
Sample Input:	Correct label of uploaded image from end-user
Expected output:	Messages 'Model trained successfully' and 'Saved model to disk'
Actual output:	Messages 'Model trained successfully' and 'Saved model to disk'
Remarks:	Test succeeded



Figure 7.4 Integration test case for Call to DNN training module Check

```

C:\ProgramData\python\socket2.py
Connected to 127.0.0.1 on 53554
Using TensorFlow backend.
2018-08-01 20:53:08.715459: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
Loaded model from disk
It is the number: 5
connecting to 127.0.0.1 port 15000
sending "It is the number: 5"
closing socket
client disconnected
Connected to 127.0.0.1 on 53569
Using TensorFlow backend.
2018-08-01 20:56:17.848141: I T:\src\github\tensorflow\tensorflow\core\platform\cpu_feature_guard.cc:140] Your CPU supports instructions that this TensorFlow binary was not compiled to use: AVX2
Model trained successfully.
Saved model to disk

```

Figure 7.5 Integration test case for Call to DNN training module Success

7.3 System Testing

7.3.1 System Test Case 1

This test case checks if the DNN gives the closest match to the number '1' in the uploaded image, even on incorrect classification, as described in Table 7.4 and shown in Figure 7.6.

Table 7.4 System test case for Output Check-Pass

SI No. of test case:	1
Name of test:	Check test
Item / Feature being tested:	Output
Sample Input:	Upload of image of the number '7'. Upon press of SUBMIT button.
Expected output:	'It is the number 7'.
Actual output:	'It is the number 1'.
Remarks:	Test failed



Figure 7.6 System test case for Output Check-Pass

7.3.2 System Test Case 2

This test case checks if the DNN performs the correct classification of the number '8' in the uploaded image, as described in Table 7.5 and shown in Figure 7.7.

Table 7.5 System test case for Output Check-Fail

SI No. of test case:	2
Name of test:	Check test
Item / Feature being tested:	Output
Sample Input:	Upload of image of the number '8'. Upon press of SUBMIT button.
Expected output:	'It is the number 8'.
Actual output:	'It is the number 8'.
Remarks:	Test succeeded

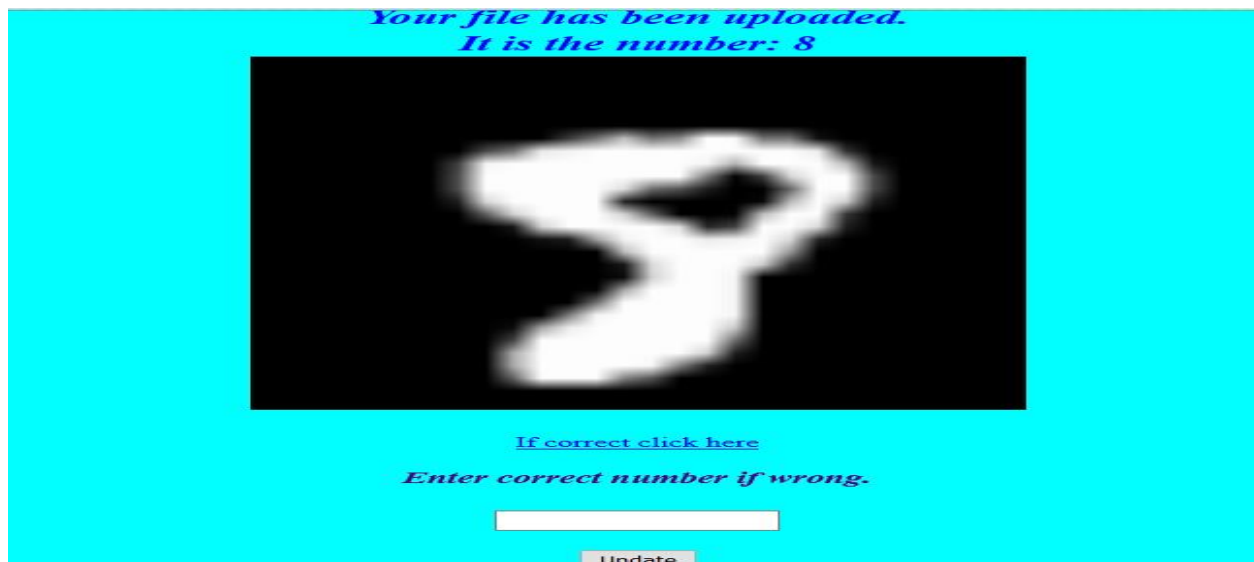


Figure 7.7 System test case for Output Check-Fail

Chapter 8

DISCUSSION OF RESULTS

The outcomes of test results for a variety of input images and user interactions with the application are discussed in the following sections of the chapter.

8.1 Home Page

The end-user can use the Browse button to search his file system for an image file and the Upload button to upload the image file for recognition and classification, as shown in Figure 8.1.

