HOME SECURITY AND DOOR CONTROL USING RAPID FACE DETECTION METHOD

Report submitted to the SASTRA Deemed to be University
as the requirement for the course

CSE300 - MINI PROJECT

Submitted by

B ADITYA SRINIVAS

(Reg. No.: 122003016, B.Tech Computer Science and Engineering)

B HESHWA

(Reg. No.: 122003091, B.Tech Computer Science and Engineering)

N A KKRISH

(Reg. No.: 122003120, B.Tech Computer Science and Engineering)

January 2022

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SCHOOL OF COMPUTING

THANJAVUR – 613 401

<u>Bonafide Certificate</u>

This is to certify that the report titled "**Home security and Door control using rapid face detection method**" submitted as a requirement for the course, CSE300: **MINI PROJECT** for B.Tech. is a bonafide record of the work done by **Mr. B Aditya Srinivas (Reg. No.: 122003016, B.Tech Computer Science and Engineering), B Heshwa (Reg. No.: 122003091, B.Tech Computer Science and Engineering), N A Kkrish (Reg. No.: 122003120, B.Tech Computer Science and Engineering)** during the academic year 2021-22, in the School of Computing, under my supervision.

Signature of Project Supervisor	**:** [illegible]
Name with Affiliation	**:** Dr. A. Suganya, Asst.Professor, CSE, SoC
Date	**:** 13-01-2022

Mini Project *Viva voc*e held on ________________

Examiner 1 **Examiner 2**

Acknowledgements

First and foremost, we are thankful to God Almighty for giving us the potential and strength to work on this project fruitfully. I express our gratitude to **Dr. S. Vaidhyasubramaniam**, Vice-Chancellor, SASTRA Deemed to be University for providing us with an encouraging platform for the course of study.

We also thank **Dr. R. Chandramouli**, Registrar, SASTRA Deemed to be University, for providing us the opportunity to work in the SOC, SASTRA Deemed to be University for the course of the work.

We render our solicit thanks to **Dr. Umamakeswari A**, Dean, SOC and **Dr. Santhi B**, Associate Dean, CSE, SASTRA Deemed to be University for introducing the mini project in our curriculum.

We owe a debt of earnest gratitude towards our guide **Dr. A. Suganya,** Professor/SoC for her continued support and guidance throughout the course of our work. The intriguing questions raised by our panel members **Dr. Rajkumar K,** Professor/CSE, **Dr. Ifjaz Ahmed M,** Professor/CSE, and **Mr. Bharathy C**, Professor/CSE from the department of CSE helped us in shaping this project to its worth.

We also extend our sincere thanks to our parents for their continuous moral support and friends without whose thought-provoking suggestion and encouragement, along with our guide, this work would not have been what it is today.

Table of Contents

List of Figures

Abbreviations

CNN	Convolution Neural Network
UI	User Interface
UX	User Experience
API	Application Programming Interface
GPU	Graphics Processing Unit
CPU	Central Processing Unit
XML	Extensible Markup Language
TV	Tele-vision
viz.	namely
IP	Internet Protocol
OS	Operating System
RAM	Random Access Memory
LED	Light Emitting Diode
CV	Computer Vision

Abstract

The number of home thefts is quite increasing nowadays. This project helps them to limit those thefts and give more control to the users on their doors. The camera on the door will be always watching the front of the door. If any person shows up, this faster face detection model detects the human within half of a second, takes a snapshot and send it to the user’s mobile through the internet. The user can control the action of the door whether to open/close. The detailed view is, the picture is first captured from the camera and stored into the memory of Raspberry Pi, then with help of IoT, it sends the picture to the desired user.

The main advantage of this setup is when the user is far away from home. They can be aware of robbers and surveillance their view from the door whenever they needed, for the security of their place. This app also features SOS ability for an emergency, if they got in any bizarre situation, can use this feature, it sends messages to the emergency contact.

KEY WORDS: Face detection, RCNN, IoT, Android, API, Raspberry Pi

CHAPTER 1

PROBLEM STATEMENT

Overview

Security is one of the biggest threats in homes nowadays. Burglars are now effectively using technology, for example, social media apps like Instagram, Facebook, etc., to find whether we are at home or not and sneaks into homes obliviously. According to the latest report *Har Ghar Surakshit 2018 Report: India's Security Paradox* by India’s leading lock manufacturer *Godrej*, a whopping number of 669 house robbery happens every day. And an increase of ~10% extra robbery is seen when compared to the previous year’s rate. An important point to note here is 70% of all thefts in India are home thefts.

If we analyze one of the typical scenarios where the house is occupied by a bachelor who works in a software company, leaves home by morning and returns by night. This house has a high level of threat happening ratio when compared to homes living a joint family where either one of the family members will be available in the home every time. Likewise in another scenario where old aged parents live in home solitarily, their lack of physical energy becomes an advantage to the burglars.

In this state of theft angst, more than 60% of people in India are not prepared to handle house threats. 61% of individuals would rather not move up to cutting edge wellbeing for homes. This ignorance showed by the people is mainly due to the two hardships faced by them. One, the product is pricey and the other is the set-up is a time-consuming process.

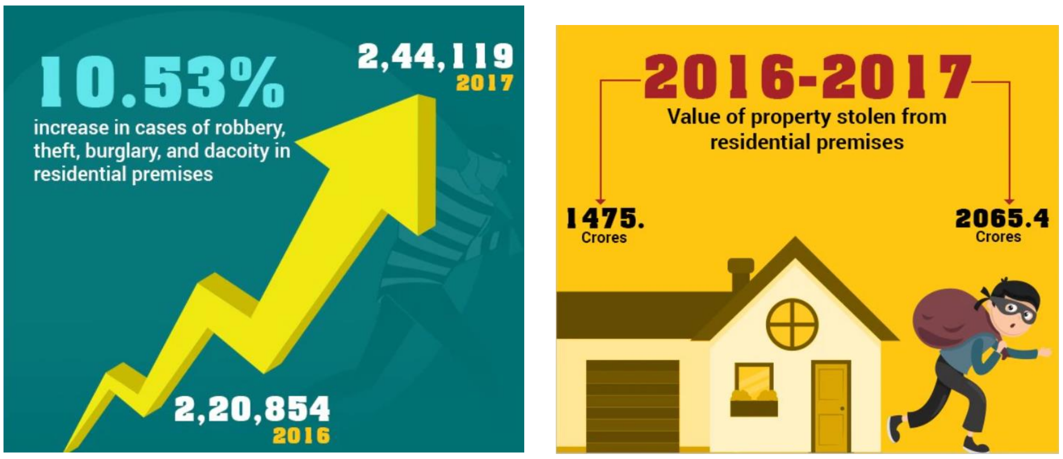

10.53%
increase in cases of robbery, theft, burglary, and dacoity in residential premises
2,44,119
2017
2,20,854
2016
2016-2017
Value of property stolen from residential premises
1475.
Crores
2065.4
Crores

Fig 1.1 **Statistics from report**

CHAPTER 2

SUMMARY OF THE BASE PAPER

Title: . A fast face detection method via convolutional neural network

Publisher: Elsevier

Year: 2019

Journal: Neurocomputing

Indexing: SCI

All face detectors nowadays are mostly working under the principle of Convolution Neural Networks. These face detectors based on convolutional neural networks (CNN) have considerably further developed the identification execution. In any case, the detection speed is as yet the greatest bottleneck, which frustrates the down to earth sending of these face finders on asset restricted stages and can't be used effectively in real-time applications.

This can be achieved by implementing computer vision induced with deep learning techniques, which uses artificial agents to learn or derive knowledge from live data. Face detection is performed on feature maps. Just by extracting the needed features, it is not required to extract multi scale features on an image-pyramid employed in the traditional methods. Hence the efficiency of the face detector is significantly improved.

