JARVIS: User-defined Postures Detection for Smart Home

Christopher Albert Priatko Theodore Fabian Rudy

Supervisor: Prof. Michael R. Lyu
Computer Science and Engineering
The Chinese University of Hong Kong

January 16, 2023

Table of Contents

Do you have one of these?

Introduction - Motivation

Voice commands limitations

- Accessibility: Can only be accessible in a certain distance from device
- Control: Limited predetermined controls and language issues
- Privacy Issue: Reliance on major software providers

Introduction - Objective

This semester, we aim to do our projects through different objectives that we have

- Research different aspects of computer vision that is viable for the the projects
- Compare the performance between those computer vision
- Implement a working prototype of action recognition with pre-determined gestures

Table of Contents

Literature Review

Object Detection

YOLO, SSD, R-CNN,

EfficientDet, etc

Face Detection

Dlib, FaceNet, OpenFace,

VGG-Face, etc

Pose Estimation

HRNet, ViTPose, OpenPose,

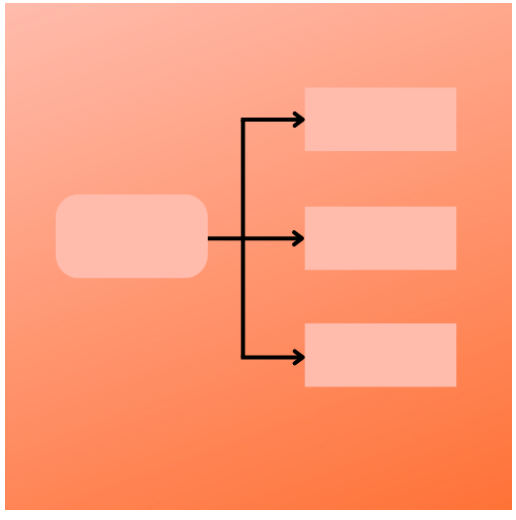
AlphaPose, etc

Literature review (*cont.*)

Action Recognition

PoseConv3D, VideoMAE,

CTR-GCN, HD-GCN, etc



Multi-Task Learning

HRNet, PoseConv3D,

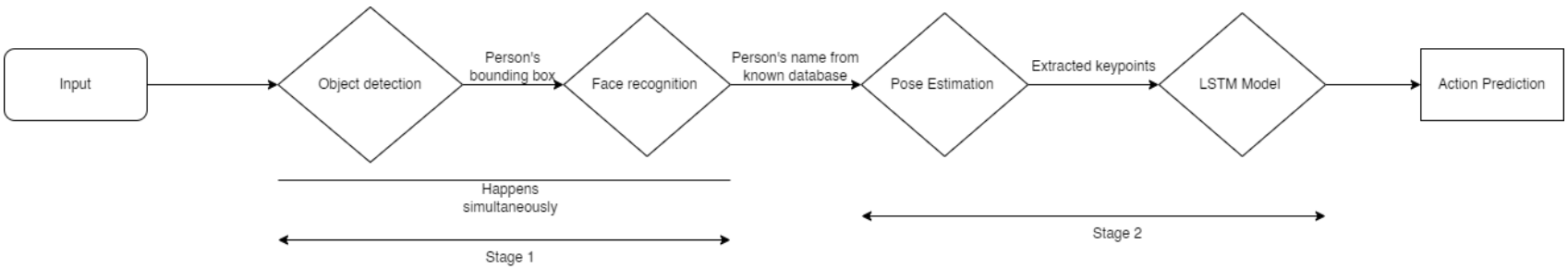
OpenPose, AlphaPose, etc

Issues

- High computational power needed for each library
- Our goal is to have everything run locally
- Performance is our most important aspect

