A Major Project Final Report on

MachLearn

A Face Recognition and Digit Recognition Android App

Submitted in Partial Fulfillment of the Requirements for the Degree of

Bachelors of Engineering in Software Engineering

Under Pokhara University

Submitted by:

Mr. Ranjan Shrestha, 15758

Mr. Sabin Katwal, 15766

Mr. Akash Shrestha, 15777

Mr. Shree Krishna Yadav, 15789

Under the Supervision of:

(Asst. Prof. Roshan Kumar Sah)

……………………………..

Date:

21st Nov, 2019

Department of Software Engineering

Nepal College of Information Technology

Balkumari, Lalitpur

ACKNOWLEDGEMENT

We had taken great efforts in this project. However, it would not have been possible without the kind support and help of many individuals, teachers, friends and organizations. We would like to extend our sincere thanks to all of them. We are highly thankful to Nepal College of Information Technology for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project. We would like to express our deepest gratitude towards our project coordinator **Prof. Dr. Roshan Chitrakar** and supervisor **Asst. Prof. Roshan Kumar Sah** for his kind co-operation, encouragement and supervision which helped us in completion of this project. We would like to express our special thanks to him for giving us this opportunity to explore our idea and implement them as projects.

Our thanks and appreciations also go to our colleague in developing the project and people who have willingly helped us out with their abilities.

Mr. Ranjan Shrestha, 15758

Mr. Sabin Katwal, 15766

Mr. Akash Shrestha, 15777

Mr. Shree Krishna Yadav, 15789

Nepal College of Information Technology

Balkumari, Lalitpur.

21st November 2018

Abstract

The purpose of making this project is to create a digit and face recognition android app where the recognition of digits and faces can be done in a lot more simpler way. Recognition procedures such as face recognition and digit recognition uses techniques to process degraded or blurred images. Our project focuses on identifying and detecting an object, face or digit in a digital image. Face recognition and digit recognition are the fast growing projects of machine learning in today's context. We see images and recognize them using our brain. We can distinguish between a lion and cat easily, read a sign or recognize human's face. But it is difficult to find solutions like above using a computer: they look easy as human brains are incredibly good at recognizing images.

The machine learning field has made a great progress on addressing these difficult problems in the last few years. We have found that a kind of model called deep convolutional neural network (a neural network that uses the copy of the same neurons) can achieve reasonable performance on hard visual tasks of recognition which can match and exceed human performance in some domains.

Our project is looking forward to target the machine learning concepts using python, save its weights in a text file and convert it to an android app which is face recognition and digit recognition app. The application can predict the inputted image of a digit, can recognize it, predict it and display the results. It can detect whether there is a face or not in an image. This application can also predict the image among 10 different categories of cifar 10 dataset whether it is a cat or dog. With the additional features, it can also detect the emotions depicted in the inputted image.

Keywords: Face recognition, digit recognition, android app, machine learning, and deep convolutional neural network.

List of Tables

List of Figures

Table of Contents

1 INTRODUCTION

Our project is an android app which services to provide the concepts of machine learning and the techniques of supervised learning (classification) to detect faces and recognize digits. By using the tensorflow and keras (python libraries for training, testing and modeling the machine learning projects) , we will be able to load the images of the digits(monochrome image each of 28*28 pixels) and predict it whether it is 0,1,2,3,4,5,6,7,8 or 9. On the other hand, there will be color images of different types based upon which our app can predict whether it's an image of cat or automobile. In other ways it can also detect whether an image contains face or not and it can predict whether a person is sad or happy in the inputted image.

This document provides the scope and context of the project to be undertaken. It also provides a schedule for the completion of the project, including a list of all the deliverables and presentations required.

This application reads the convolutional model made in python and imports the necessary data from it and the images are inputted to the app. The user can select the images and click on predict button and the app shows the predicted data.

With the help of this app one can easily use the latest features of machine learning concepts like face recognition and digit recognition features.

- User can interact with the app using interface of the app.
- User can load and change the images and click predict button to get desired results.
- Programmer can change and update the images in the app.

This project is a complete and full featured application that lets you enjoy machine learning concepts to next level.

1.1 Problem Statement:

The existing system is the system currently in use. Most of the projects are based on traditional recognition of the objects in the present period in most of the cases. This makes working with large data size and complex problems hard to solve easily. So we decided to use deep learning concepts to recognize faces and digits to solve our problem more easily and precisely. The limitations of the existing system are:

- Time consuming.
- Lack of accuracy.
- More manpower to perform real-time jobs.
- Inefficient coding.
- Lack of machine learning concepts.

So our project solves above issues in following ways:

- Based on deep learning concepts for better results.
- Based on new machine learning frameworks like tensorflow and keras.
- More accurate results.

As we know that people can easily identify which digit an image contains, whether it's a cat or dog, whether a person in the image is happy or sad or whether an image contains face or not. But for machines, it is hard to predict which digit an image contains and identify the faces in the images. So we decided to create the project that can solve these problems.

- To recognize faces in the images accurately.
- To recognize objects among 10 categories in the inputted image.
- To recognize digits in the image more precisely.
- To detect emotions in the image.

Using machine learning frameworks like tensorflow and keras, our app can predict the faces, digits and emotions depicted in the images.

1.2 Project Overview:

This application lets you know how machines use convolutional neural network models and predict the digits, faces, etc. depicted in an image. Our model is about 70-80% correct. The model is coded in python and extracted in android studio and shown in apps.

An example of our app is shown as below:-

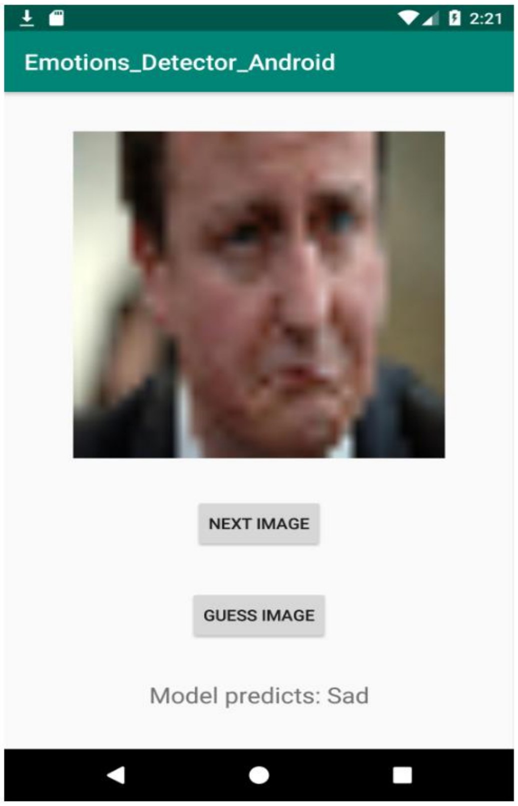
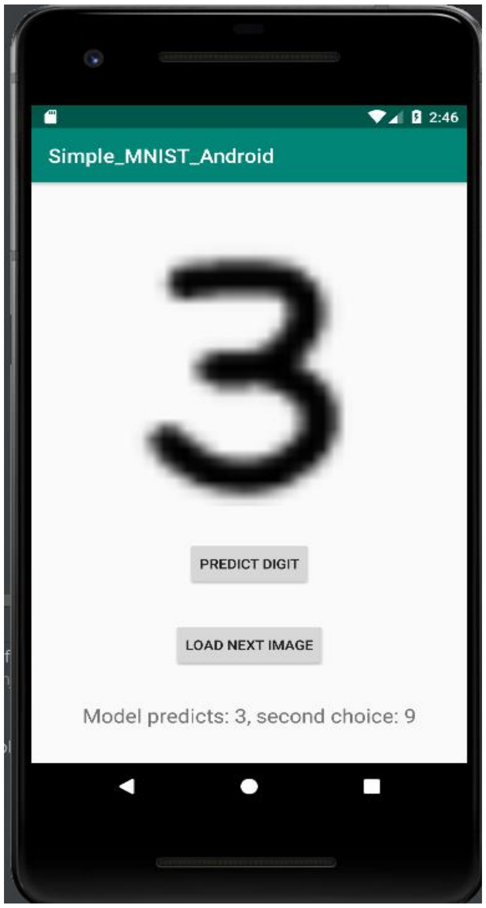


Fig 1: Emotion detection



Model predicts: 3, second choice: 9

Fig 2: Digit Recognition

1.3 Project Objectives:

The objective of this project is to solve user's problem to make machine interpret an inputted image and find out which digit is it, which image is it, does it contain face or not and such thing. The machine does not always produce accurate results as it depends on various factors like number of trains, number of images used for training and more. The app makes certain prediction and displays first prediction and second prediction.