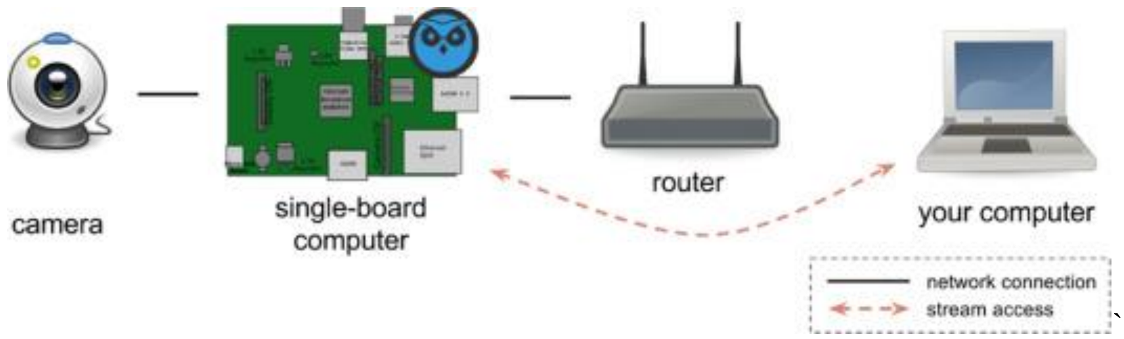


Fig 2.1 **Flow of this project**

CHAPTER 3

MERITS AND DEMERITS OF THE BASE PAPER

Merits

Object detection was first implemented using trajectory-based techniques, which was inefficient, so local cuboids were used on the trajectory path to make it more reasonable. After a while, CNN replaced these techniques. Faster RCNN is much more effective than general CNN and hence used.

Faster RCNN was used to operate in crowded or dense scenes which were hard to accomplish using general CNN methods to differentiate objects on a huge dataset.

Demerits

The Faster RCNN model trained in the base paper was built from scratch, which could have yielded better results if the pre-trained Faster RCNN model was used. The layers that the base paper followed were not sufficient to yield a result of more than 60% accuracy. If we make it to run more iterations, then it overfits with the data provided in training.

CHAPTER 4

PRODUCT DESIGN

To build a durable and simple product we used some electronic devices like Raspberry Pi Model 3B, relay switch, electro-magnetic latch and batteries. On the other hand, the software side includes a lot of open-source tools and editors like Android Studio, Google Colab, Figma, TensorFlow and many.

The face detection was first done with the help of the base paper and by following some YouTube tutorials when we got struck. Then it was amplified with the inputs given by our guide. The complete face detection was implemented in Google Colab, a cloud service platform, and tested out the results by there.

Our Android app ***Tele-Port****'s* User Interface (UI) was first designed in Figma, a UI design tool, and later implemented in Android Studio step-by-step. Then we integrated our app with Firebase for cloud storage to save the pictures and credentials of users and doors.

Side-by-side we worked in hardware path, Raspberry Pi was set up and implemented a face detection model and tried working thereby capturing the image whenever a face is detected. Then bought an electro-magnetic latch which is controlled by a relay.

The hardware is then connected with a server and API's been created uniquely which was accessed by the Raspberry for sending the photo captured to an integrated cloud storage Firebase. Then it redirects that image to the respective user by a push notification and in-app image holder.

The user has now options to control the door via the app, see the history - records of door controlled by the user, view the person in front of the door, add more doors to control, and more.

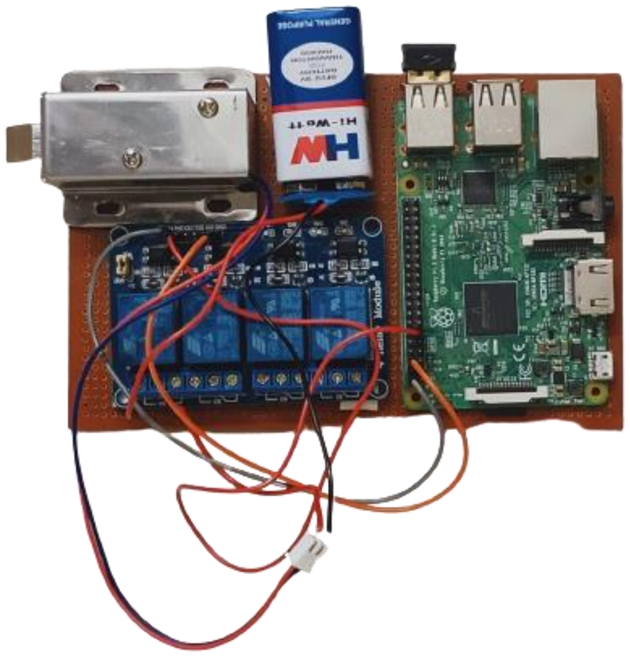


Fig 4.1 **Final setup of project**

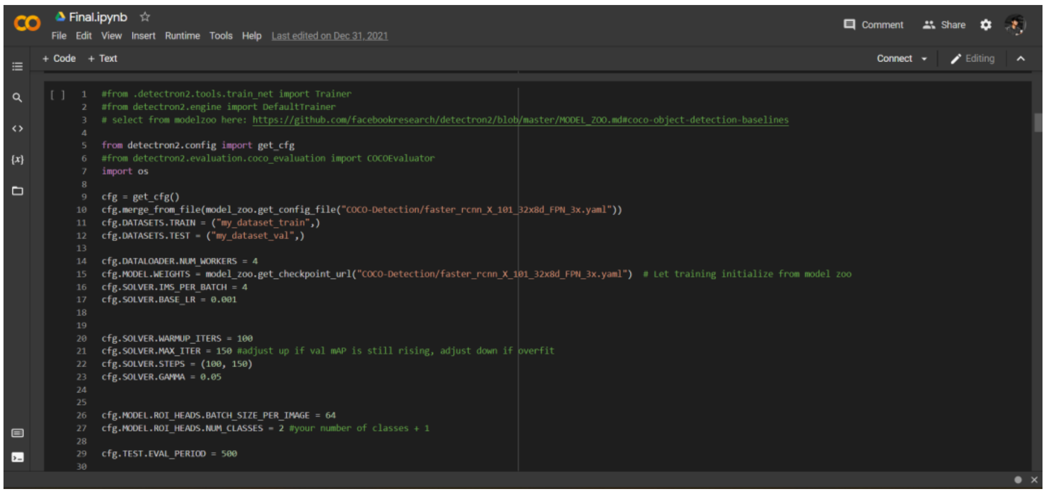
CHAPTER 5

TOOLS USED

Google Colab

Google Colab is an open-source product from Google Research. Anybody with having G-Mail account can write and execute python code through the browser. This is similar to the local tool named Jupyter Notebook with no setup to do. This also gives free access to GPUs and CPUs as computing resources. By using this we don't need to use our computer's resources to execute our program.

Using this Colab we programmed a lot. Mainly our face detection code is implemented mostly in Google's Colab. We trained a lot of face detection models, literally more than 4 models, where we finally created a model that was inspired by our base paper. The GPUs that were available helped us a lot to train our model faster.


Final.ipynb
File Edit View Insert Runtime Tools Help Last edited on Dec 31, 2021
Comment Share
+ Code + Text
Connect Editing

```
#from .detectron2.tools.train_net import Trainer
#from detectron2.engine import DefaultTrainer
# select from modelzoo here: https://github.com/facebookresearch/detectron2/blob/master/MODEL_ZOO.md#coco-object-detection-baselines

from detectron2.config import get_cfg
#from detectron2.evaluation.coco_evaluation import COCOEvaluator
import os

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("my_dataset_train",)
cfg.DATASETS.TEST = ("my_dataset_val",)

cfg.DATALOADER.NUM_WORKERS = 4
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml")  # Let training initialize from model zoo
cfg.SOLVER.IMS_PER_BATCH = 4
cfg.SOLVER.BASE_LR = 0.001


cfg.SOLVER.WARMUP_ITERS = 100
cfg.SOLVER.MAX_ITER = 150 #adjust up if val mAP is still rising, adjust down if overfit
cfg.SOLVER.STEPS = (100, 150)
cfg.SOLVER.GAMMA = 0.05


cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 64
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 2 #your number of classes + 1

cfg.TEST.EVAL_PERIOD = 500

```

Fig 5.1 **Google Colab**

Figma

Figma is an online web-based editor to design a User Interface (UI) for mobile applications, web applications and many. It can be used in all stages of UI/UX designing starting from wireframing to prototyping. This is different from other graphic editing tools because it is so easy to use and edit.