Solution

Two stage framework



Solution

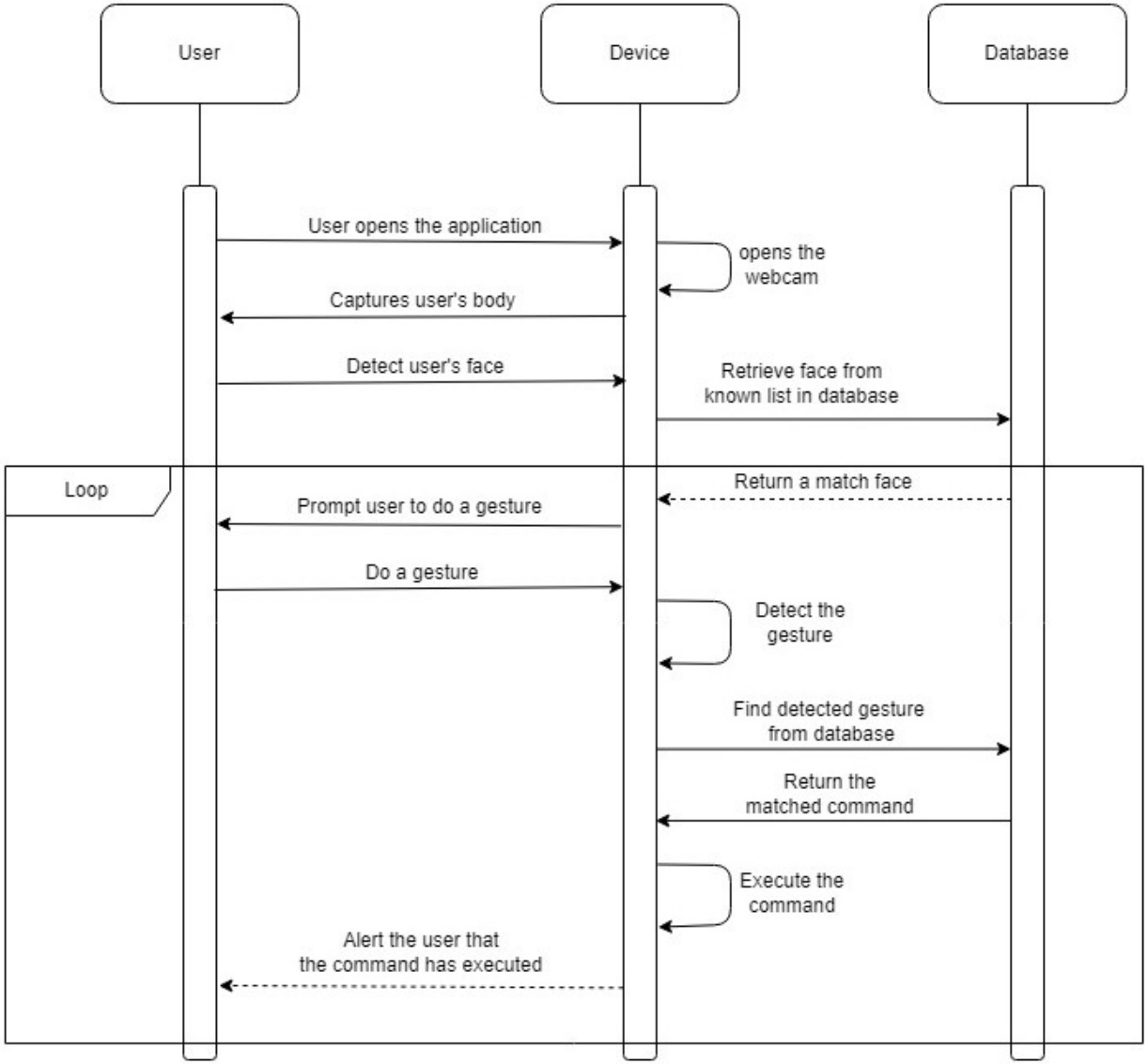

User
Device
Database
User opens the application
opens the webcam
Captures user's body
Detect user's face
Retrieve face from known list in database
Loop
Return a match face
Prompt user to do a gesture
Do a gesture
Detect the gesture
Find detected gesture from database
Return the matched command
Execute the command
Alert the user that the command has executed

Table of Contents

Stage 1 - Overview

Object Detection

Face Detection

Stage 1 - Object Detection

Table: Performance of Object Detection Models

Models	Object Detected	Average FPS
YOLOv5	Person, TV	14
YOLOv7	Person, TV, Chair	10

†Our conclusion:

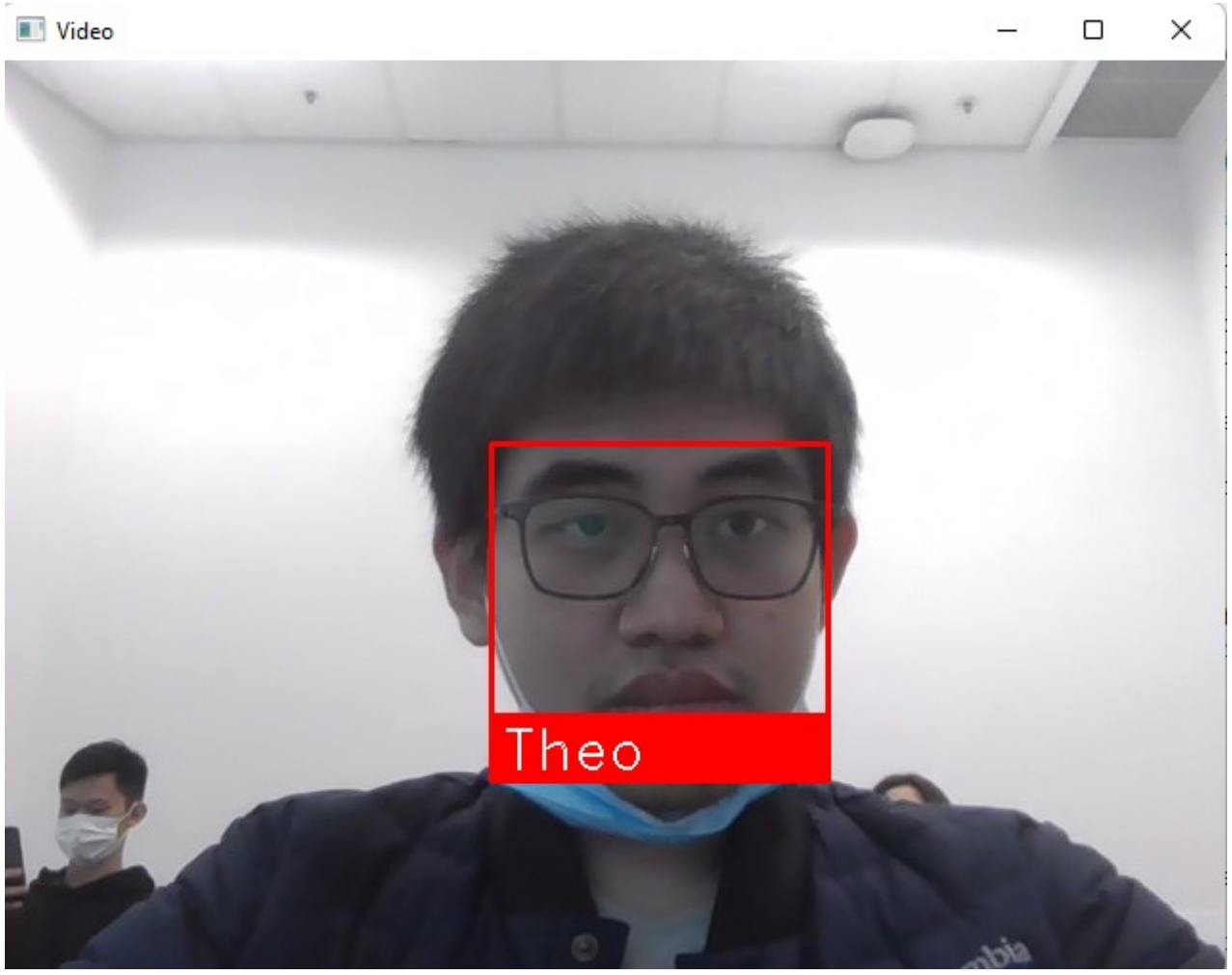
YOLOv5 due to higher FPS and insignificant accuracy difference

YOLOv5 accuracy is still good for our use case (Single Person)

Stage 1 - Object Detection (cont.)

