JARVIS: User-defined Postures Detection for Smart Home Term 1 Final Report

Theodore Fabian Rudy
Christopher Albert Priatko

Supervised by

Prof. Michael R. Lyu

Computer Science and Engineering
The Chinese University of Hong Kong
January 16, 2023

Contents

Acknowledgement

We would like to express our deepest gratitude to Professor Michael R. Lyu for giving us the opportunity to be our supervisor for our Final Year Project. We would also like to express our deepest gratitude to Mr. Huang Jen-tse for giving us invaluable knowledge, advice and guidance during our Final Year Project.

Chapter 1

Introduction

1.1 Overview

The focus of this final year project is to utilize different aspects of computer vision to process human gestures and/or poses captured by video feed and turn it into command preset in the computer. This report describes the introduction to the topic, the work done in the first semester, and all the implementation challenges that was faced.

1.2 Background

With the development of technology, more of the things that are faced in daily life has become easier. One of the example of that aspect is smart technology. This mechanism turn traditional sensors and actuators to be connected through network, enabling it to increase the capability of achieving centralised control over devices, which brings increase in life quality, well-being, safety, productivity, energy efficiency and many more [56]. Not only that this mechanism allow it to be connected through network, the devices connected to the network will also have the capability to have computing-like functionalities [58].

The evolution of this smart technology happens with more integration of artificial intelligence (AI) into the system. Bowes et al. (2012) described that it was in the second generation smart home technology that artificial intelligence started to be used. They mentioned that the implementation of smart home technology in the generation uses AI-based devices, where it is able to detect changes of environment, monitor health condition and many more [11]. In general, AI was mostly used for analytical functionality and was more described as 'reactive' to a predefined trigger actions. Only in the third generation of smart home technology that AI is more utilised to have the capability to inter-operate with other devices and have multiple functionalities. Not only limited in the usage of behavioural analysis to predict user needs and optimization, but this also enable integration with other devices, making it possible to capture, process and transmit data among devices within network [43]. This generation also mark the emerging system of voice-over interface bringing life to home automation system, enabling it to be more interactive.

With the usage of voice-over interface for smart home system being made available for public by different providers. Google Nest and Amazon Echo being two examples of speech recognition in a form of virtual assistant being implemented. As mentioned in the previous sentence, the functionalities of virtual assistant reduces things that is needed to do by the users, but also to enable people who are disabled or elderly to access and dynamically operate and put control over smart things in the home [76]. Despite its power and capability to control centralised system for smart home, the development for interface for this interaction is pretty much stopped here.

Along with the development of voice controlled smart home system, there are others subjects that earn recognition over its progress and evolution. A subject in particular that has gotten much attention is is computer vision.

Traditionally a subject where the task involved in it are mostly tasks like feature extraction engineered by human in a specific algorithm, its capability of computer to process images and videos, increases significantly. Rather than defining the features and analysis of the input and output, the involvement of deep learning in the subject automate the tasks of feature engineering, extraction and classification into one process involving neural network. The neural network learns the value of features that differentiate one image to another, and automatically adjust the weight and parameter for it to recognise the generalised features and pattern for the thing detected. *Figure 1.1* explains more of the work flow and the difference between them.