We designed our app Tele-Port's UI here. It took some number of dedicated hours to design it and prototype it completely. The design was simple and followed a minimalistic approach. The colours used were pastel colours so it will be pleasing and not too bright or too dull.

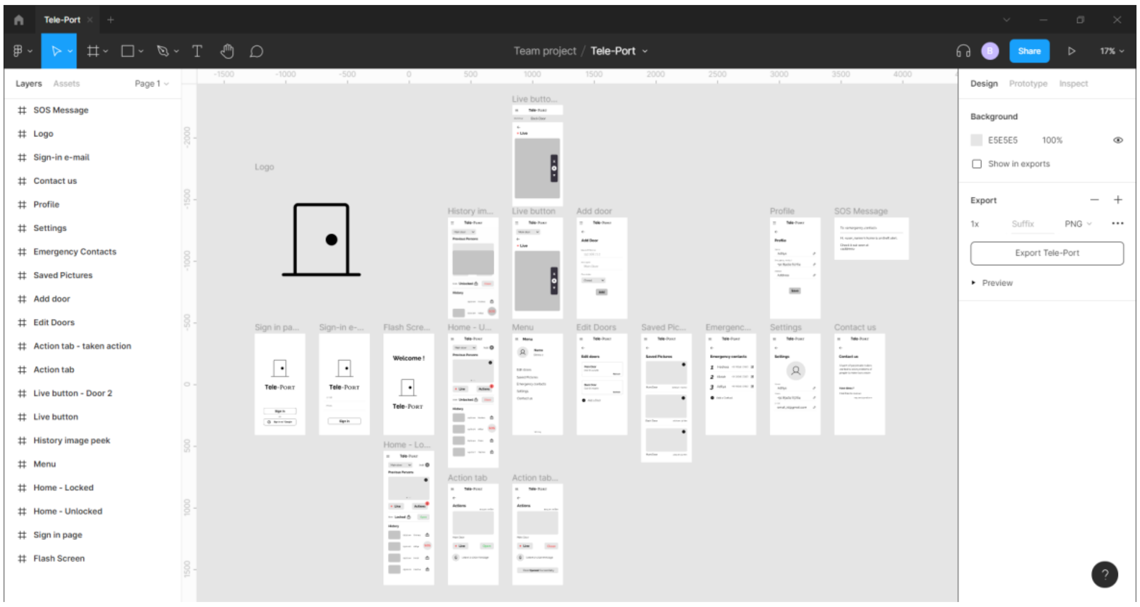


Fig 5.2 **Figma – Total designs for the app**

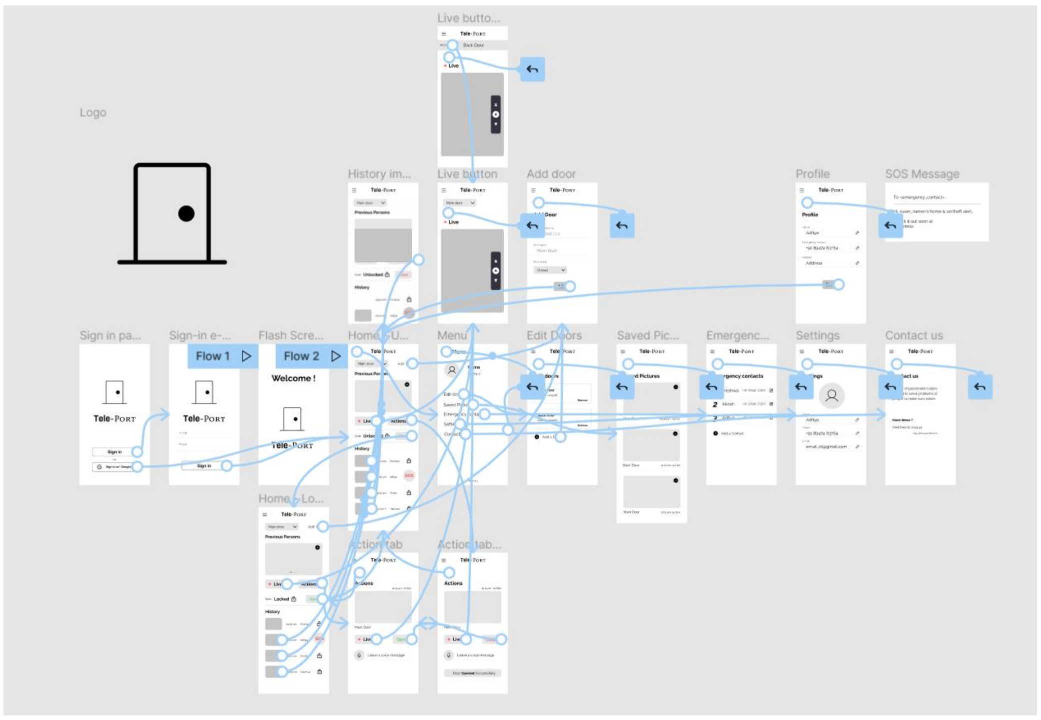
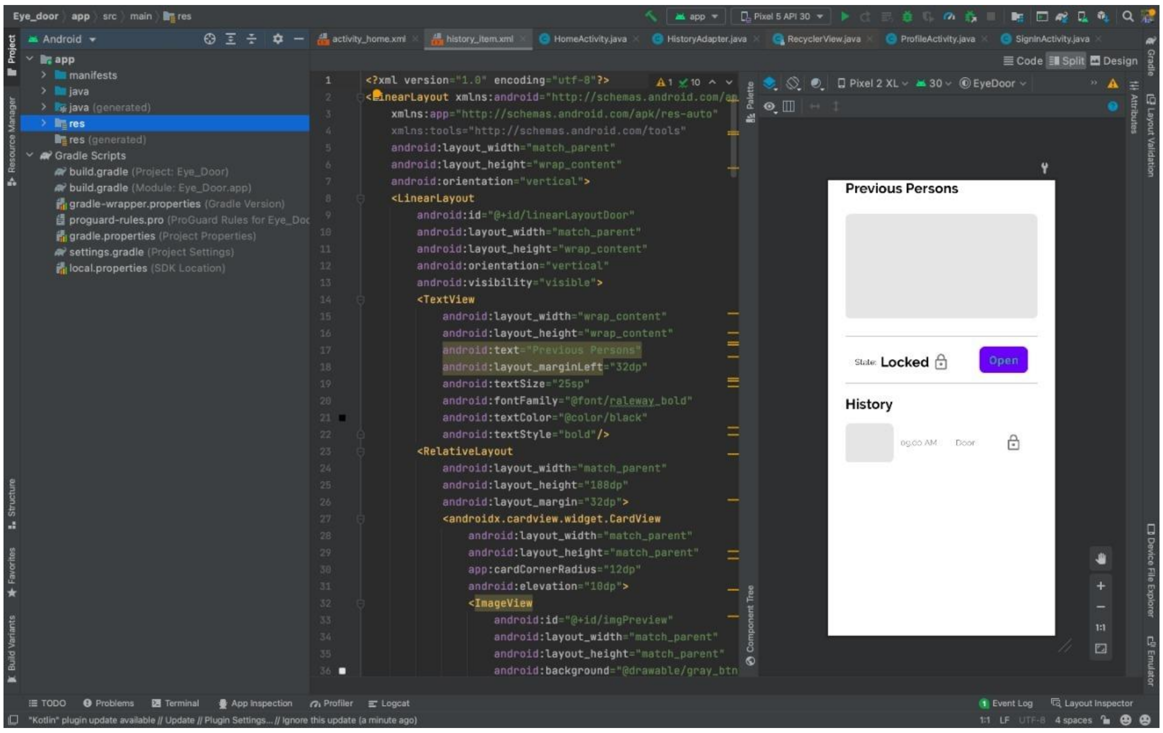


Fig 5.3 **Figma prototyping interface**

Android Studio

Android Studio is an open-source platform for creating apps for various mobiles.It is mainly coded using Java for activity and XML for designing. Kotlin is also used for some purpose. Code modules allow us to divide the project into parts of different blocks to independently build, test and debug.