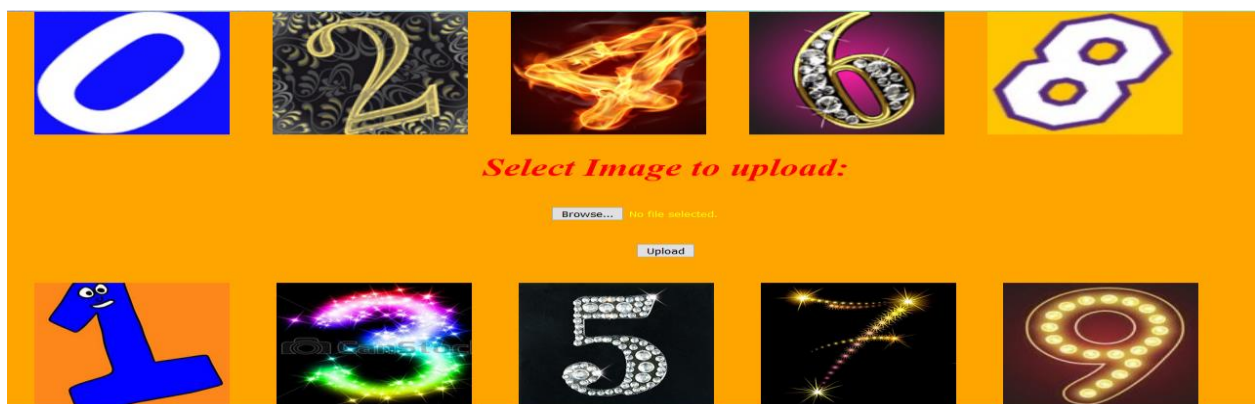


Figure 8.1 Home Page

8.2 Success for White image on Black background

The DNN was able to identify the image of the number '5' written in white on a black background, as shown in Figure 8.2.

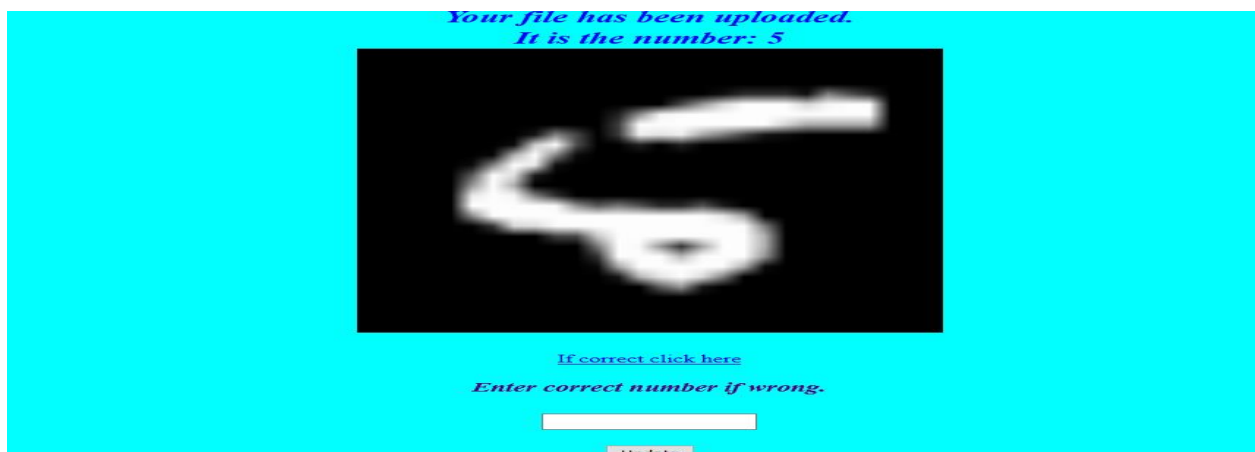


Figure 8.2 Success for White image on Black background.

8.3 Success for Black image on White background

The DNN was able to identify the image of the number '3' written in black on a white background, as shown in Figure 8.3.