This app is just a simple example of machine learning models. Machine learning helps us to make complex tasks perform easier. We train our machine with large amount of data and make it learn with that data and algorithm. The objective of this project is to make recognition system which includes the following:

- Create a model based on machine learning algorithms for better results.
- Improve recognition process more accurately.
- Introduce deep learning concepts in an android app.
- Create a model based on machine learning algorithms for better results.
- Improve recognition process more accurately.
- Introduce deep learning concepts in an android app.
- To work with low resolution images (28*28 for digits and 32*32 for color images) and predict them accurately as much as we can.
- To identify global trends and analyze available images and data.
- To reduce human efforts for large computations.
- To discover patterns in our data and images.
- To predict digits and images based on machine learning.
- To introduce the concepts of convolutional neural networks in an android app with simple design.

1.4 Project Scope and Limitation:

The scope of the project is to identify the images of cats or automobiles, identify digits and faces. To meet the requirements, we use the simple and basic approach of machine learning concepts. The image is displayed on the screen of the app and the predict option lets the model identify it. The technique can be used in various fields like to take attendance by recognizing faces, to identify handwritten digits and more.

Scope:

- Any user can easily interact with the simple app design.
- Any user can see images and get desired results.

The limitation of this application is that it cannot 100% accurately predict everything but much of the times it is correct.

Limitations:

- The app is not correct every time.

1.5 Significance of Study:

The importance of this application is that it allows us to use the predicted images by the machine to work with various data need to work with our daily activities like we can use it to recognize faces of the person and their emotions and to detect handwritten digits. Machines makes things reliable to work with as hard computational tasks are made easier by them. Our app is a useful app to show machine learning and deep learning concepts.

2 Literature Review

This section consists of the literature study on the image processing system called face and digit recognition system. Our project is looking forward to define all the possible services so that there is an intelligent system of face and digit recognition for the required and general services for all the users. We found various similar products that have already been developed in the market. But we found that our app was far better than others. We provide users with variety of options like face recognition, digit recognition and emotion detection all in one app.

2.1 Review

On comparing our project with other similar apps, we came to conclude the following:

Apps / Features	Image Recognizer	Face Recognition	MachLearn
Ease of Use	Medium	Low	High
User Interface	Yes	yes	yes
Depth in recognition	yes	No	Yes
Clean app design	Yes	No	yes
User friendly	yes	No	Yes
High quality imagery	No	No	No
Focus Group	Public	Public	Public
Password Protected	No	No	No
Accurate function	yes	No	Yes

Table 1: Comparison between apps

Facial recognition is a category of biometric software that maps an individual's facial features mathematically and stores the data as a faceprint to detect the presence of face. Digit recognition is another category which helps us to distinguish between the digits. They play an important role in machine learning capabilities of machines.

2.2 Supervised Learning

Supervised learning as the name indicates the presence of a supervisor as a teacher. Basically supervised learning is a learning in which we teach or train the machine using data which is well labeled that means some data is already tagged with the correct answer. After that, the machine is provided with a new set of examples (data) so that supervised learning algorithm analyses the training data (set of training examples) and produces a correct outcome from labeled data.

2.3 Classification technique

A classification problem is when the output variable is a category, such as "Red" or "blue" or "disease" and "no disease" or "face" or "not a face".

2.4 Convolutional Neural Network (CNN)

When it comes to Machine Learning, Artificial Neural Networks perform really well. Artificial Neural Networks are used in various classification task like image, audio, words. Different types of Neural Networks are used for different purposes, for example for predicting the sequence of words we use Recurrent Neural Networks, similarly for image classification we use Convolution Neural Network. In this blog, we are going to build basic building block for CNN. In a regular Neural Network there are three types of layers:

Input Layers: It's the layer in which we give input to our model. The number of neurons in this layer is equal to total number of features in our data (number of pixels in case of an image).

Hidden Layer: The input from Input layer is then feed into the hidden layer. There can be many hidden layers depending upon our model and data size. Each hidden layers can have different numbers of neurons which are generally greater than the number of features. The output from each layer is computed by matrix multiplication of output of the previous layer with learnable weights of that layer and then by addition of learnable biases followed by activation function which makes the network nonlinear.

Output Layer: The output from the hidden layer is then fed into a logistic function like sigmoid or softmax which converts the output of each class into probability score of each class.

The data is then fed into the model and output from each layer is obtained this step is called feedforward, we then calculate the error using an error function, some common error functions are cross entropy, square loss error etc. After that, we backpropagate into the model by calculating the derivatives. This step is called Backpropagation which basically is used to minimize the loss.

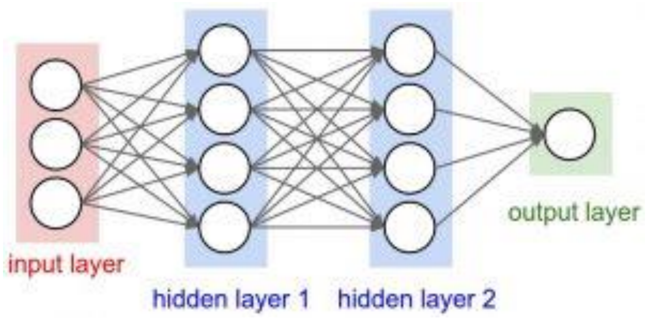


Fig 3: ConvNets

Reference:- https://www.geeksforgeeks.org/introduction-convolution-neural-network/

Image source: cs231n.stanford.edu

Convolution Neural Networks or covnets are neural networks that share their parameters. Imagine you have an image. It can be represented as a cuboid having its length, width (dimension of the image) and height (as image generally have red, green, and blue channels).

Now imagine taking a small patch of this image and running a small neural network on it, with say, k outputs and represent them vertically. Now slide that neural network across the whole image, as a result, we will get another image with different width, height, and depth. Instead of just R, G and B channels now we have more channels but lesser width and height. This operation is called Convolution. If patch size is same as that of the image it will be a regular neural network. Because of this small patch, we have fewer weights.

Now let's talk about a bit of mathematics which is involved in the whole convolution process.

Convolution layers consist of a set of learnable filters. 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 34x34x3. Possible size of filters can be axax3, 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.

Layers used to build ConvNets

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

Types of layers:

Input Layer: This layer holds the raw input of image with width 32, height 32 and depth 3.

Convolution Layer: This layer computes the output volume by computing dot product between all filters and image patch. Suppose we use total 12 filters for this layer we’ll get output volume of dimension 32 x 32 x 12.

Activation Function Layer: This layer will apply 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 x 32 x 12.

Pool Layer: This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents from overfitting. Two common types of pooling layers are **max pooling** and **average pooling**. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

Fully-Connected Layer: This layer is regular neural network layer which 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.

Mathematical Formula for CNN:-

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k]$$

$$p = \frac{f-1}{2}$$

$$[n, n, n_c] * [f, f, n_c] = \left[\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor, \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor, n_f \right]$$

$$\mathbf{Z}^{[l]} = \mathbf{W}^{[l]} \cdot \mathbf{A}^{[l-1]} + \mathbf{b}^{[l]} \qquad \mathbf{A}^{[l]} = g^{[l]}(\mathbf{Z}^{[l]})$$

$$\mathbf{dA}^{[l]} = \frac{\partial L}{\partial \mathbf{A}^{[l]}} \quad \mathbf{dZ}^{[l]} = \frac{\partial L}{\partial \mathbf{Z}^{[l]}} \quad \mathbf{dW}^{[l]} = \frac{\partial L}{\partial \mathbf{W}^{[l]}} \quad \mathbf{db}^{[l]} = \frac{\partial L}{\partial \mathbf{b}^{[l]}}$$

$$\mathbf{dZ}^{[l]} = \mathbf{dA}^{[l]} * g'(\mathbf{Z}^{[l]})$$

$$\mathbf{dA} += \sum_{m=0}^{n_h} \sum_{n=0}^{n_w} \mathbf{W} \cdot \mathbf{dZ}[m,n]$$

2.5 Keras and Tensorflow

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. It was developed with a focus on enabling fast experimentation. Use Keras if you need a deep learning library that:

Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).

Supports both convolutional networks and recurrent networks, as well as combinations of the two.

Runs seamlessly on CPU and GPU.

TensorFlow is an open source library for fast numerical computing. It was created and is maintained by Google and released under the Apache 2.0 open source license. The API is nominally for the Python programming language. Unlike other numerical libraries intended for use in Deep Learning like Theano, TensorFlow was designed for use both in research and development and in production systems. It can run on single CPU systems, GPUs as well as mobile devices and large scale distributed systems of hundreds of machines.

2.6 MNIST and CIFAR 10 dataset

The MNIST database of handwritten digits, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting. Each image is monochrome 28*28 pixels.