We used the android studio to code the entire Tele-Port app. Using the design, we did on Figma, designed as it is on Android Studio.



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/ap
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <LinearLayout
        android:id="@+id/linearLayoutDoor"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:visibility="visible">
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Previous Persons"
            android:layout_marginLeft="32dp"
            android:textSize="25sp"
            android:fontFamily="@font/raleway_bold"
            android:textColor="@color/black"
            android:textStyle="bold"/>
        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="188dp"
            android:layout_margin="32dp">
            <androidx.cardview.widget.CardView
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                app:cardCornerRadius="12dp"
                android:elevation="10dp">
                <ImageView
                    android:id="@+id/imgPreview"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"
                    android:background="@drawable/gray_btn
```

Fig 5.4 **Android App making**

API & Server

API stands for Application Programming Interface. We created some API calls for controlling the opening and closing of the door in Flask. Made Raspberry Pi as a server and hosted it there. To make that server publicly accessible from anywhere around the world, we used the port-forwarding technique to achieve it.

Firebase

Firebase is an online product by Google. It is an application improvement programming that empowers engineers to foster Android, iOS and Web applications. It gives devices to following, investigation, analytics and fixing application crashes. It is mainly used for saving the database of the app.

Our Tele-Port app is also integrated with Google's Firebase for the database from user's credentials to door's credentials (IP Addresses of each door). We have used Firebase as our main backend. Used four services of Firebase, viz.,

1. Authentication:

   This Firebase's authentication will take care of the complete authentication of the app, for using email and password verification.

2. Realtime database:

   We have used Firebase's real-time database in-order to store all the data from door status, user details, etc.

3. Storage:

   To store the images captured from the camera when someone comes in front of the door, we use Firebase's storage for that.

4. Push notification:

   When a person comes in front of the door, the picture captured is sent to the user's mobile via this Firebase's push notification service.

Raspberry Pi 3, Model-B

Raspberry Pi is a hand-sized pc that costs much lesser than a real computer that plugs into any monitor and uses the usual keyboard and mouse. This has been used in a wide array of digital maker projects. Raspberry Pi brings out its OS, names Raspbian OS. Raspberry Pi has memory (RAM), processor, USB ports, Ethernet port, display connector, camera connector and many.

We used Raspberry Pi 3 Model B for this mini-project. This Pi 3 Model B is equipped with 1 GB RAM, 4 USB 2 ports, 1.2 GHz 64-bit CPU, 100 Base Ethernet port, Full-size HDMI.

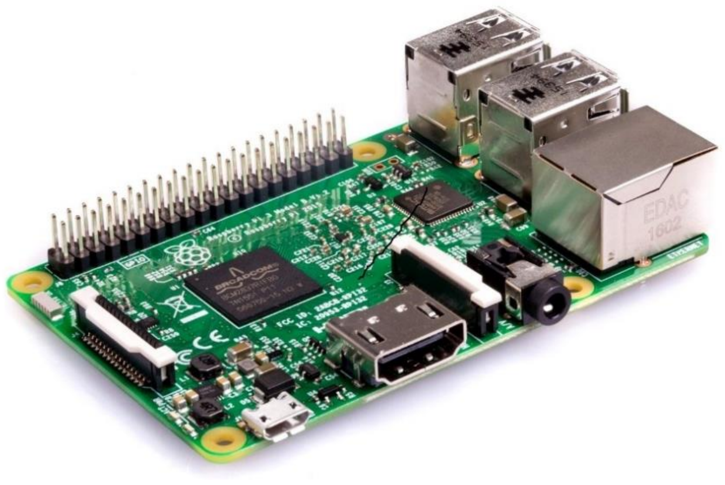


Fig 5.5 **Raspberry Pi 3, Model-B**

5V Relay

Relay is an automatic switch, used commonly in the automatic control circuit to control a high current using a low current signal. We used a 5V relay which means, the input voltage of this relay signal ranges from 0 to 5V. This 5V relay can be controlled by any 5V digital output from our microcontroller. The LED in the relay indicated the status of the relay. There will be three wires access from the relay.

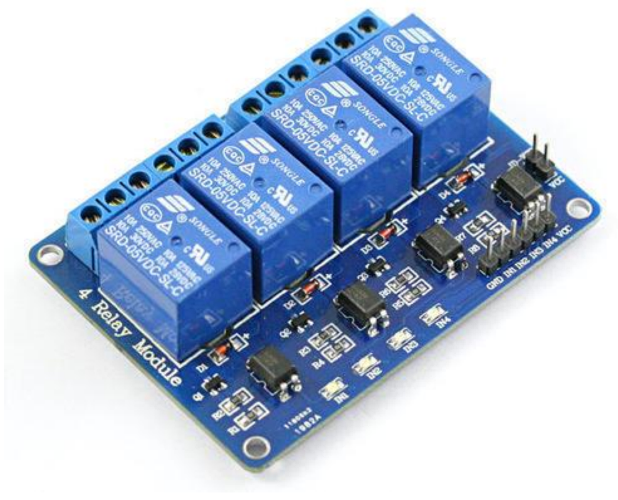


Fig 5.6 **Four Channel 5V relay**

Solenoid electro-magnetic latch

This DC 12V Solenoid Electromagnetic Cabinet Door Lock can be utilized for locking a door, stockpiling rack, file organizer and so on. The secret method of opening can be utilized for a crisis. The lock functions as the circuits detach, and it will open as the moment power-on. It is consistent, though, and energy-saving and has a long life expectancy. In the counter burglary and shockproof plan, the lock is superior to different sorts of locks. After interfacing the wires and when the flow is free, the electric lock can handle the entryways opening and closing.

We bought this latch online and used this for our project. This latch needs a constant 12 V DC supply for its operation. We provided this by batteries and connected one wire from the relay for controlling its actions. The relay is connected to the Raspberry Pi, and while the user presses the button it makes the actions accordingly.

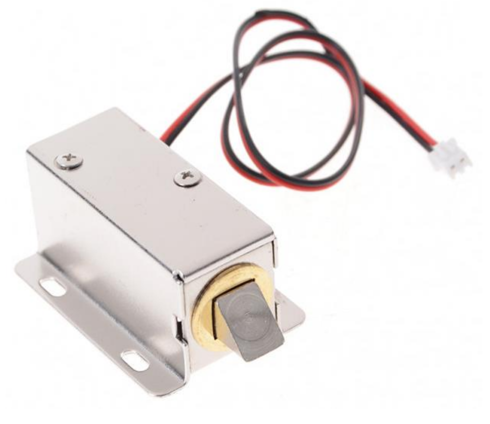


Fig 5.7 **Solenoid Electro-magnetic latch**

LabelImg and Roboflow

LabelImg is a free, open-source device for graphically marking pictures. It's written in Python. It's a simple, free method for marking two or three hundred pictures to evaluate an object location project. LabelImg upholds naming in VOC XML or YOLO text document design.

Roboflow Annotate is a annotate tool that extraordinarily smoothes out the most common way of going from raw pictures to a prepared and deployed CV model. Regardless of whether you want to address a solitary annotation or mark a whole dataset, you can now do it using Roboflow without downloading a different program.

We used LabelImg and Roboflow for annotating the faces in the images for training the model.

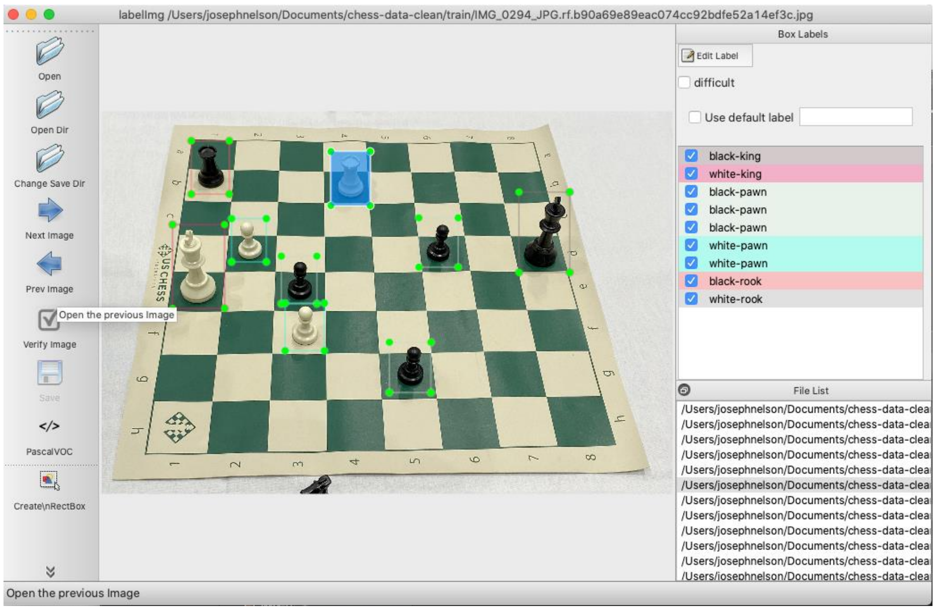


Fig 5.8 **LabelImg software**

Datasets

Datasets are a collection of data. We downloaded a variety of image datasets. Both face and non-face (generic) datasets we used for several models. Some of the datasets we used are '10k US Adult', 'Caltech256', 'WiderFace AFW', 'FDDB', 'IndoorCVPR' and 'VOC2012'.

CHAPTER 6

SOURCE CODE

Face Detection