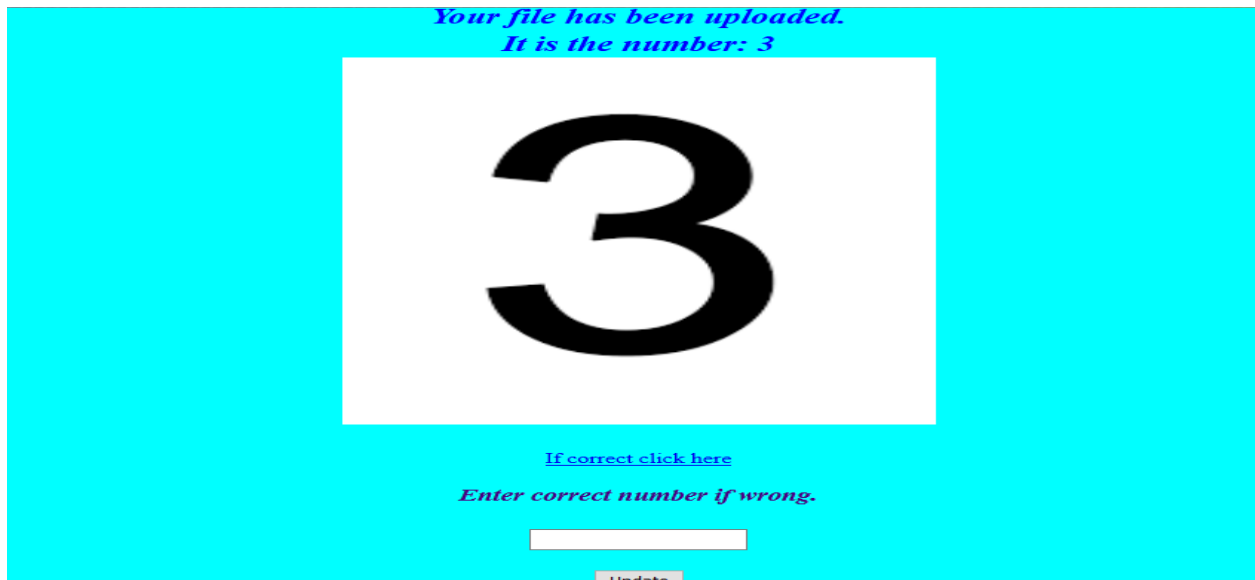


Figure 8.3 Success for Black image on White background

8.4 Success for Colored image on White background

The DNN was able to identify the image of the number '2' written in color on a white background, as shown in Figure 8.4.

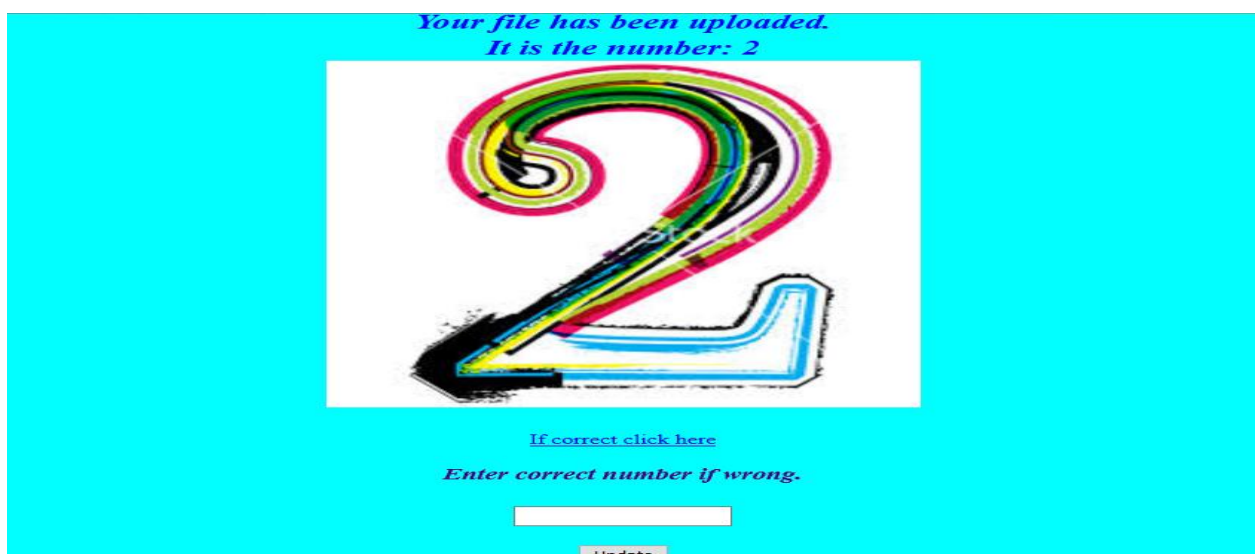


Figure 8.4 Success for Colored image on White background

8.5 Success for Colored image on Black background

The DNN was able to identify the image of the number '8' written in color on a black background, as shown in Figure 8.5.

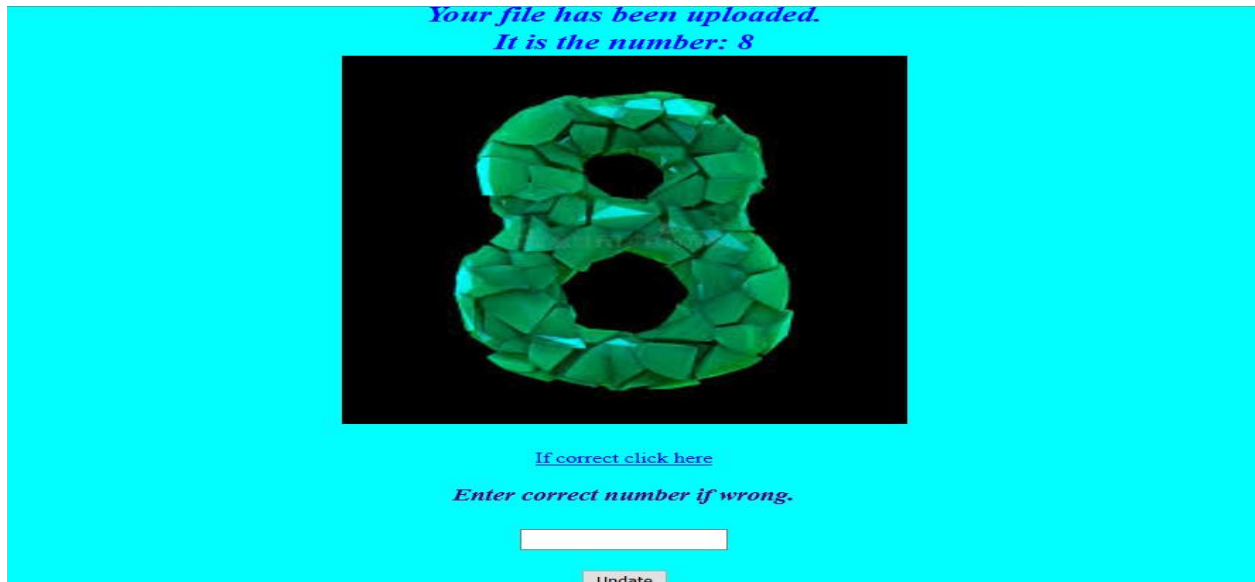


Figure 8.5 Success for Colored image on Black background

8.6 Success for Colored image on Color background

The DNN was able to identify the image of the number '9' written in color on a colored background, as shown in Figure 8.6.