The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class. Images include airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck.

2.7 Android

Android, initially developed by Android Inc. and later acquired by Google in 2007, is an open source mobile operating system developed primarily for touch screen mobile devices and tablets. Built upon a modified version of the Linux kernel, Android is currently one of the most used mobile operating systems in the world. In addition to the mobile devices, various other devices like televisions, cars, wrist watches, game consoles and digital cameras use Android today. The UI of android is written in Java, while its core and kernels are written in C and C++. Android 10 is the current version of android.

The functionality of the devices running Android can be extended by installation of several custom applications, commonly called as “apps”, which are primarily written using Java. The applications run on top of Android Runtime and make use of core libraries set up by the application framework. On the foundation lies the Linux kernel, which acts as a bridge between the underlying hardware and the android library.

2.8 Features not provided

- Our project may not be a complete digit and face recognition app as it is only specific to particular features only. It will not able to give all the detailed features of the face and digit recognition system.

3 Methodology

We have planned to work following these methodologies for the application of knowledge, skills, tools and techniques to a broad range of activities in order to meet the requirements of our project.

3.1 Software Development Life Cycle: Machine learning model with incremental model

The framework that we are going to use in this app is machine learning framework that is tensorflow. 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.

For this process, we are planning to use the incremental model of software process model. This model combines linear sequential model with iterative prototype model. The process is repeated following the delivery of each increment until the complete product is produced.

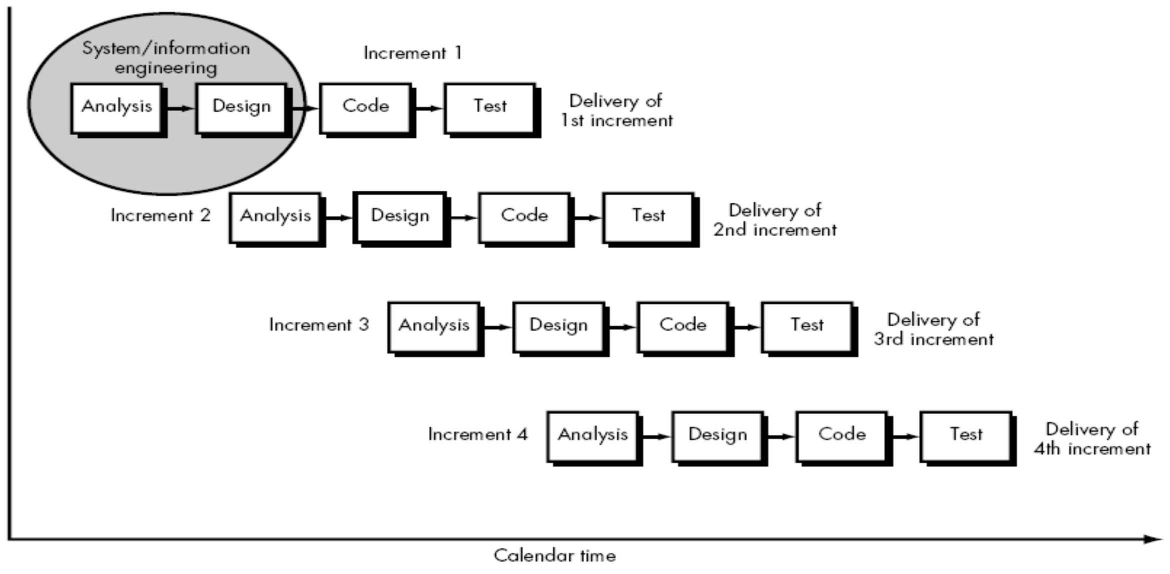


Fig 4: Incremental Model

Reference:- https://www.researchgate.net/figure/Incremental-Model_fig3_311559089

Incremental model includes the following phases:

3.1.1 Analysis Phase

In this phase, the requirements of the software was analyzed which resulted in software requirement specifications.

3.1.2 Design Phase

In this phase, analysis of the SRS was translated into the system's design. Various diagrams like use-case diagram were developed.

3.1.3 Coding Phase

This phase involves the coding as per the design and formation of a working system at the end of the process.

3.1.4 Testing Phase

In this phase, the system was tested. With each testing, certain changes were made as per the suggestion. This was done in an incremental manner until a satisfactory system was made.

3.2 Technologies to be used:

- XML for front-end
- Android(Java code) for front end
- Python (keras and tensorflow framework) for back-end

3.3 Tools to be used:

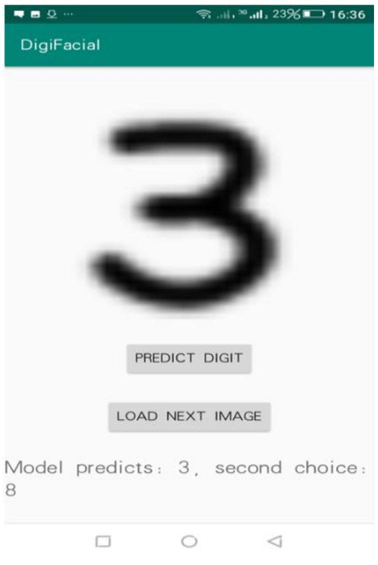
The tools used for documentation, designing and developing UI, testing are listed as below in the table:

TOOLS	PURPOSE
XML	Design
Atom/Pycharm	Text editor for Python code
Android studio	Text editor for Android code
Android Emulator	Testing app real time

Table 2: Tools to be used

4 Work In Progress

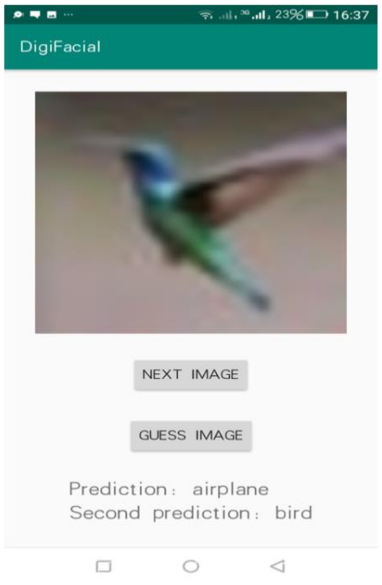
- Digit Recognition



Model predicts: 3, second choice: 8

Fig 5: App predicts 3

- Object Recognition



Prediction: airplane
Second prediction: bird

Fig 6: App predicts bird as second choice

- Face Recognition

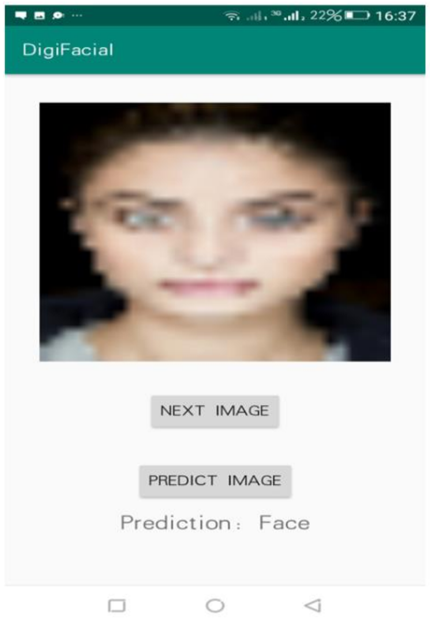


Fig 7: App predicts face

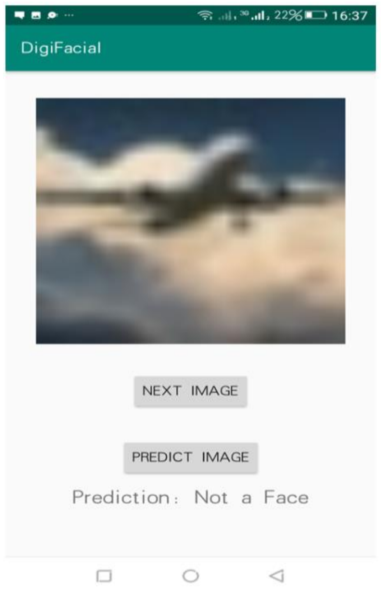
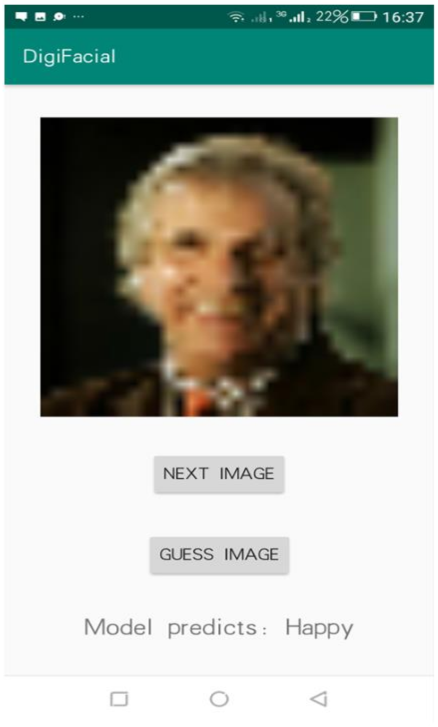