- YOLOv5 uses COCO for its pretrained model, which included 80 different classes (including person)
- Build our own dataset for our project (Explained in data collection)

Stage 1 - Face Recognition



Face Recognition using Dlib

Stage 1 - Data Collection

- Use Open Images dataset for training
- Limit to 2000 images, focusing on person and non-person

Stage 1 - Model

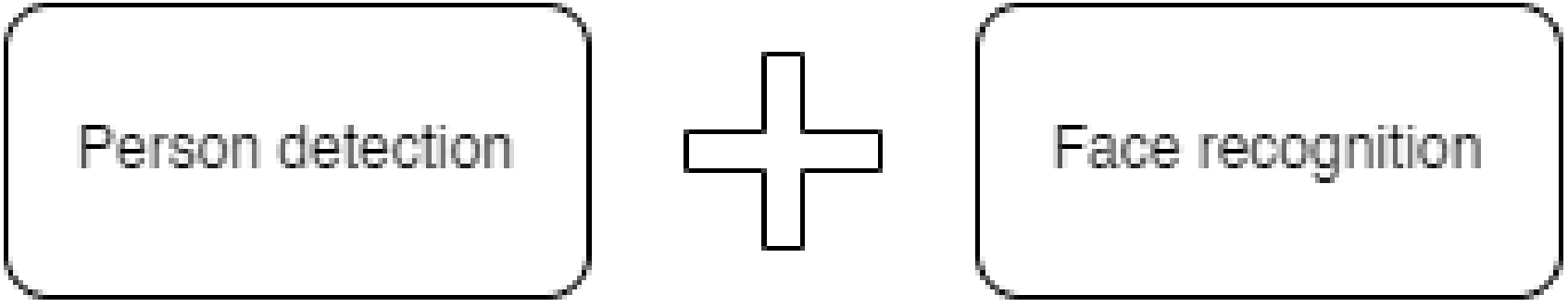
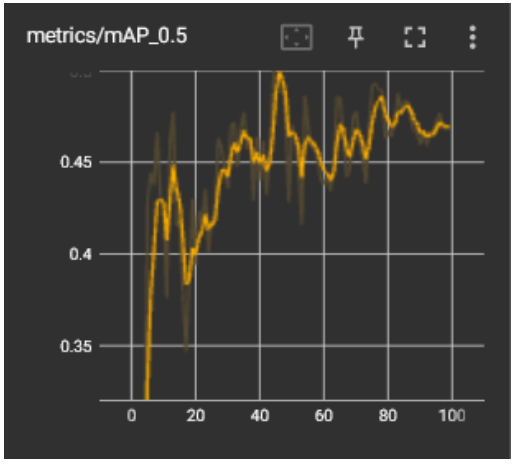
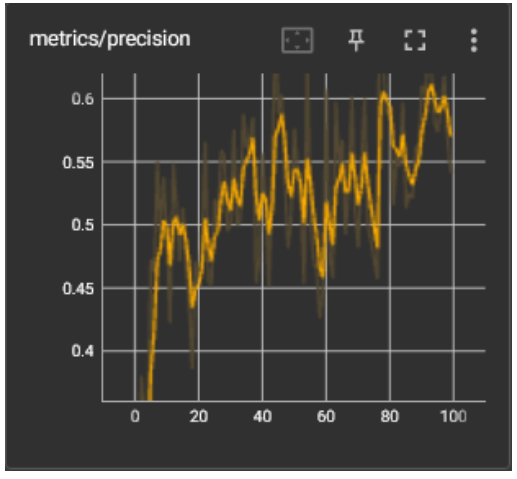


Figure: Model of Object Detection + Face Recognition network

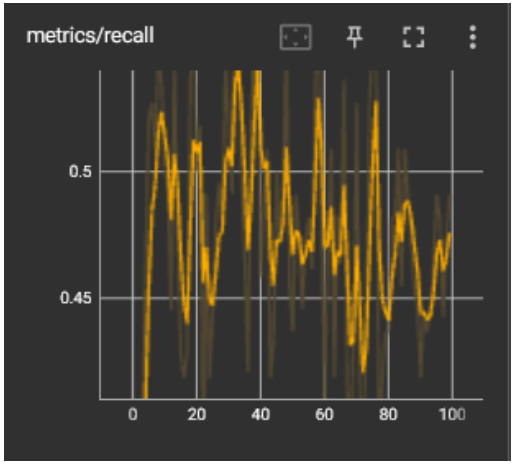
Stage 1 - Training performance



mAP



Precision



Recall

Stage 1 - Training Performance (*cont.*)