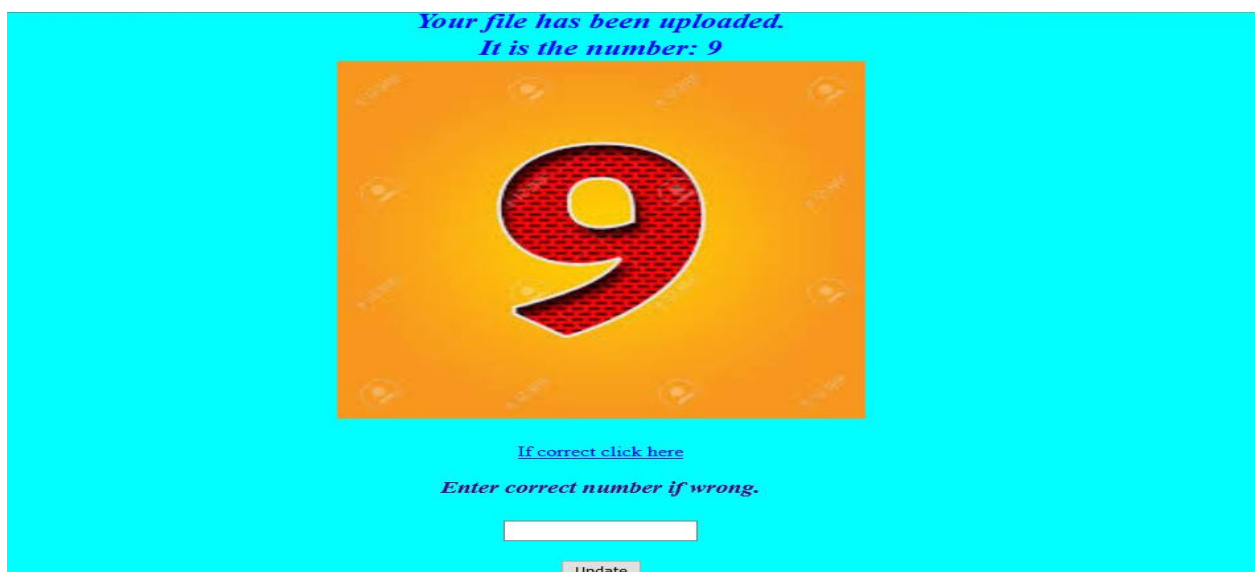


Figure 8.6 Success for Colored image on Color background

8.7 Success for White image on Color background

The DNN was able to identify the image of the number '2' written in white on a colored background, as shown in Figure 8.7.



Figure 8.7 Success for White image on Color background

8.8 Failure for White image on Black background

The DNN incorrectly identified the image of the number '4' written in white on a black background as that of the number '9', as shown in Figure 8.8.