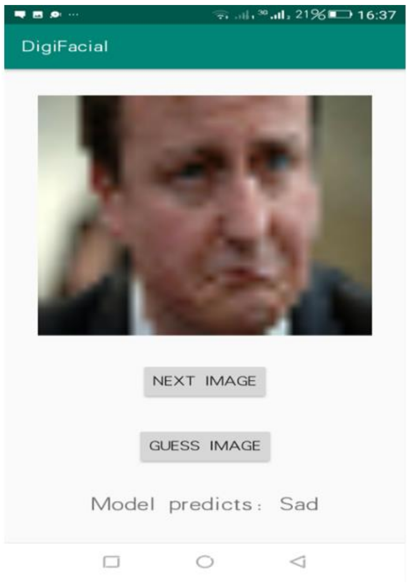
Fig 8: App predicts non-face

- Emotion Detection



Model predicts : Happy

Fig 9: App predicts happy face



Model predicts : Sad

Fig 10: App predicts sad face

5 Codes and Outputs

1) Sample Codes:-

- Digit Recognition code for python (Training model):

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

mnist_data = input_data.read_data_sets('MNIST_data', one_hot=True)
# train_image = mnist_data.train.images[0]
# train_label = mnist_data.train.labels[0]

# print(train_image)
# print(train_label)

# y = Wx + b
x_input = tf.placeholder(dtype=tf.float32, shape=[None, 784], name='x_input')
W = tf.Variable(initial_value=tf.zeros(shape=[784, 10]), name='W')
b = tf.Variable(initial_value=tf.zeros(shape=[10]), name='b')
y_actual = tf.add(x=tf.matmul(a=x_input, b=W, name='matmul'), y=b, name='y_actual')
y_expected = tf.placeholder(dtype=tf.float32, shape=[None, 10], name='y_expected')

cross_entropy_loss = tf.reduce_mean(
    input_tensor=tf.nn.softmax_cross_entropy_with_logits(labels=y_expected,
logits=y_actual),
    name='cross_entropy_loss')

optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.5, name='optimizer')
train_step = optimizer.minimize(loss=cross_entropy_loss, name='train_step')

saver = tf.train.Saver()

session = tf.InteractiveSession()
session.run(tf.global_variables_initializer())

tf.train.write_graph(graph_or_graph_def=session.graph_def,
                     logdir='.',
                     name='mnist_model.pbtxt',
                     as_text=False)

for _ in range(1000):
    batch = mnist_data.train.next_batch(10)
    train_step.run(feed_dict={x_input: batch[0], y_expected: batch[1]})

saver.save(sess=session,
       save_path='../Mnist/mnist_model.ckpt')
```

```
# [0.01 0.10 0 0 0 0 0 0 0 0.9]
correct_prediction = tf.equal(x=tf.argmax(y_actual, 1), y=tf.argmax(y_expected, 1))
accuracy = tf.reduce_mean(tf.cast(x=correct_prediction, dtype=tf.float32))
print(accuracy.eval(feed_dict={x_input: mnist_data.test.images, y_expected:
mnist_data.test.labels}))

print(session.run(fetches=y_actual, feed_dict={x_input: [mnist_data.test.images[0]]}))
```

- Digit Recognition code for python (Freeze Graph):

```
import tensorflow as tf
from tensorflow.python.tools import freeze_graph, optimize_for_inference_lib

freeze_graph.freeze_graph(input_graph='mnist_model.pbtxt',
                          input_saver='',
                          input_binary=True,
                          input_checkpoint='mnist_model.ckpt',
                          output_node_names='y_actual',
                          restore_op_name='save/retore_all',
                          filename_tensor_name='save/Const:0',
                          output_graph='frozen_mnist_model.pb',
                          clear_devices=True,
                          initializer_nodes='')
input_graph_def=tf.GraphDef()
with tf.gfile.Open('frozen_mnist_model.pb', 'rb') as f:
    data = f.read()
    input_graph_def.ParseFromString(data)

output_graph_def=optimize_for_inference_lib.optimize_for_inference(input_graph_def=
input_graph_def,
                                                                  input_node_names=['x_input'],
                                                                  output_node_names=['y_actual'],

placeholder_type_enum=tf.float32.as_datatype_enum)
f = tf.gfile.FastGFile(name='optimized_frozen_mnist_model.pb',
                       mode='w')
f.write(file_content=output_graph_def.SerializeToString())
```

- Facial Recognition code for python (Training model):-

```
from keras.datasets import cifar10
import glob
from PIL import Image
import numpy as np
from keras.models import Sequential
from keras.layers import Reshape, Dense, Flatten, Dropout
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras import backend as K
K.set_image_dim_ordering('th')
K.set_learning_phase(1)
from freeze_graph_script2 import freeze_session
import tensorflow as tf


# Function to load the first 60 images from our cifar 10 data set and use these as the non
face images
def load_non_face_images():
    (_, _), (cifar_10_images, _) = cifar10.load_data()
    cifar_10_images = cifar_10_images[:60]
    cifar_10_images = cifar_10_images.astype('float32') / 255.0
    return cifar_10_images


# Function to load about 60 face images from our Face_Images dir and format them
def load_face_images(dir_name):
    output_images = []
    for image_name in glob.glob(dir_name + '/*'):
        image = Image.open(image_name).resize((32, 32))
        image_data = np.asarray(image, dtype='float32') / 255.0
        output_image_data = np.transpose(image_data, (2, 0, 1))
        output_images.append(output_image_data)
    return np.asarray(output_images)


# Function to divide data into training and testing data and also assign labels
def separate_data(face_images, non_face_images):
    train_images = []
    train_labels = []
    test_images = []
    test_labels = []
    for i in range(10):
        test_images.append(face_images[i])
        test_labels.append([1, 0])
        test_images.append(non_face_images[i])
        test_labels.append([0, 1])
    for i in range(len(face_images) - 10):
        train_images.append(face_images[i + 10])
        train_labels.append([1, 0])
        train_images.append(non_face_images[i + 10])
        train_labels.append([0, 1])
    return np.asarray(train_images), np.asarray(train_labels), np.asarray(test_images),
np.asarray(test_labels)


# Function to flatten image data arrays for easy input into model
def reshape_image(input_array):
    output_array = []
    for image in input_array:
        output_array.append(image.reshape(-1))
    return np.asarray(output_array)


# Function to create a sequential keras model, involves:
# a convolution to map input image to 32 features
# a max pool to pool each 2x2 block to 1x1 block so image reduced from 32x32 to 16x16
# a dense layer to map each of the 32x16 features to a neuron
# a dropout layer to prevent over fitting
# a final dense layer to read all neurons into 2 output neurons ([1, 0] or [0, 1])
def create_model():
    model = Sequential()
    model.add(Reshape((3, 32, 32), input_shape=(3072, )))
    model.add(Conv2D(32, (3, 3), input_shape=(3, 32, 32), padding='same',
activation='relu',
                     kernel_constraint=maxnorm(3)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
    model.add(Dropout(0.5))
    model.add(Dense(2, activation='softmax'))
    return model


# Function to train and asses model accuracy
def train_and_assess_model(model, train_images, train_labels):
    epochs = 100
    l_rate = 0.01
    decay = l_rate / epochs
    optimizer = SGD(lr=l_rate, momentum=0.5, decay=decay, nesterov=False)
    model.compile(loss='categorical_crossentropy', optimizer=optimizer,
metrics=['accuracy'])
    print(model.summary)

    model.fit(train_images, train_labels, validation_data=(train_images, train_labels),
epochs=epochs, batch_size=12)
```

```
    frozen_graph = freeze_session(K.get_session(),
output_names=[model.output.op.name])
    tf.train.write_graph(frozen_graph, '.', 'face_recognition.pb', False)
    print(model.input.op.name)
    print(model.output.op.name)

    scores = model.evaluate(train_images, train_labels, verbose=0)
    print('Accuracy: %.2f%%' % (scores[1] * 100))
    return model


# Function to predict output and compare to actual output
def predict_images(model, test_images, test_labels):
    for i in range(len(test_labels)):
        test_image = np.expand_dims(test_images[i], axis=0)
        print('Predicted label: {}, actual label: {}'.format(model.predict(test_image),
test_labels[i]))