```
!pip install -U torch==1.5 torchvision==0.6 -f
https://download.pytorch.org/whl/cu101/torch_stable.html
!pip install cython pyyaml==5.1
!pip install -U
'git+https://github.com/cocodataset/cocoapi.git#subdirectory=PythonAPI'
import torch, torchvision
print(torch.__version__, torch.cuda.is_available())
!gcc --version
```

```
!pip install detectron2==0.1.3 -f
https://dl.fbaipublicfiles.com/detectron2/wheels/cu101/torch1.5/index.html
```

```
import detectron2
from detectron2.utils.logger import setup_logger
setup_logger()

# import some common libraries
import numpy as np
import cv2
import random
from google.colab.patches import cv2_imshow

# import some common detectron2 utilities
from detectron2 import model_zoo
from detectron2.engine import DefaultPredictor
from detectron2.config import get_cfg
from detectron2.utils.visualizer import Visualizer
from detectron2.data import MetadataCatalog
from detectron2.data.catalog import DatasetCatalog
```

```
#  Import and Register Custom Dataset
!curl -L "https://app.roboflow.com/ds/gt3trRECfJ?key=6ZVGLTvNLL" >
roboflow.zip; unzip roboflow.zip; rm roboflow.zip
```

```
from detectron2.data.datasets import register_coco_instances
register_coco_instances("my_dataset_train", {},
"/content/train/_annotations.coco.json", "/content/train")
register_coco_instances("my_dataset_val", {},
"/content/valid/_annotations.coco.json", "/content/valid")
```

```
register_coco_instances("my_dataset_test", {},
"/content/test/_annotations.coco.json", "/content/test")
```

```
#visualize training data
my_dataset_train_metadata = MetadataCatalog.get("my_dataset_train")
dataset_dicts = DatasetCatalog.get("my_dataset_train")

import random
from detectron2.utils.visualizer import Visualizer

for d in random.sample(dataset_dicts, 3):
    img = cv2.imread(d["file_name"])
    visualizer = Visualizer(img[:, :, ::-1],
metadata=my_dataset_train_metadata, scale=0.5)
    vis = visualizer.draw_dataset_dict(d)
    cv2_imshow(vis.get_image()[:, :, ::-1])
```

```
#We are importing our own Trainer Module here to use the COCO validation
evaluation during training. Otherwise no validation eval occurs.

from detectron2.engine import DefaultTrainer
from detectron2.evaluation import COCOEvaluator

class CocoTrainer(DefaultTrainer):

  @classmethod
  def build_evaluator(cls, cfg, dataset_name, output_folder=None):

    if output_folder is None:
        os.makedirs("coco_eval", exist_ok=True)
        output_folder = "coco_eval"

    return COCOEvaluator(dataset_name, cfg, False, output_folder)
```

```
from detectron2.config import get_cfg
import os

cfg = get_cfg()
cfg.merge_from_file(model_zoo.get_config_file("COCO-
Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml"))
cfg.DATASETS.TRAIN = ("my_dataset_train",)
cfg.DATASETS.TEST = ("my_dataset_val",)
cfg.DATALOADER.NUM_WORKERS = 4
cfg.MODEL.WEIGHTS = model_zoo.get_checkpoint_url("COCO-
Detection/faster_rcnn_X_101_32x8d_FPN_3x.yaml")
cfg.SOLVER.IMS_PER_BATCH = 4
cfg.SOLVER.BASE_LR = 0.001


cfg.SOLVER.WARMUP_ITERS = 100
cfg.SOLVER.MAX_ITER = 150
cfg.SOLVER.STEPS = (100, 150)
cfg.SOLVER.GAMMA = 0.05

cfg.MODEL.ROI_HEADS.BATCH_SIZE_PER_IMAGE = 64
cfg.MODEL.ROI_HEADS.NUM_CLASSES = 2

cfg.TEST.EVAL_PERIOD = 500


os.makedirs(cfg.OUTPUT_DIR, exist_ok=True)
trainer = CocoTrainer(cfg)
trainer.resume_or_load(resume=False)
trainer.train()
```

```
#test evaluation
from detectron2.data import DatasetCatalog, MetadataCatalog,
build_detection_test_loader
from detectron2.evaluation import COCOEvaluator, inference_on_dataset

cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.85
predictor = DefaultPredictor(cfg)
evaluator = COCOEvaluator("my_dataset_test", cfg, False,
output_dir="./output/")
val_loader = build_detection_test_loader(cfg, "my_dataset_test")
inference_on_dataset(trainer.model, val_loader, evaluator)
```

```
#  Predict an image
import os

cfg.MODEL.WEIGHTS = os.path.join(cfg.OUTPUT_DIR, "model_final.pth")
cfg.DATASETS.TEST = ("my_dataset_test", )
cfg.MODEL.ROI_HEADS.SCORE_THRESH_TEST = 0.7   # set the testing threshold
for this model
predictor = DefaultPredictor(cfg)
test_metadata = MetadataCatalog.get("my_dataset_test")
```

```
# Image prediction
img_path =
"/content/test/159_0016_jpg.rf.04ad316f2879b326cf5dc559d9ab8fd7.jpg"

im = cv2.imread(img_path)
outputs = predictor(im)
v = Visualizer(im[:, :, ::-1],
              metadata=test_metadata,
              scale=0.8
                )
out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
cv2_imshow(out.get_image()[:, :, ::-1])
```

Raspberry Pi - Server code

```
import cv2 ,requests ,json ,pyrebase ,os
import datetime
from detectron2.utils.visualizer import Visualizer
from detectron2.engine import DefaultPredictor

USERNAME="bas@user.com"
PASSWORD="teleport"
USERID="pjAwkk0xpwgDBJYIjKJnjC8hHF92"
DOORID="-Mruul1HApY4XbnROpJL"
i=0
def firebaseInitalization():
    firebaseConfig = {
        "apiKey": "AIzaSyDhGWVroOvZ8zkkJWb73rYxWORYMV-kVhg",
        "authDomain": "eye-door.firebaseapp.com",
        "projectId": "eye-door",
        "databaseURL": "https://eye-door-default-rtdb.firebaseio.com/",
        "storageBucket": "eye-door.appspot.com",
        "messagingSenderId": "471384265439",
        "appId": "1:471384265439:web:3e902d476da793a643cf6a",
        "measurementId": "G-Q2PEX5GXN7"
    }
    firebase = pyrebase.initialize_app(firebaseConfig)
    return firebase

def authenticate_user(firebase):
    auth = firebase.auth()
    user = auth.sign_in_with_email_and_password(USERNAME, PASSWORD)
    user = auth.refresh(user['refreshToken'])
    return user

def sendRequest(url):
    URL = "https://fcm.googleapis.com/fcm/send"
    KEY =
"AAAAbcCxEt8:APA91bGibdHtp4QtOi8By5_QjAJXy_wwisNjPvalRY_rCwi7T1_tLidWTBPCsK
CKAVNPbg4CS2-CTLARcCvEs8AbFh7V0WiHfL_j-ToPI7Y-KbTHKMDQn90cczg3FPxFnF2tF1E2-
Q3-"
    notification={
        "body": "Please take action",
        "title": "Someone came to your home",
        "image": url
    }
    body = {
        "to": "/topics/"+DOORID,
        "notification": notification