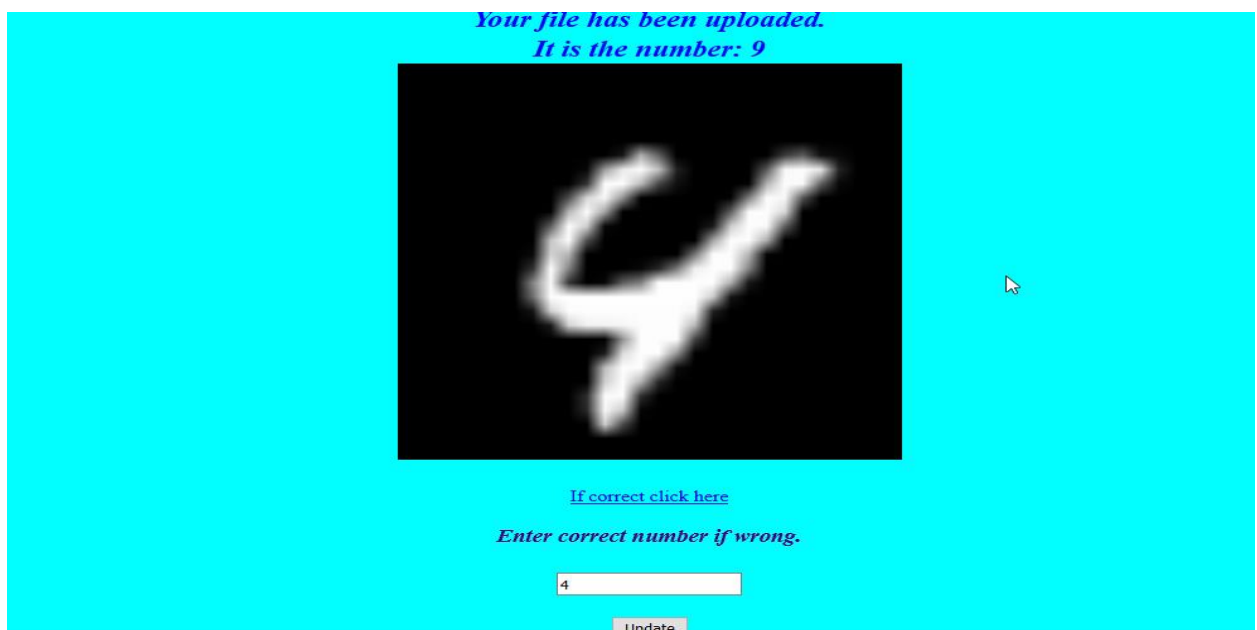


Figure 8.8 Failure for White image on Black background

8.9 Failure for Black image on White background

The DNN incorrectly identified the image of the number '7' written in black on a white background as that of the number '2', as shown in Figure 8.9.

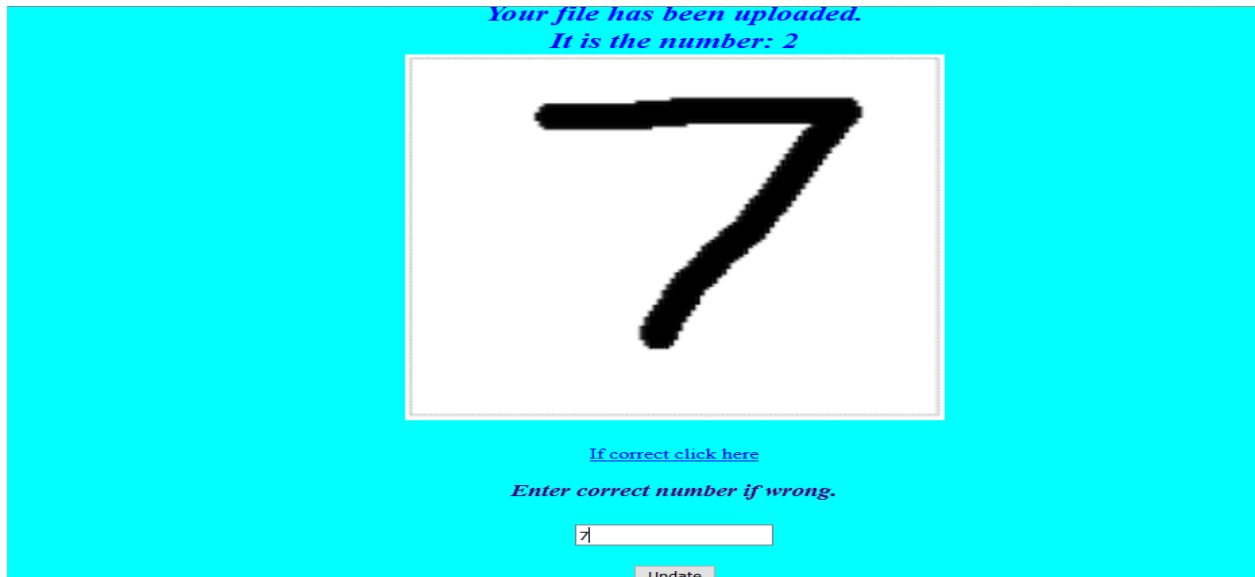


Figure 8.9 Failure for Black image on White background

8.10 Failure for Color image on Black background

The DNN incorrectly identified the image of the number '1' written in color on a black background as that of the number '7', as shown in Figure 8.10.

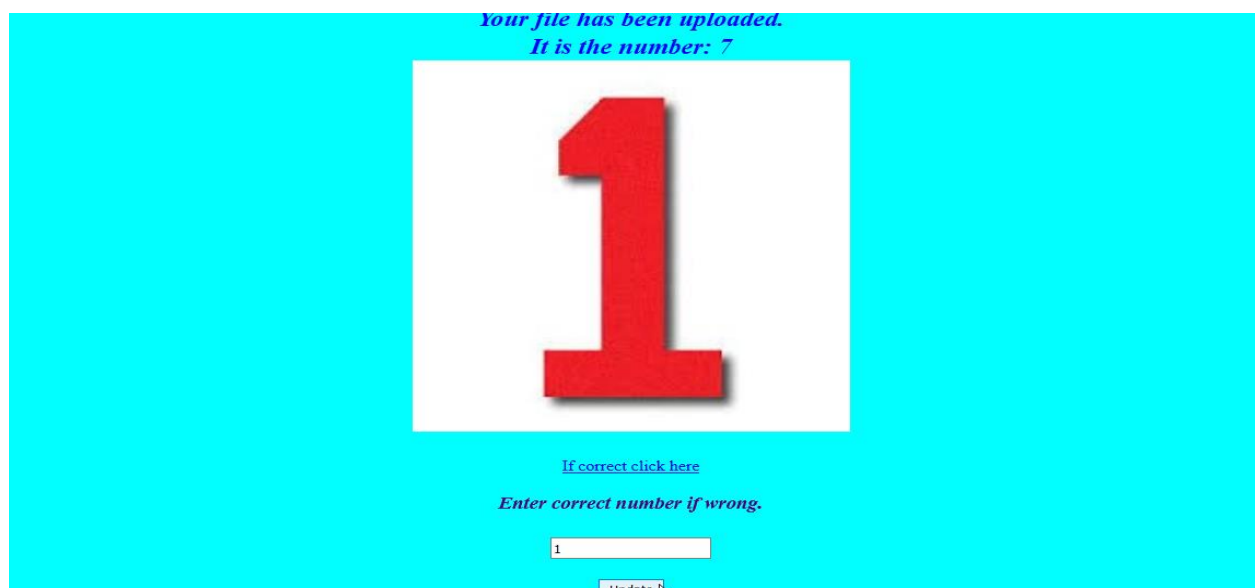


Figure 8.10 Failure for Color image on Black background

8.11 Failure for Color image on White background

The DNN incorrectly identified the image of the number '6' written in color on a white background as that of the number '5', as shown in Figure 8.11.