# Function to get the data, format the data, train the model, and predict some test images
def run():
    face_images = load_face_images('Face_Images')
    non_face_images = load_non_face_images()

    train_images, train_labels, test_images, test_labels = separate_data(face_images,
non_face_images)
    train_images = reshape_image(train_images)
    test_images = reshape_image(test_images)

    model = create_model()
    model = train_and_assess_model(model, train_images, train_labels)
    predict_images(model, test_images, test_labels)


# Run the program
run()
```

- Facial Recognition code for python (Freeze Graph):-

```
import tensorflow as tf
from tensorflow.python.framework.graph_util import convert_variables_to_constants


def freeze_session(session, keep_var_names=None, output_names=None,
clear_devices=True):
    graph = session.graph
    with graph.as_default():
        freeze_var_names = list(set(v.op.name for v in
tf.global_variables()).difference(keep_var_names or []))
        output_names = output_names or []
        output_names += [v.op.name for v in tf.global_variables()]
        input_graph_def = graph.as_graph_def()
        if clear_devices:
            for node in input_graph_def.node:
                node.device = ''
        frozen_graph = convert_variables_to_constants(session, input_graph_def,
output_names, freeze_var_names)
    return frozen_graph
```

- Emotion Detection code for python (Training Model):-

```
import tensorflow as tf
import numpy as np
from PIL import Image
import glob
from keras.models import Sequential
from keras.layers import Reshape, Flatten, Dense, Dropout
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.optimizers import SGD
from keras.constraints import maxnorm
from keras import backend as K
K.set_image_dim_ordering('th')
K.set_learning_phase(1)
from Freeze_Graph_Script import freeze_graph


# Function to load all images in a given directory and resize and reformat them
def load_and_format_images(dir_name):
    output_images = []
    for image_name in glob.glob(dir_name + '/*'):
        image = Image.open(image_name).resize((32, 32))
        output_image_data = np.asarray(image, dtype='float32') / 255.0
        output_image_data = np.transpose(output_image_data, (2, 0, 1))
        if output_image_data.shape == (3, 32, 32):
            output_images.append(output_image_data)
    return np.asarray(output_images)


# Function to separate image arrays into training and testing sets and assign labels ([1, 0]
for happy, [0, 1] for sad)
def separate_data(happy_images, sad_images):
    train_images = []
    train_labels = []
    test_images = []
    test_labels = []
    for i in range(5):
        test_images.append(happy_images[i])
        test_labels.append([1, 0])
        test_images.append(sad_images[i])
        test_labels.append([0, 1])
    for i in range(len(happy_images) - 5):
        train_images.append(happy_images[i + 5])
        train_labels.append([1, 0])
        train_images.append(sad_images[i + 5])
        train_labels.append([0, 1])
    return np.asarray(train_images), np.asarray(train_labels), np.asarray(test_images),
np.asarray(test_labels)


# Function to flatten each of the images into a 1 x 3072 element array rather than 3 x 32 x
32 matrix
def reshape_images(input_array):
    output_array = []
    for image in input_array:
        output_array.append(image.reshape(-1))
    return np.asarray(output_array)


# Function to build up the sequential model layer by layer
# Conv and MaxPool layers map pixels to 32 filters and compress the image data
# Dense layers map filters to neurons in a network
# Dropout layer prevents over fitting by getting rid of excess noise
def create_model():
    model = Sequential()
    model.add(Reshape(target_shape=(3, 32, 32), input_shape=(3072, )))
    model.add(Conv2D(32, (2, 2), input_shape=(3, 32, 32), padding='same',
activation='relu',
                     kernel_constraint=maxnorm(3)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(512, activation='relu', kernel_constraint=maxnorm(3)))
    model.add(Dropout(0.5))
    model.add(Dense(2, activation='softmax'))
    return model


# Function to train and test the model
# Train the model with SGD optimizer and categorical crossentropy loss function using
train images and labels
# Assess the model's accuracy using the test images and labels and print out the final
```

```
score
def train_and_assess_model(model, train_images, train_labels, test_images, test_labels):
    epochs = 100
    learning_rate = 0.001
    decay = learning_rate / epochs
    sgd = SGD(lr=learning_rate, momentum=0.5, decay=decay, nesterov=False)
    model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])
    print(model.summary())

    model.fit(train_images, train_labels, validation_data=(train_images, train_labels),
epochs=epochs, batch_size=12)

    print(model.input.op.name)
    print(model.output.op.name)
    frozen_graph = freeze_graph(K.get_session(), output_names=[model.output.op.name])
    tf.train.write_graph(frozen_graph, '.', 'emotions_detector.pb', as_text=False)

    scores = model.evaluate(test_images, test_labels, verbose=0)
    print('Accuracy: %.2f%%' % scores[1] * 100)

    return model


# Function to get the results of our model prediction based on test images and compare to
test labels
def predict_images(model, test_images, test_labels):
    for i in range(len(test_images)):
        test_image = np.expand_dims(test_images[i], axis=0)
        print('Predicted label: {}, actual label: {}'.format(model.predict(test_image),
test_labels[i]))


def run():
    happy_images = load_and_format_images('Happy')
    sad_images = load_and_format_images('Sad')

    train_images, train_labels, test_images, test_labels = separate_data(happy_images,
sad_images)
    train_images = reshape_images(train_images)
    test_images = reshape_images(test_images)

    model = create_model()
    model = train_and_assess_model(model, train_images, train_labels, test_images,
test_labels)
    predict_images(model, test_images, test_labels)
```