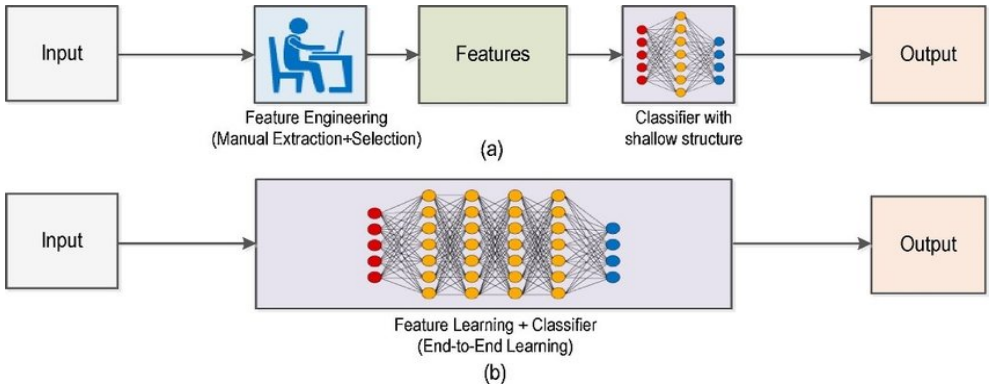


Figure 1.1: Difference in flow of traditional computer vision and modern (deep learning) computer vision, adopted from (Walsh et al., 2019) [69].

With more tasks and applications that are implemented using modern computer vision, the growth of computer vision is formidable. This growth enables a lot of enterprises to implement computer vision tasks, such as autonomous vehicles or so we call it self-driving car. Similar to what was mentioned in the third generation of smart home technology earlier, autonomous vehicles uses different sensors and AI as a part for them to enable the interaction between them with the whole system without human intervention. However, Computer Vision plays a huge part in it, where some of smaller tasks that is able to be

solved through modern computer vision. Despite all this, there are a lot of potential that Computer Vision has to offer, and more methods and techniques used to increase performance, accuracy, precision, and others.

1.3 Objectives

The goal of our project is to utilize computer vision to make virtual assistant to be controllable by gestures, which is customizable by user. By that, we will not only be needing feature limited to gesture detection or in other words, action recognition, but to also employ person detection to understand if a person gets into a frame of the camera. With that too, two stage detection will be needed in this area, for person detection to pass the face detection to the face recognition system. Once the user is recognized, then the user gestures can be detected.

In semester 1, the main focus will be on research and prototype demo.

1. Research different aspects of computer vision that is relatable to the project, including the state of the art methods

2. Comparing the performance of the frameworks

3. Developing a working prototype for action recognition with the frameworks that are known to have the best performance on edge devices, where the model for detecting sequence of poses is also embedded to the framework

At the end of the semester one, we have made the simplified overall architecture for the project, where it is explained too in *Figure 1.2*

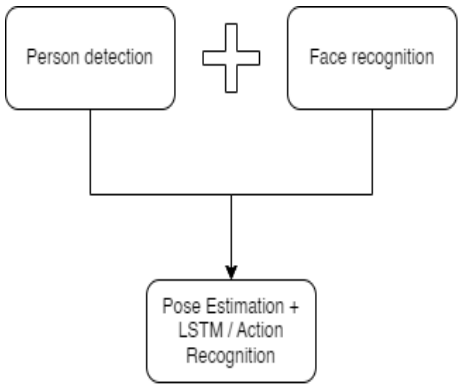


Figure 1.2: Overall architecture design of the project

1.4 Glossary

Terminologies that are not used as much and more specific in term of computer vision or AI explained in *Table 1.1*

Table 1.1: Terms in the papers that are used

Terms	**Explanations**
Action Recognition	A task in computer vision to identify actions of a person identified in the video or image.
Artificial Intelligence	Systems or machines that mimic human intelligence to perform tasks and can iteratively improve themselves based on the information they collect.
Bounding Box	A rectangle determined by coordinates of a certain object detected on an image. Usually used in object detection task to describe the spatial location of an object.

Continued on next page

Table 1.1 – *Continued from previous page*

Terms	**Explanations**
Computer Vision	A field with the focus of enabling computers to process and understand visual side, including videos and images.
Convolutional Neural Network	A Deep Learning that assume input as images, with the capability to differentiate the objects that contained in the image.
Deep Learning	A part of artificial intelligence and machine learning where neural networks are designed to learn and improving on its own by examining the algorithm. This remove the need to supervise the training process in machine learning.
LSTM	A recurrent neural network that has the ability to not only process a single frame of image or a single data point, but also entire sequences of data.
Machine Learning	A part of artificial intelligence utilizing the use of data for the algorithm to learn a certain task, gradually improving its accuracy in the task.
Neural Network	A computing system that have the resemblance of human brain, mimicking the way brain send neural signal to one another. Each signal passed in the system will adjust the weight on each neurons. This weight output from one neuron will affect the next node in the connection.