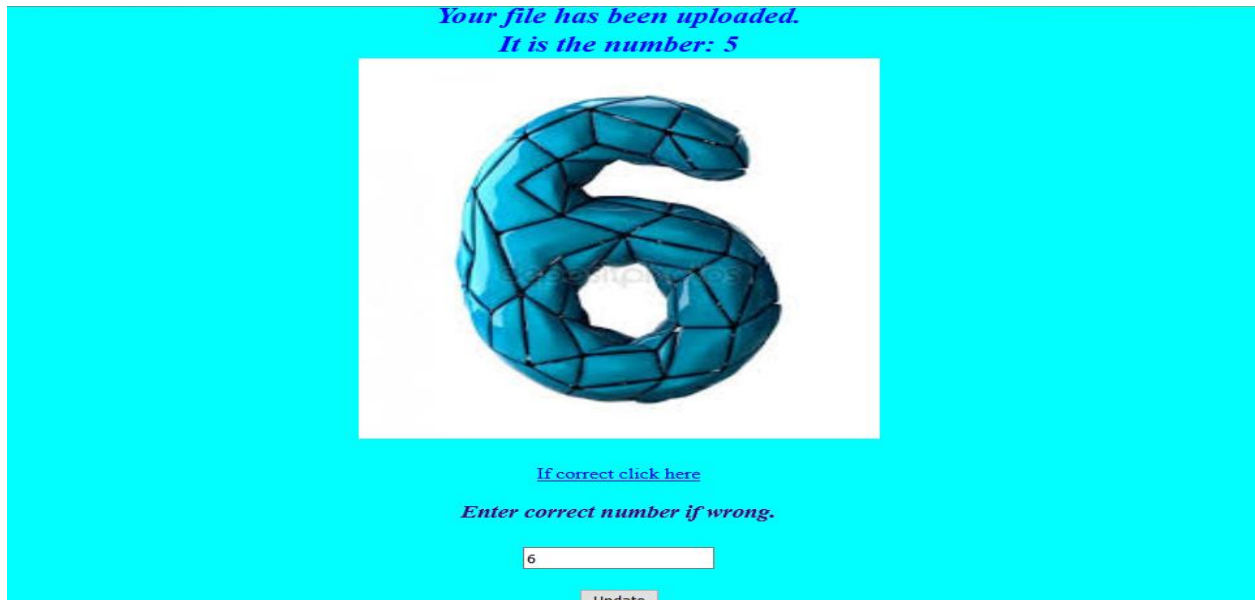


Figure 8.11 Failure for Color image on White background

8.12 Failure for Colored image on Color background

The DNN incorrectly identified the image of the number '0' written in color on a colored background as that of the number '5', as shown in Figure 8.12.

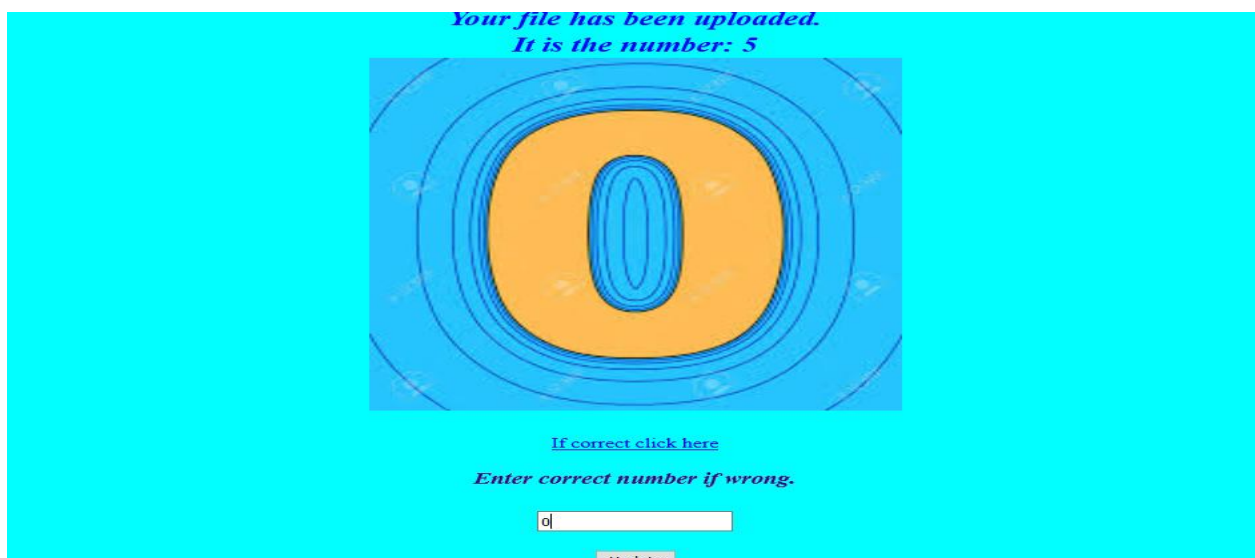


Figure 8.12 Failure for Colored image on Color background

8.13 Failure for White image on Color background

The DNN incorrectly identified the image of the number '0' written in white on a colored background as that of the number '5', as shown in Figure 8.13.