```
run()
```

2) Sample outputs:-

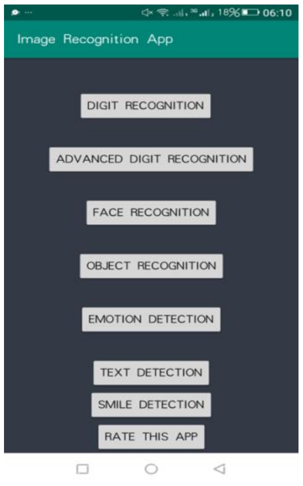
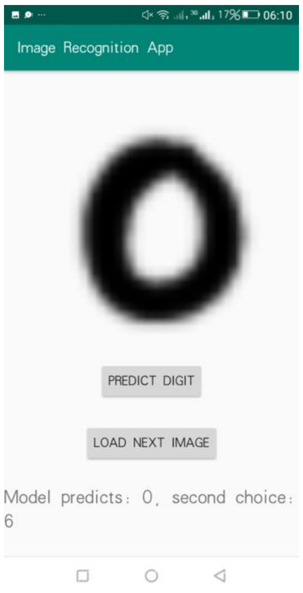
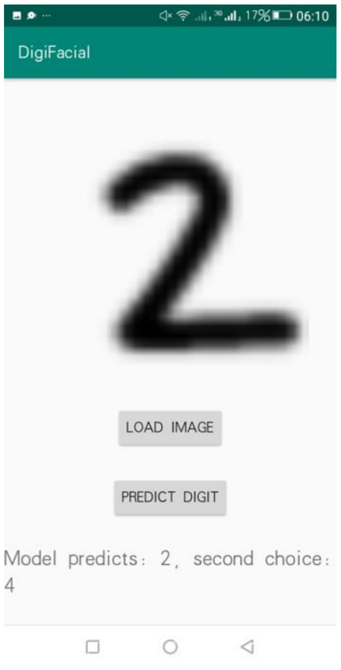


Fig 11: App first page



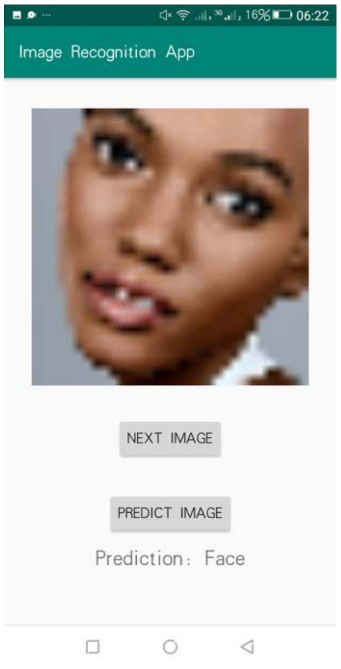
Model predicts: 0, second choice: 6

Fig 12: App predicts 0



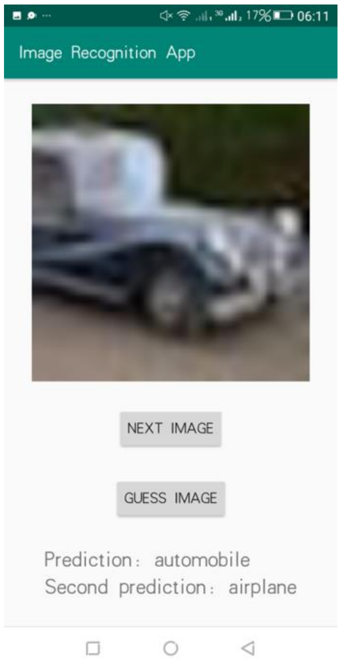
Model predicts: 2, second choice: 4

Fig 13: App predicts 2



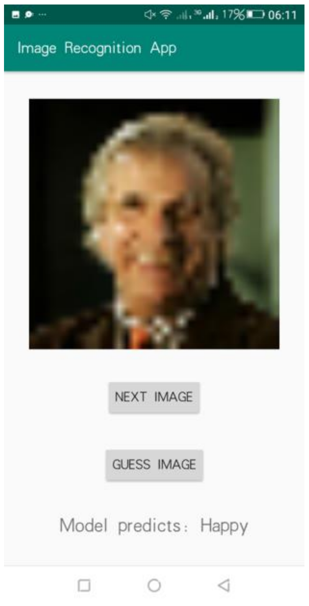
Prediction: Face

Fig 14: App predicts face



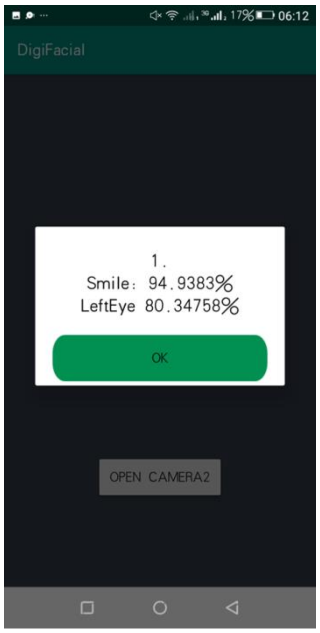
Prediction : automobile
Second prediction : airplane

Fig 15: App predicts automobile



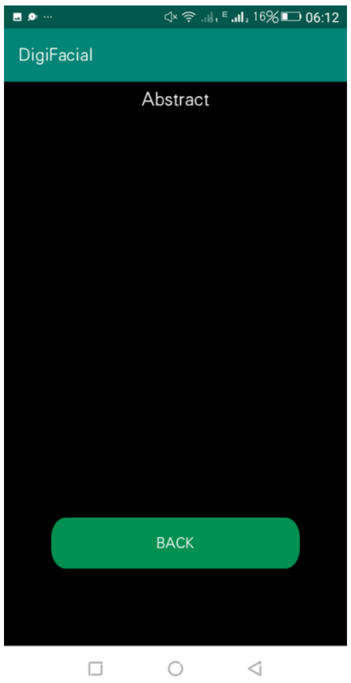
Model predicts : Happy

Fig 16: App predicts happy face



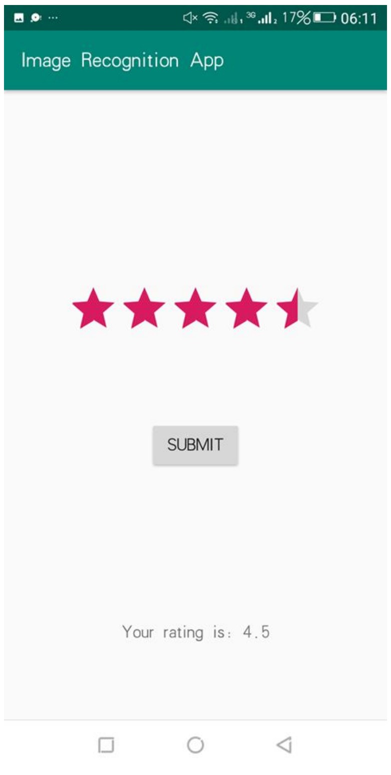
1 .
Smile : 94 . 9383%
LeftEye 80 . 34758%

Fig 17: App predicts smile percentage



Abstract

Fig 18: App predicts a text



Your rating is: 4.5

Fig 19: Rate this app

6 Technical Description

This application can be implemented by using Keras and Tensorflow framework of Python. Our project is based on the principle of machine learning especially based on digit and face recognition. “The goal of this application is to provide messages to the user that machine can also predict something in the field of image classification.”

Facial recognition: Facial recognition is a biometric software application capable of uniquely identifying or verifying the presence of face or not. Facial recognition is mostly used for security purposes, taking attendance, though there is interest in other areas of use.

Digit recognition: Digit recognition is the ability of a machine to receive and interpret intelligible handwritten input from various sources and identify them.