Continued on next page

Table 1.1 – *Continued from previous page*

Terms	**Explanations**
Object Detection	A task in computer vision designed to detecting instances based on the features available to classify the class in digital images and/or videos.
Pose Estimation	A task in computer vision designed to predict and track the location of a person or object. For human pose estimation, the things that are tracked are keypoints of human joint, where its unity will resemblance of a human pose.
Recurrent Neural Network	A type of neural network where the architecture allow it to have a cycle in between. This allows output of the network to depend on the prior elements within the sequence. This technique is used for ordinal or temporal problem, such as natural language processing, image captioning, language translation, etc.
YOLO	An abbreviation for You Only Look Once, a revolutionary object detection system that utilize deep learning to achieve real-time object detection.

Chapter 2

Related Work

2.1 Object Detection

The interest in object detection started in 2012, during the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [38]. This competition focused on building an efficient object detection algorithm using the ImageNet database as the training dataset for the algorithm. In this competition, Krizhevsky et al. (2012) has decided to approach the challenge by building an object detection method using Convolution Neural Network (CNN), called AlexNet. With CNN, AlexNet manage to perform much better compared to its rivals, hence the rise of CNN on object detection algorithm.

Furthermore, research on Deep Neural Networks has accelerated after ILSVRC 2012, and one improvement they have made is on the training. Since deep neural networks are harder to train, Kaiming et al. [29] has developed a residual learning framework to ease the training of deeper neural networks. Also, the residual networks manage to gain accuracy from the increase depth. The framework has helped on development of deep neural network, especially for many visual recognition tasks.

2.1.1 Two-Stage Detection Framework

There are two famous framework on CNN-based object detection algorithm. The first one is two-stage detection framework. Two-stage detection framework works by using region proposals to create Region of Interest (ROI) [25]. These ROIs then converted to feature maps using CNN, which then resize to extract the features that will be used by SVM for object detection and bounding box regressor to adjust the bounding box size. This method is more time efficient since the algorithm does not need to detect all parts of an image; instead, it focuses on parts that have high chance of an object present. This method created a model family called R-CNN model family. In this project, we are analyzing two examples from the family, which are R-CNN and Fast R-CNN.

R-CNN is the first CNN-based object detection algorithm after Krizhevsky's algorithm. [24]. The algorithm works in 3 different modules. The first module generates around 2000 region proposals. After that, for each region proposal, the algorithm extracts a dimensional feature vector using CNN. Finally, the algorithm utilize SVM to classify the object present in the region proposal. This algorithm achieved better accuracy than Krizhevsky's algorithm; however, R-CNN still has several drawbacks, mainly in the training and testing speed. Since it uses feature extraction from each object proposal and we have around 2000 region proposals, it consumes a lot of space, roughly hundreds of gigabyte of storage. Furthermore, during testing, the algorithm is slow to detect the object in an image, taking around 47 seconds per image.

Due to these issues present in R-CNN, Ross Girshick, the creator of R-CNN, decided to modify parts of R-CNN to improve the performance and make the algorithm more time efficient, which he called Fast R-CNN (Fast Region-based CNN) [23]. Fast R-CNN utilize similar object detection algorithm as

R-CNN; however, with Fast R-CNN, instead of using 2000 region proposals to extract feature vectors with CNN, Fast R-CNN only uses one fixed-size region that has been through Region of Interest (RoI) pooling to extract feature vectors. Furthermore, Fast R-CNN uses a more optimized softmax classifier and bounding box regressors in one stage instead of a softmax classifier, SVM, and regressors in three different stages. Because of this, Fast R-CNN manages to reduce the time for object detection and model training, going from 84 hours to 9.5 hours, while simultaneously keeping similar or in some cases better accuracy when compared to R-CNN.