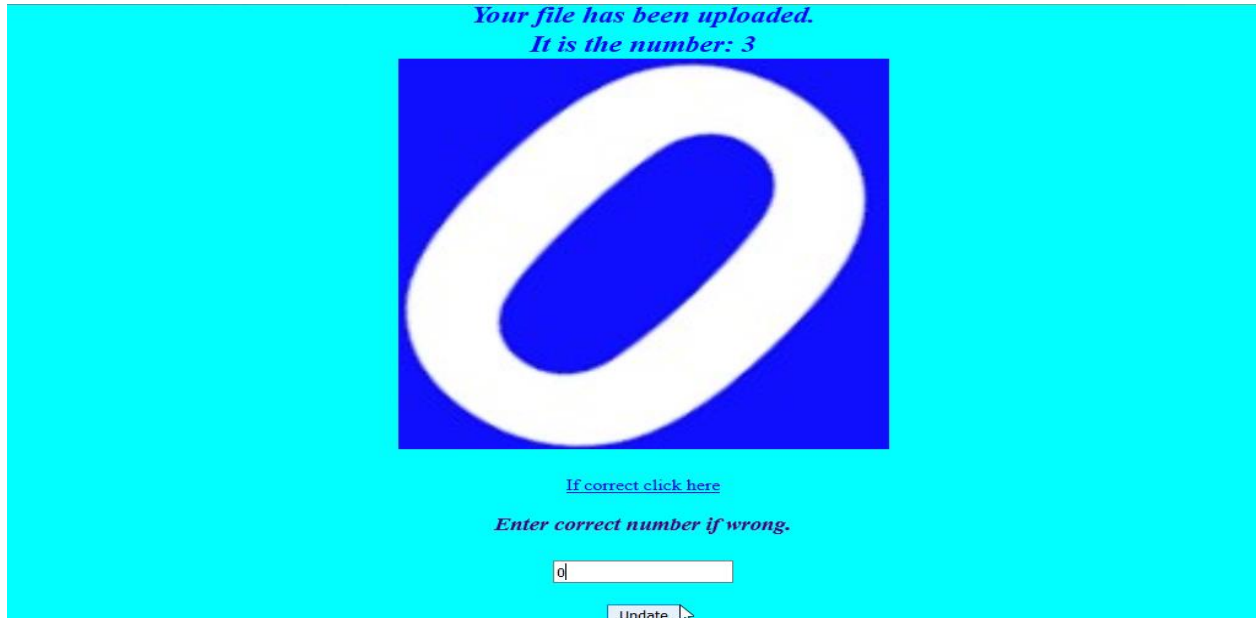


Figure 8.13 Failure for White image on Color background

Chapter 9

CONCLUSION AND FUTURE WORK

The NR System focuses on providing the end-user, the ability to obtain the number conveyed by the handwriting present in an uploaded image. The application has successfully been designed and implemented in Python and PHP languages, using the Keras framework, with the Tensorflow framework as a backend, along with the OpenCV library. The application can be used to add new images to the datasets of the DNN, in case of incorrect classification at first, to improve its learning and accuracy rates.

The application can be given images that differ in the color of the pixels making the number or their backgrounds or both. The Keras library makes it simpler to develop a model of the DNN, providing the abstraction of a high-level API. The system is tested and re-tested with different kinds of images to ensure its effectiveness and provide error free functionality to the end user.

This application restricts its scope to be a web-based applications. It can further be extended to images of classes, other than numbers, like human faces, animals, vehicles, etc. It can be modified to work with medical scan images (eg. MRI's). It can also be enhanced to work with images extracted from frames of a video and perform multiple facial recognitions on each frame, at the same time (eg. Identifying people from security camera footage).

REFERENCES

- [1] Ertam and G. Aydın, "Data classification with deep learning using Tensorflow," *2017 International Conference on Computer Science and Engineering (UBMK)*, Antalya, 2017, pp. 755-758. doi: 10.1109/UBMK.2017.8093521
- [2] L. Yuan, Z. Qu, Y. Zhao, H. Zhang and Q. Nian, "A convolutional neural network based on TensorFlow for face recognition," *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chongqing, 2017, pp. 525-529
- [3] Hongsuk Yi, HeeJin Jung and Sanghoon Bae, "Deep Neural Networks for traffic flow prediction," *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*, Jeju, 2017, pp. 328-331. doi: 10.1109/BIGCOMP.2017.7881687
- [4] D. Medved, P. Nugues and J. Nilsson, "Predicting the outcome for patients in a heart transplantation queue using deep learning", *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, Seogwipo, 2017, pp. 74-77
- [5] P. Vidnerova and R. Neruda, "Evolving keras architectures for sensor data analysis," *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*, Prague, 2017, pp. 109-112. doi: 10.15439/2017F241
- [6] <https://keras.io/>
- [7] <https://www.tensorflow.org/>
- [8] <https://www.python.org/>
- [9] <http://www.php.net/>
- [10] <https://www.w3schools.com/>
- [11] <https://opencv.org/>
- [12] <https://stackoverflow.com/>
- [13] <https://www.quora.com/>
- [14] <https://github.com/>
- [15] <https://www.udemy.com/>