    }
    headers = {'content-type': 'application/json',
               "Authorization":"key="+KEY}
    r = requests.post(url=URL,headers=headers,data=json.dumps(body))
    print(r)

def uploadImage(firebase,user,filename):
    global i
    storage = firebase.storage()
    PATH = "screenshots/img"+str(i)+".jpg"
    i +=1
    if i==10:
        i=0

    storage.child(PATH).put(filename)
    url = storage.child(PATH).get_url(user['idToken'])

    db = firebase.database()
    # data = {"data":url}
    # data ={"imgUrl":url,"name":"Default User","St"}
    # db.child("image").remove()
    db.child("Users").child(USERID).child("Doors").child(DOORID).update({"i
mage":url})
    return url



def main():
    filename = "image.jpg"
    firebase = firebaseInitalization()
    user =  authenticate_user(firebase)
    cap = cv2.VideoCapture(0)
    time = datetime.datetime.now()
    while True:
        _, img = cap.read()
        im = img

        predictor = DefaultPredictor(cfg)
        outputs = predictor(im)
        v = Visualizer(im[:, :, ::-1], metadata=test_metadata, scale=0.8)
        out = v.draw_instance_predictions(outputs["instances"].to("cpu"))
        cv2_imshow(out.get_image()[:, :, ::-1])

        if len(faces)!=0:
            now = datetime.datetime.now()
            if(now-time).total_seconds()>30:
                cv2.imwrite(filename, img)
                url = uploadImage(firebase,user,filename)
                sendRequest(url)
                time = datetime.datetime.now()
        cv2.imshow('Portal', img)
        k = cv2.waitKey(30) & 0xff
        if k == 27:
            break
    cap.release()


if __name__ == '__main__':
    main()
```

```
from flask import Flask

app = Flask(__name__)

@app.route('/')
def index():
    return 'Hello world'


@app.route('/opendoor')
def openDoor():
    res = {}
    res['status'] = True
    res['output'] = 'Open Door'