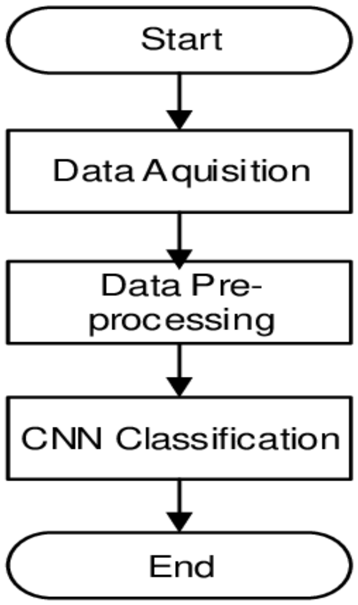


Fig 20: Block diagram of facial recognition system.

The use-case diagram for our MachLearn android app is shown as below:-

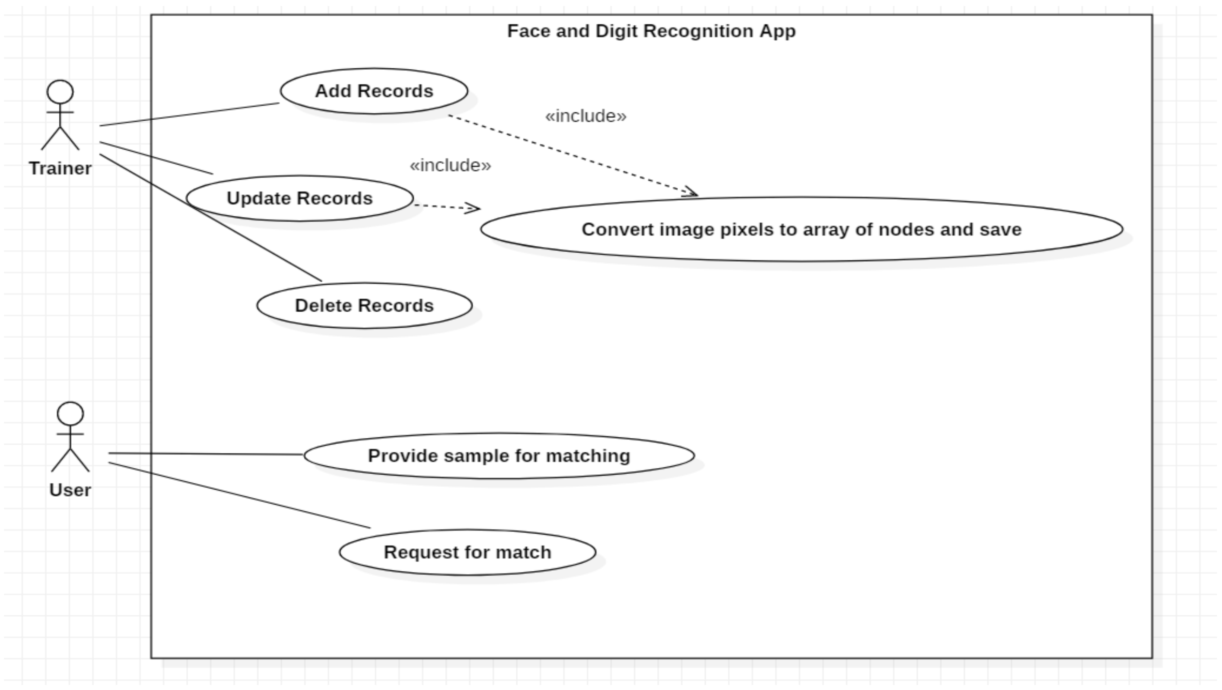

Face and Digit Recognition App
Trainer
Add Records
«include»
«include»
Update Records
Convert image pixels to array of nodes and save
Delete Records
User
Provide sample for matching
Request for match

Fig 21: Use-case diagram for MachLearn App

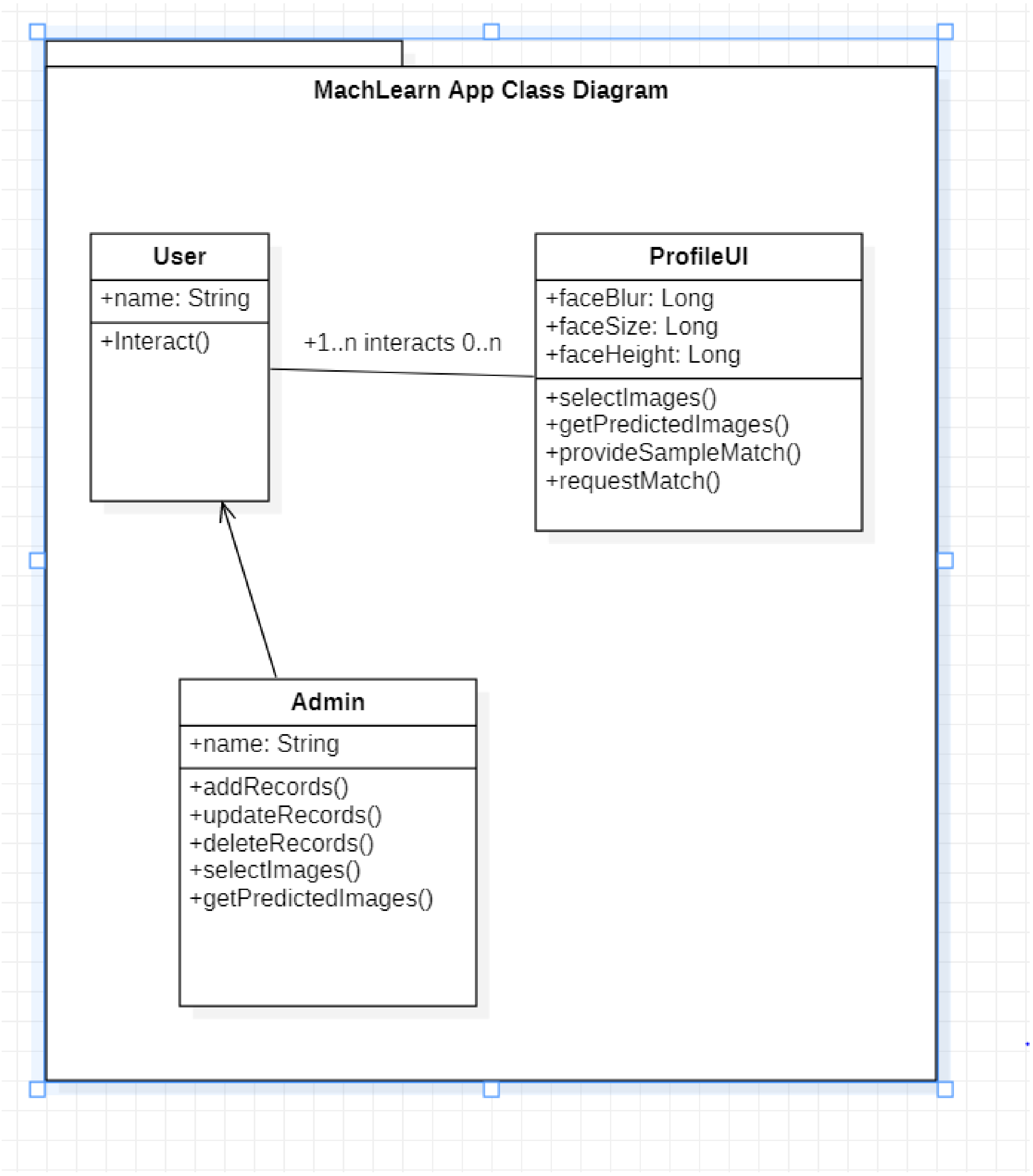


Fig 22: Class diagram for MachLearn app

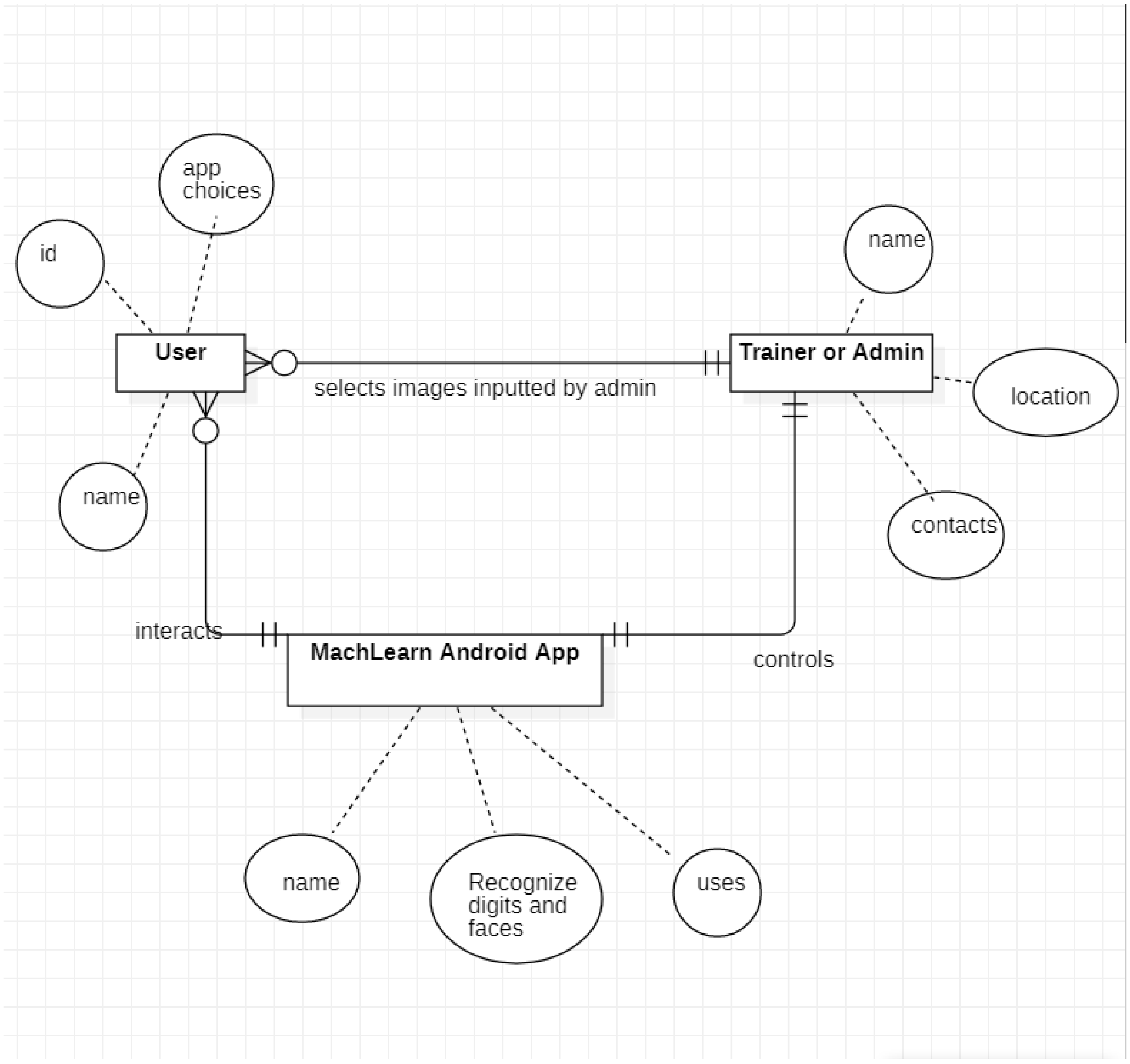


Fig 23: ER diagram for MachLearn app

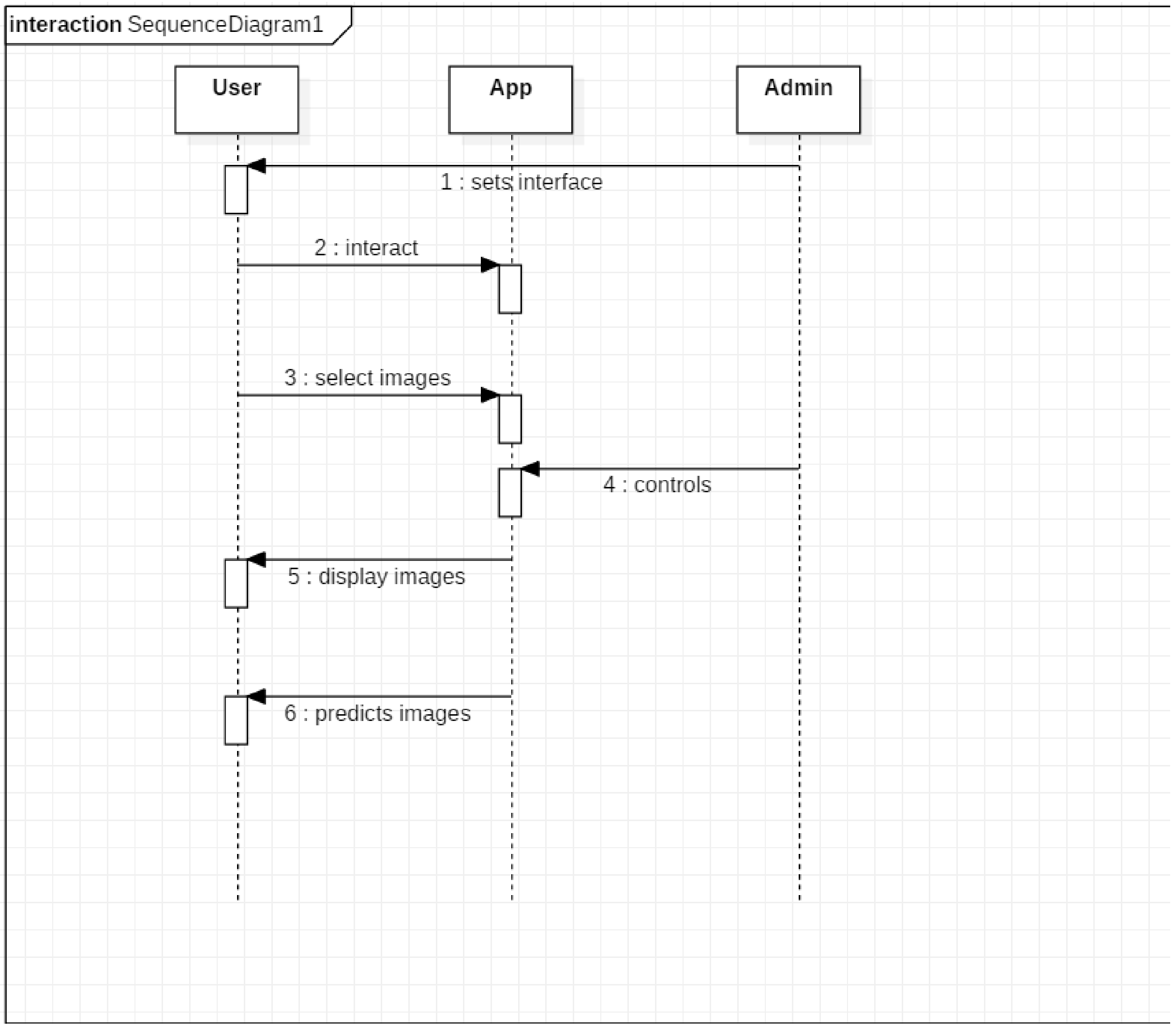


Fig 24: Sequence diagram for MachLearn app

Features of the project:

- Users access
- High quality imagery
- Clean app design
- Accurate functions
- Attractive features
- User-friendly
- Multiple features in a single app

7 Proposed Results

When our project is at its final phase, it will be able to provide users with the services that are essential yet not included in any other existing application. The end product will have the following end results:

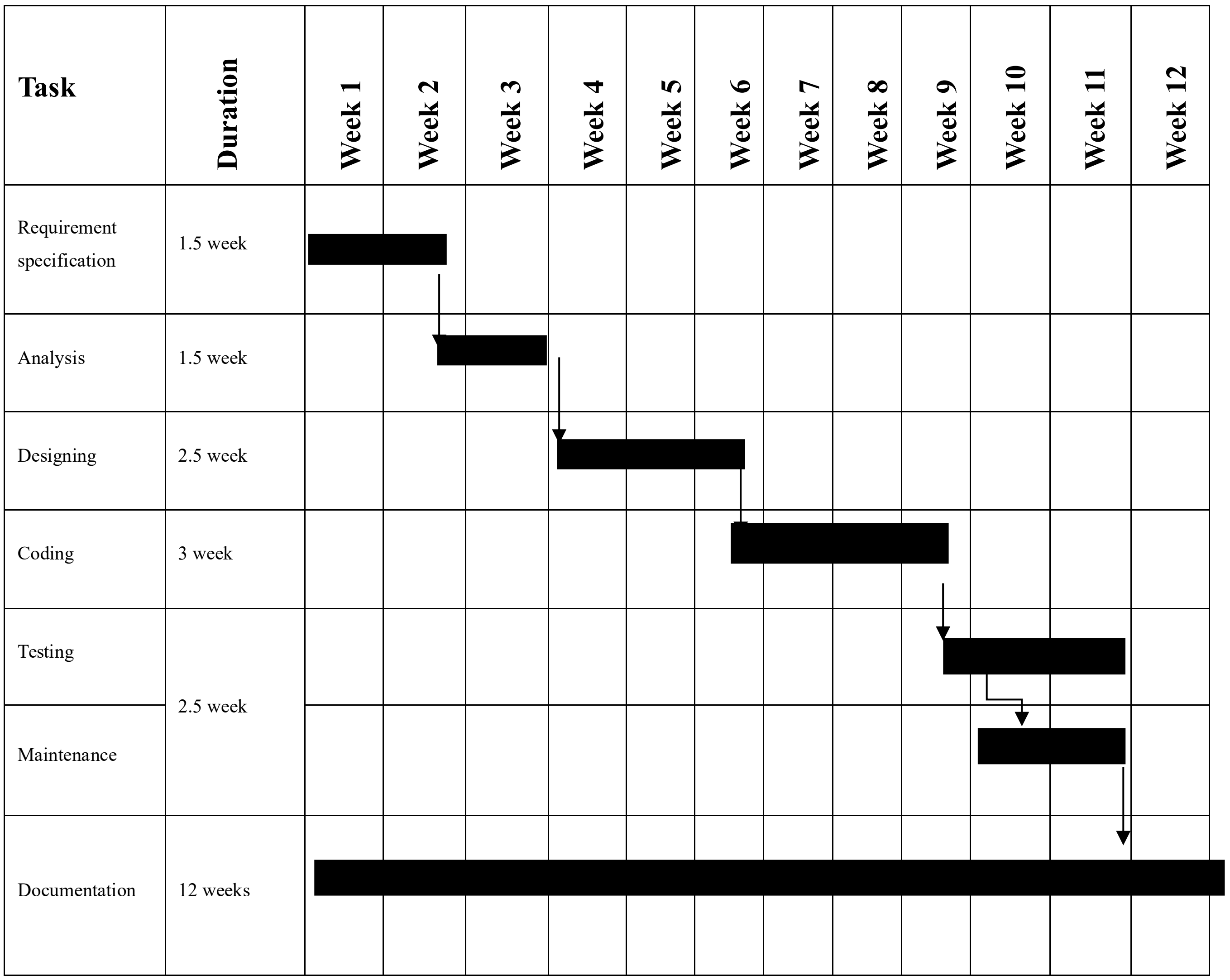
- Users can access digit and face recognition latest app features.
- User can select any inputted image from a list of wide range of images.
- User can ask the app to predict the selected image of any category.
- They will also have the power to promote the flow of information in the images.
- The end product will be able to give users with a proper UI/UX so that they can enjoy service smoothly.
- Users can extract the required information predicted by the app in an efficient way for various other purposes.

8 Project Task and Time Schedule