ABBREVIATIONS

ANN	Artificial Neural Network
DNN	Deep Neural Network
LSTM	Long short-term memory
MNIST	Modified National Institute of Standards and Technology
NR	Number Recognition
PHP	Hypertext Preprocessor
WAMP	Windows, Apache, MySQL and PHP
FBI	Form-based Interface
ReLU	Rectified Linear Unit
JSON	JavaScript Object Notation

ABSTRACT

The Number Recognition (NR) System is a web-based application. It provides a user-friendly, interactive Form-Based Interface (FBI) based on HTML5 form elements. The End-user forms to be requested by the browser and the trained DNN are on separate servers. All communication among the servers, uses a socket connection.

NR System is used to obtain the number conveyed by the handwriting present in an uploaded image. The end-user will be able to update the dataset used by the initially trained DNN, in order to improve its learning and output accuracy in the future. This decreases the error rate of the output produced by the DNN. The Admin trains the DNN and maintains the initial datasets, on a server, which is separate from that storing the End-user forms.

NR System has a scope of two levels of users. At the first level, the Admin is allowed to maintain the DNN and the datasets used to train it, relevant and critical, for the application to run. The Admin may also change the structure of the DNN, in order to achieve faster convergence. At the second level, the end-user is allowed to upload images of handwritten numbers for classification. The End-user may enter the correct label in case of wrong output, to improve the accuracy of the NR System.

The application has successfully been designed and implemented in Python and PHP languages, using the Keras framework, with the Tensorflow framework as a backend, along with the OpenCV library. It can be used for adding new images to the datasets of the DNN, in case of incorrect classification at first, to improve its learning and accuracy rates. It can accept images that differ in pixel colors of the numbers or their backgrounds or both. The Keras library makes it easier to develop a model of the DNN, providing the abstraction of a high-level API. The system has been tested and re-tested using different kinds of images to ensure its effectiveness and provide error free functionality to the end user.

ABOUT THE COMPANY

'Contineo' is a pioneering software platform for implementation and administration of academic autonomy. Organizations embarking on academic autonomy face two challenges: a framework for implementation & administration of the autonomous process itself and academic and innovation within this framework. Contineo executes full academic autonomy culminating in secure, confidential, accurate, efficient and auditable examinations of both the digitized answer script and conventional paper and pen variety. The Contineo IT platform helps organizations to quickly get their autonomous processes under control and provides insightful analytics so they can focus on academic innovation and research. With Contineo, your academic journey can be:

Implement and administer autonomy - Foster academic excellence - Accelerate academic innovation and streamline research activities

Contineo's innovative design philosophy provides users unprecedented power and control over processes through time and context aware dashboards held together in a clean, uncluttered, intuitive interface. Exception based management is intrinsic. Important functions like answer script evaluation have a unique management framework for control of the complex process through analytical visualization. Employing the familiar web 2.0 entry points, Contineo's research collaboration-spaces is an advancement on the crowdsourcing ideas of the internet. It empowers research stakeholders to connect, collaborate, create and share without constraints of physical location.

Founded in 2007, e-Sutra Chronicles Pvt. Ltd is a technology products firm based in Bangalore, India. Our innovative, category creating products for the Indian and international education sector are marketed under the brand name of 'Contineo'. Our vision is to build world class products with Indian relevance and global applicability. Our mission is to accelerate the adoption of academic autonomy in India through IT.

ACKNOWLEDGMENT

At the very onset I would like to place our gratefulness to all those people who helped me in making the Internship a successful one.

Coming up, this internship to be a success was not easy. Apart from the sheer effort, the enlightenment of the very experienced teachers also plays a paramount role because it is they who guided me in the right direction.

I would like to thank the **Management of RNS Institute of Technology** for providing such a healthy environment for the successful completion of internship work.

In this regard, I express sincere gratitude to our Director **Dr. H N Shivashankar** and the Principal **Dr. M K Venkatesha**, for providing us all the facilities.

I am extremely grateful to our own and beloved Professor and Head of Department of Information science and Engineering, **Dr. M V Sudhamani**, for having accepted to patronize me in the right direction with all her wisdom.

I place my heartfelt thanks to **Dr. M V Sudhamani** Professor and HOD, Department of Information Science and Engineering for having guided internship and all the staff members of the department of Information Science and Engineering for helping at all times.

I thank **Mr. Avinash, Chief Architect, e-Sutra Chronicles**, for providing the opportunity to be a part of the Internship program and having guided me to complete the same successfully.

I also thank our internship coordinator **Mr. R Rajkumar**, Assistant Professor, Department of Information Science and Engineering. I would thank my friends for having supported me with all their strength and might. Last but not the least, I thank my parents for supporting and encouraging me throughout. I have made an honest effort in this assignment.

AKHILESH V