    return resx


if __name__ == '__main__':
    app.run(debug=True,host='0.0.0.0')
```

CHAPTER 7

SNAPSHOTS

Face detection

X-axis - No. of iterations

Y-axis - Accuracy

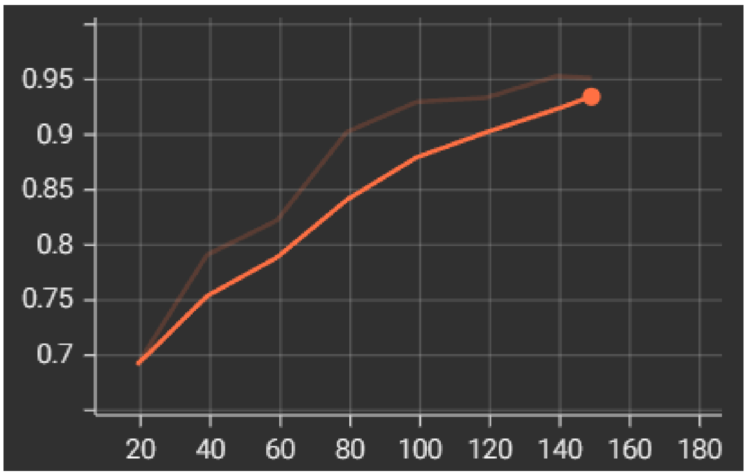


Fig 7.1 **Accuracy of model over the iterations**

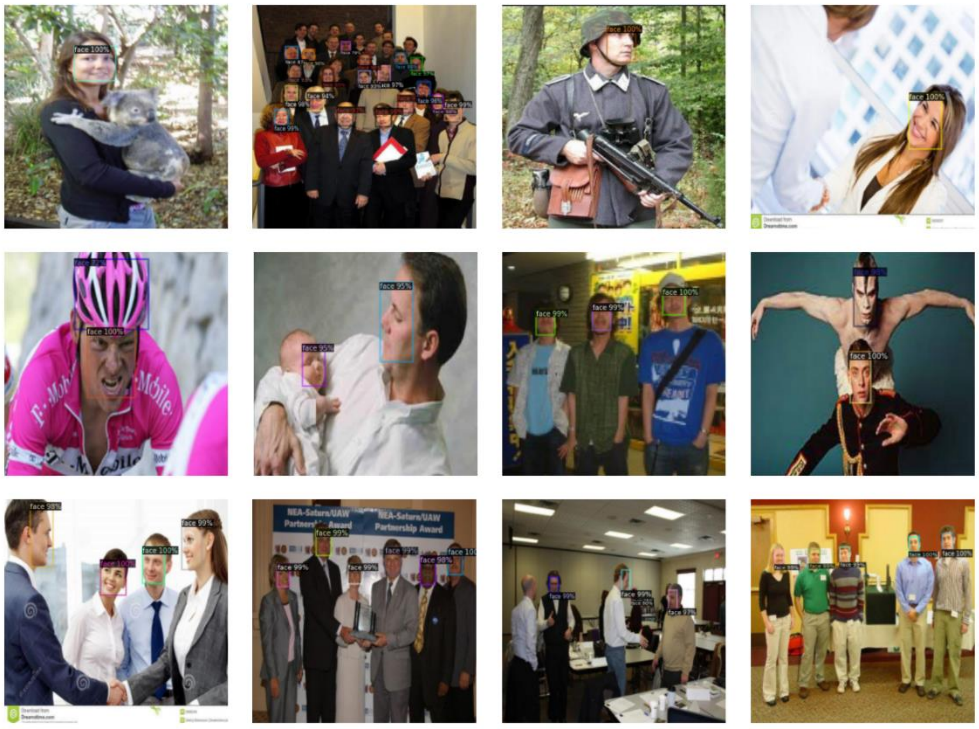


Fig 7.2 **Testing images detections**

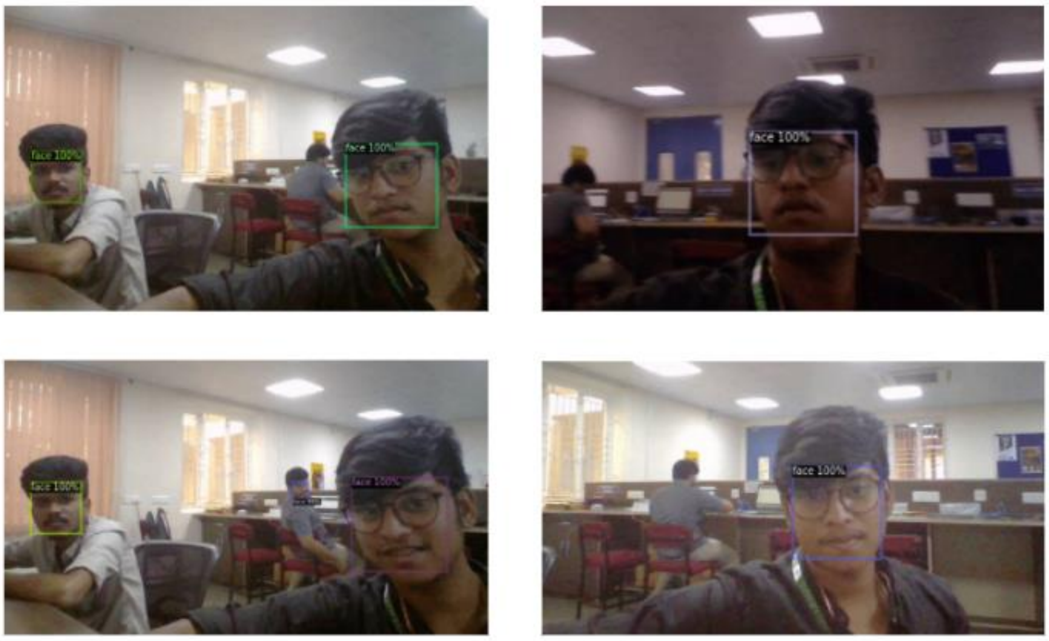
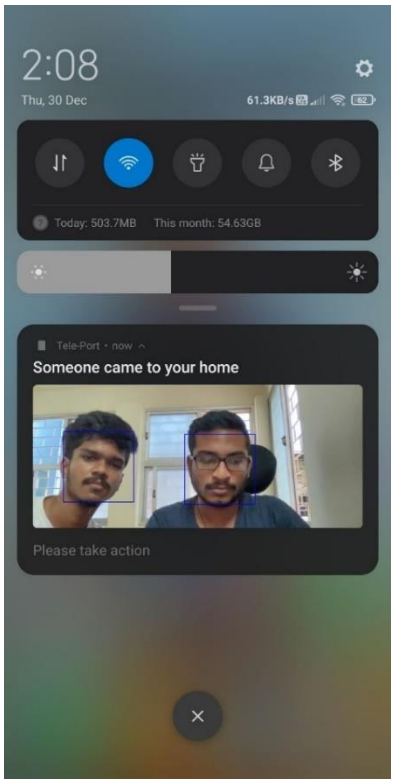
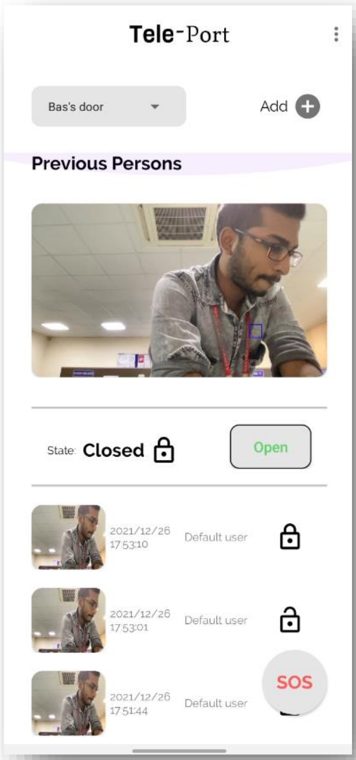


Fig 7.3 **Faces detected on Live camera**


2:08
Thu, 30 Dec
61.3KB/s
Today: 503.7MB This month: 54.63GB
Tele-Port • now
Someone came to your home
Please take action

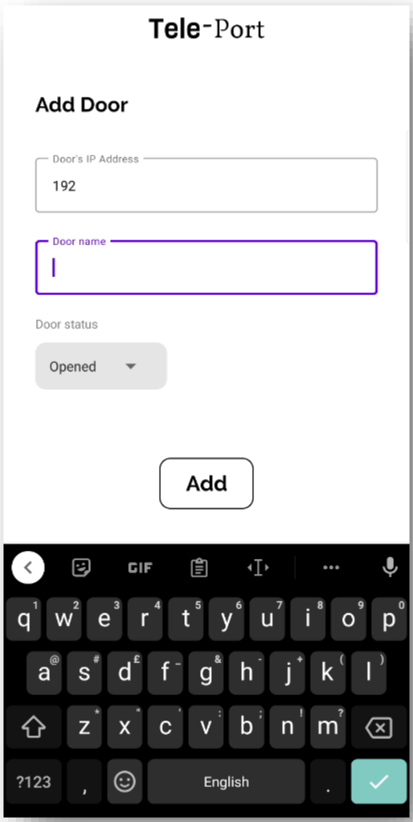

Tele-Port
Bas's door
Add
Previous Persons
State: Closed
Open

2021/12/26 17:53:10	Default user	
2021/12/26 17:53:01	Default user	
2021/12/26 17:51:44	Default user	

SOS

Fig 7.4 **Notification from app**

Fig 7.5 **Tele-Port App's home screen**

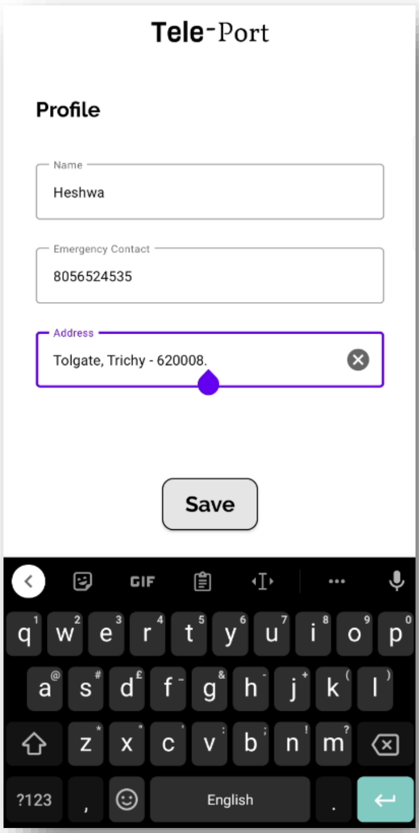


Door's IP Address: 192

Door name:

Door status: Opened

Fig 7.6 **Adding door to the app**

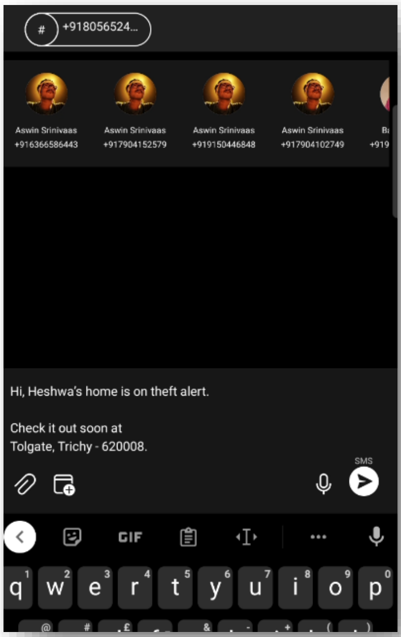


Name: Heshwa

Emergency Contact: 8056524535

Address: Tolgate, Trichy - 620008.

Fig7.7 **Editing User detail**



Hi, Heshwa's home is on theft alert.

Check it out soon at
Tolgate, Trichy - 620008.

Fig 7.8 **SOS feature sending message to emergency contact**

CHAPTER 8

CONCLUSIONS

A face detection model was created, which will detect the faces of the persons standing in front of the door. The results yielded an accuracy of more than 95%. First of all, the input image is converted to frames, which are given to the model for face detection. The face detected picture is sent to the mobile of the user with the help of Firebase. The user now has total access to control the door and see the history of the door unlocks.

Features of Tele-Port app:

- Notification
    - Photo of the person detected in front of the door is sent to the user's mobile via push notification.
- Control the door
    - User has the action of either Open/Close the door accordingly
- History
    - The history of actions taken by the user can be viewed on the home screen of the app. The action taken by the person will also be shown with the image.
- Add many doors
    - The user can add and control many doors in a single app. The process of adding a door is so simple that the user just needs to enter the IP address and name of the door.
- SOS and Emergency contact
    - SOS feature in our app is one of the highlighted features. When we spot a burglar outside the home, we can simply press the SOS button on the home screen and it texts the emergency contact set by us.

FUTURE PLANS

The Tele-port app is now bug-free and easily useable by any user. We are working on face recognition as an add-on to this. So that the burglars and thieves reported in the police department can be easily identified and referred to the users directly.

Extending the SOS feature is the next work to carry so that it can directly call the police department and report the incident as soon it identifies and alerts the neighbourhood by some alarm.

Adding a small speaker in front of the door is also a part we are working on. So that the user can deliver instructions to the person directly without opening the door.

CHAPTER 9

REFERENCES

[1] **Guanjun Guo, Hanzi Wang, Yan Yan, Jin Zheng, Bo Li,** *A fast face detection method via convolutional neural network*, 2019, *Neurocomputing*

[2] **H. Jiang, E.G. Learned-Miller**, *Face detection with the faster R-CNN*, CoRRabs/1606.03473 (2016)

[3] **R. Girshick,** *Fast R-CNN, in: Proceedings of the IEEE Conference Computer Vision and Pattern Recognition*, 2015, pp. 1440–1448

[4] **Cristina Stolojescu-Crisan**, *Access Control and Surveillance in a Smart Home*, (2021), *High-Confidence Computing*, pp. 100036

[5] **Waheb A. Jabbar**, *Design and Fabrication of Smart Home with Internet of Things Enabled Automation System*, in: IEEE Access (Vol: 7), (2019), pp. 144059 – 144074

CHAPTER 10

APPENDIX – BASE PAPER

Title: . A fast face detection method via convolutional neural network

Publisher: Elsevier

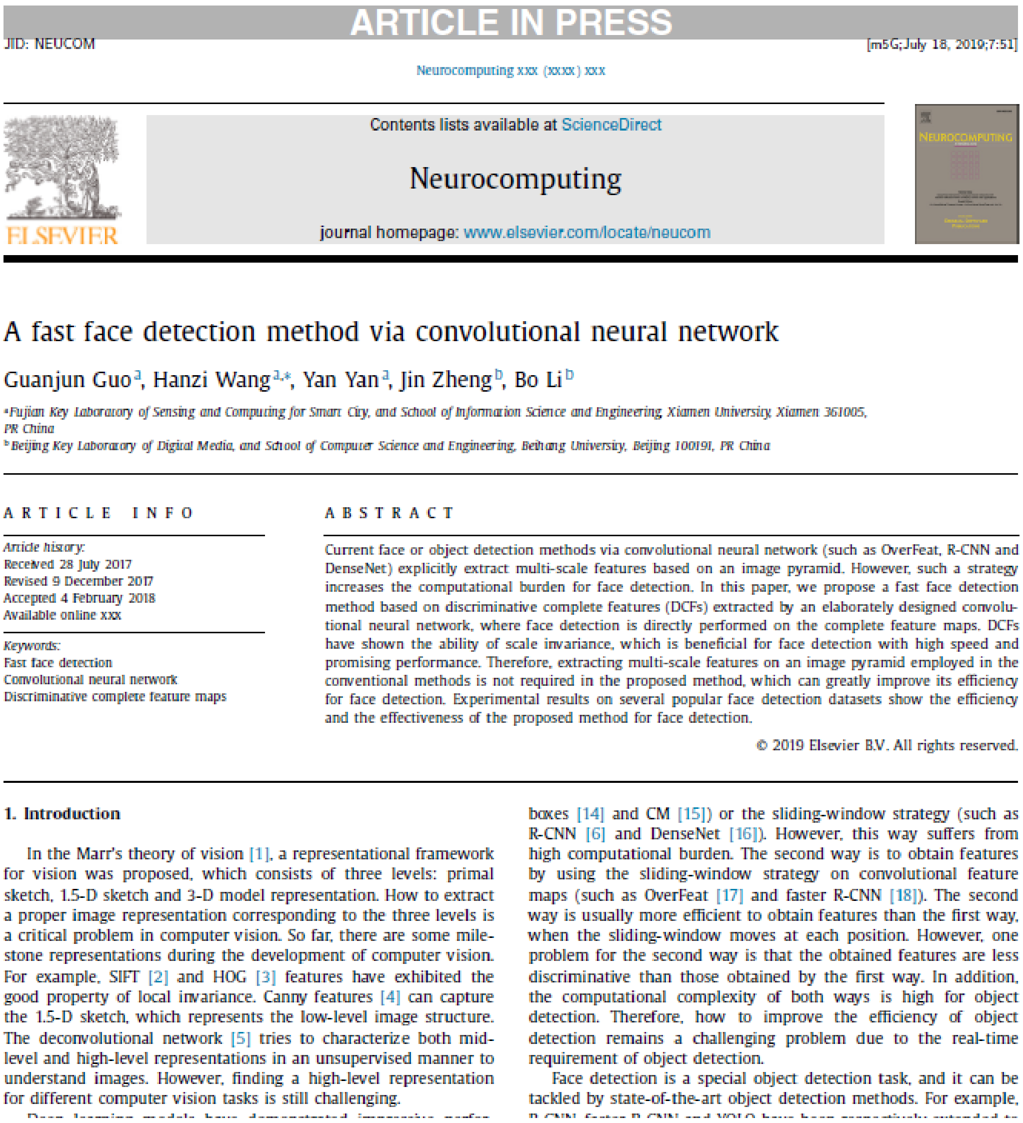
Year: 2019

Journal: Neurocomputing