The project schedule has been designed as per requirements and constraints involved. This project is scheduled to be completed in about 3 months. We are emphasizing on requirement analysis and testing. While carrying out a project, we will document every other thing to protect own self from being accused falsely. Documentation is evidence of a good project management. Testing and debugging should prioritize along with the fine documentation.

TASK	APPROX DURATION (in weeks)
Requirement specification	1.5
Analysis	1.5
Designing	2.5
Coding	3
Testing and Maintenance	2.5
Documentation	12
Total	12

Table 3: Project Task and Schedule



Task	Duration	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
Requirement specification	1.5 week	■	■										
Analysis	1.5 week		■	■									
Designing	2.5 week				■	■	■						
Coding	3 week						■	■	■	■			
Testing	2.5 week									■	■	■	
Maintenance											■	■	
Documentation	12 weeks	■	■	■	■	■	■	■	■	■	■	■	■

Fig 25: Gantt chart

9 References

1. Wikipedia - The Free Encyclopedia. Facial recognition system [online]. Available at https://en.wikipedia.org/wiki/Facial_recognition_system.
2. https://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/ for MNIST digit recognition using keras and tensorflow.
3. https://en.wikipedia.org/wiki/MNIST_database for MNIST dataset for digit recognition.
4. https://www.cs.toronto.edu/~kriz/cifar.html for cifar 10 dataset for image recognition.
5. https://www.tensorflow.org/ for tensorflow framework.
6. https://keras.io/ for keras framework.
7. http://yann.lecun.com/exdb/mnist/
8. https://machinelearningmastery.com/introduction-python-deep-learning-library-tensorflow/
9. https://www.geeksforgeeks.org/introduction-convolution-neural-network/