Table: Statistics

Phase	P	R	mAP50	mAP50-95
Validation	0.542	0.44	0.419	0.226
Training	0.5168	0.5818	0.5168	0.264

Stage 1 - Demo

Stage 2 - Overview

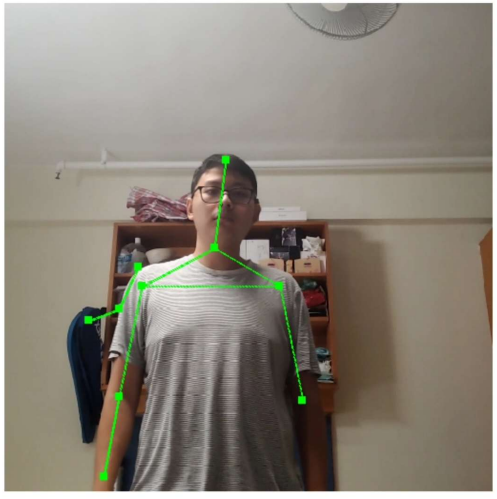
Pose Estimation

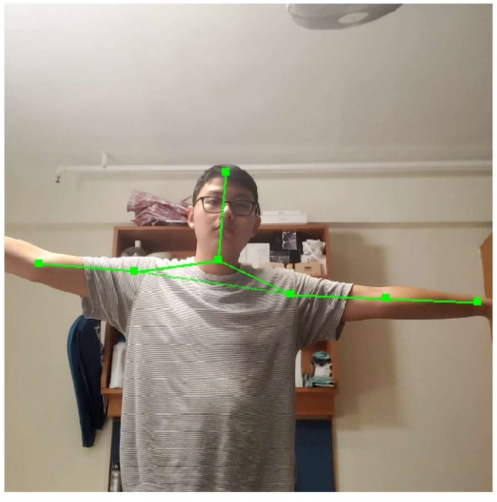
Action Recognition

Stage 2 - Pose Estimation - Sample Model

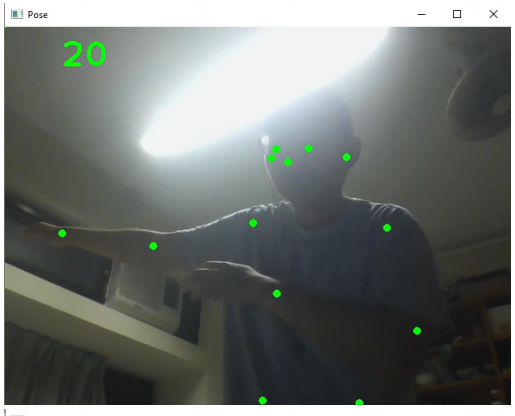
BlazePose

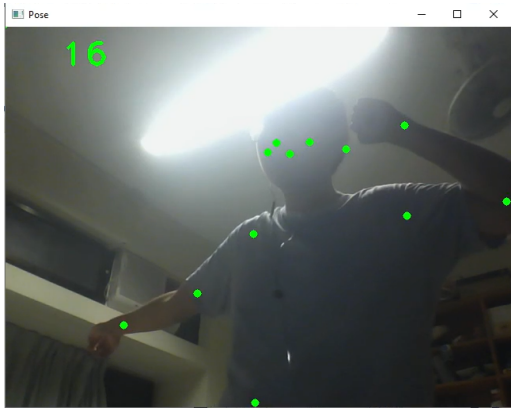
LitePose





MoveNet





Stage 2 - Pose Estimation

Table: Performance of different algorithms

Method	CPU/GPU†	FPS
OpenPose	CPU	0
BlazePose	CPU	15
AlphaPose	CPU	3-4
LitePose	Phone	30-35
Lightweight Openpose	CPU	3-4
MoveNet	CPU	14-16