Indexing: SCI

Authors: Guanjun Guo, Hanzi Wang, Yan Yana, Jin Zheng, Bo Li

Journal website: https://www.sciencedirect.com/science/article/abs/pii/S0925231219309075



ARTICLE IN PRESS

JID: NEUCOM [m5G;July 18, 2019;7:51]

Neurocomputing xxx (xxxx) xxx

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A fast face detection method via convolutional neural network

Guanjun Guo[a], Hanzi Wang[a,*], Yan Yan[a], Jin Zheng[b], Bo Li[b]

[a]Fujian Key Laboratory of Sensing and Computing for Smart City, and School of Information Science and Engineering, Xiamen University, Xiamen 361005, PR China
[b]Beijing Key Laboratory of Digital Media, and School of Computer Science and Engineering, Beihang University, Beijing 100191, PR China

ARTICLE INFO

Article history:
Received 28 July 2017
Revised 9 December 2017
Accepted 4 February 2018
Available online xxx

Keywords:
Fast face detection
Convolutional neural network
Discriminative complete feature maps

ABSTRACT

Current face or object detection methods via convolutional neural network (such as OverFeat, R-CNN and DenseNet) explicitly extract multi-scale features based on an image pyramid. However, such a strategy increases the computational burden for face detection. In this paper, we propose a fast face detection method based on discriminative complete features (DCFs) extracted by an elaborately designed convolutional neural network, where face detection is directly performed on the complete feature maps. DCFs have shown the ability of scale invariance, which is beneficial for face detection with high speed and promising performance. Therefore, extracting multi-scale features on an image pyramid employed in the conventional methods is not required in the proposed method, which can greatly improve its efficiency for face detection. Experimental results on several popular face detection datasets show the efficiency and the effectiveness of the proposed method for face detection.

1. Introduction

In the Marr's theory of vision [1], a representational framework for vision was proposed, which consists of three levels: primal sketch, 1.5-D sketch and 3-D model representation. How to extract a proper image representation corresponding to the three levels is a critical problem in computer vision. So far, there are some milestone representations during the development of computer vision. For example, SIFT [2] and HOG [3] features have exhibited the good property of local invariance. Canny features [4] can capture the 1.5-D sketch, which represents the low-level image structure. The deconvolutional network [5] tries to characterize both mid-level and high-level representations in an unsupervised manner to understand images. However, finding a high-level representation for different computer vision tasks is still challenging.

boxes [14] and CM [15]) or the sliding-window strategy (such as R-CNN [6] and DenseNet [16]). However, this way suffers from high computational burden. The second way is to obtain features by using the sliding-window strategy on convolutional feature maps (such as OverFeat [17] and faster R-CNN [18]). The second way is usually more efficient to obtain features than the first way, when the sliding-window moves at each position. However, one problem for the second way is that the obtained features are less discriminative than those obtained by the first way. In addition, the computational complexity of both ways is high for object detection. Therefore, how to improve the efficiency of object detection remains a challenging problem due to the real-time requirement of object detection.

Face detection is a special object detection task, and it can be tackled by state-of-the-art object detection methods. For example,

Project report

ORIGINALITY REPORT

3%	2%	1%	1%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Infile Student Paper	1 %
2	www.indiatoday.in Internet Source	1 %
3	gammill.com Internet Source	1 %
4	researchr.org Internet Source	<1 %
5	dokumen.pub Internet Source	<1 %
6	"International Conference on Computer Networks and Communication Technologies", Springer Science and Business Media LLC, 2019 Publication	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		