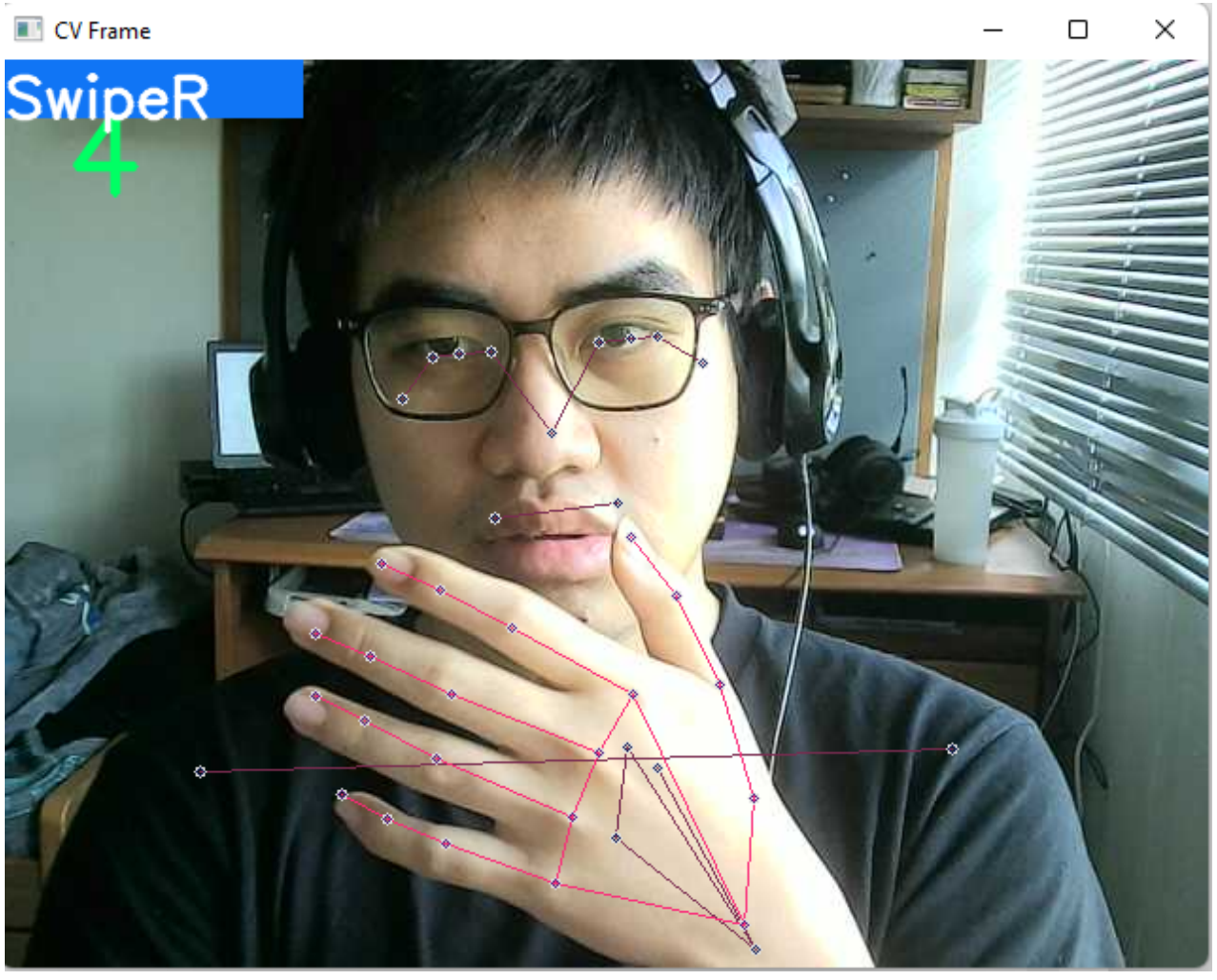
†Devices used:

Phone: Samsung Galaxy Note 10 with Snapdragon 855

CPU 1: AMD Ryzen 5 Pro 3500U with Radeon Vega 8

CPU 2: Intel Core i5-7300HQ with NVidia GTX 1050

Stage 2 - LSTM



Inference run of LSTM-based action recognition

Stage 2 - LSTM (cont.)

- Considering using other action recognition library: PYSKL, VideoMAE
- Problem encountered: High computational power is needed
- Decided to use basic LSTM since it is easier to run on low powered devices

Stage 2 - Data Collection

- Many public datasets available to use: UCF101, Kinetics, Moments
- Decided to build our own dataset with NumPy

Stage 2 - Model

```
model = Sequential()
model.add(LSTM(128, return_sequences=True, activation='relu', input_shape=(30,258)))
model.add(LSTM(64, return_sequences=False, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(64, activation='relu'))
model.add(Dense(32, activation='relu'))
model.add(Dense(actions.shape[0], activation='softmax'))
```

Figure: Model of LSTM network

Stage 2 - Training performance

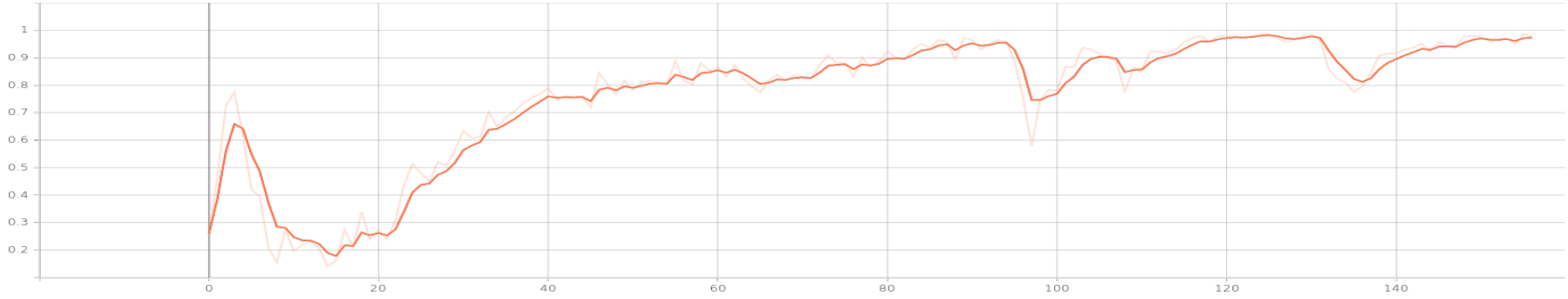


Figure: Training statistics on accuracy

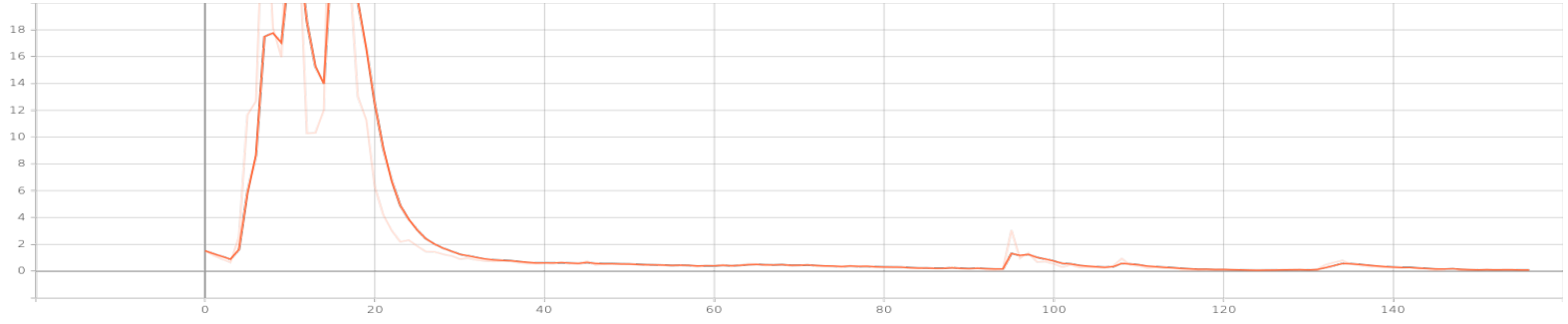


Figure: Training statistics on loss

Stage 2 - Demo

Table of Contents

Conclusion

In this term, we have managed to:

- Researched and compared different computer vision related projects in terms of performance
- Implemented a prototype for object detection with face recognition and action recognition with pre-determined gestures

In the next term, we are planning to:

- Utilize Siamese face recognition instead of Dlib to achieve better performance
- Combine Stage 1 and 2 model to a single, streamlined model
- Optimize the model to achieve best performance

Table of Contents

Q and A Session