Besides aforementioned algorithms, there are other algorithms that is derived from the R-CNN family. Algorithms such as Faster R-CNN[52] and T-CNN[34] aims to improve the performance of R-CNN based algorithms. Faster R-CNN improves the performance on Fast R-CNN algorithm by using deep convolutional network[28] to determine the region proposals instead of selective search[65], used in Fast R-CNN. Whereas T-CNN focuses on utilizing temporal and contextual information such that it improve the algorithm performance when applied to videos.

2.1.2 One-Stage Detection Framework

Although Fast R-CNN manage to greatly improve the performance of R-CNN, it still struggles with lengthy training time. With that, a new architecture for object detection has arised, in the form of one-stage detection framework. While two-stage detection framework has two stages for the object detection (using region proposals, object classification, and bounding box regression), one-stage detection framework only uses feature extraction network and convolution layers for predicting the object and adjusting the bounding box. This meant that one-stage detection framework uses less stages for object detection, and thus

improve the performance of the algorithm. However, there are problems in one-stage detection framework, mainly in its inability to detect smaller objects. In this project, we are analyzing two of the most famous one-stage detection framework model, Single Shot Multibox Detector (SSD) and You Only Look Once (YOLO).

Single Shot Multibox Detector (SSD) is one of the object detection algorithm that utilize one-stage detection framework [40]. It uses VGG16 Network for feature extraction and several convolution layers for detections, predictions, and bounding box adjustments. By using feature extraction and convolution layers for its model, SSD manages to achieve higher mAP in PASCAL VOC2012 and COCO dataset compared to any R-CNN model family. At the same time, SSD manages to achieve higher performance than Faster R-CNN (46 FPS vs 7 FPS).

Another famous one-stage detection framework model is You Only Look Once (YOLO) [51]. The model is designed based using 24 convolutional layers followed by 2 fully connected layers. It also utilize DarkNet during training. The model used in YOLO is similar R-CNN, where it proposes potential bounding boxes and score it using convolutional features. However, unlike R-CNN, YOLO only uses 98 bounding boxes instead of 2000. Furthermore, YOLO model has spatial constraints to help mitigate multiple detections of the same object and it combines individual components to a single optimized model, unlike R-CNN. This means that YOLO manages to achieve higher performance than Faster R-CNN (45 FPS vs 7 FPS); however, it comes at slightly lower accuracy, with around ten percent lower mAP compared to Faster R-CNN (73.2 vs 63.4).

2.1.3 Other Methods

Although two-stage and one-stage detection framework model are mostly used for object detection, there are some popular object detection algorithms that uses neither framework. Yang et al. [74] propose a new method of using multi-task learning [53] to improve the performance of existing object detection method. To demonstrate that, the research uses YOLOv5 as the base of this model, with the addition of multi-task learning such that this model is able to perform object detection and semantic segmentation at the same time. This model manages to achieve higher accuracy (around 9 - 11 percent higher) compared to other semantic segmentation algorithm such as PSPNet [78], BiSeNet network [75], and ASPP network [15].

2.2 Face Recognition

Face recognition (FR) is a method of detecting and identifying a person with their face and it has been used in many areas[71]. Interest of FR started with the introduction of Eigenface approach [64]. It developed to research of FR in multiple areas, from low-dimensional representation to local-feature based [71]. However, in 2012, a new approach of FR using deep learning has started with the winning of AlexNet [38]. Several famous Deep-learning based FR [7, 49, 62, 57, 36, 55, 5] are reviewed in this paper. BlazeFace [7] develops its' model from [40] with additional modifications such as enlarging receptive field size compared to other neural network architectures (such as MobileNet [[32],[54]]), using feature extractions *B*lazeBlocks, reducing the Pooling Pyramid Network architecture [33], and implement suppresion algorithm to reduce error from larger spatial resolution. YOLO5Face [49] redesign YOLOv5 to be used for face recognition with modifications to the architecture such as the use of SPP, PAN, and SILU

activation function. DeepFace [62] implements a deep neural network and affine transformation to the align and represent step in typical FR framework to reduce error. LightFace [57] implements the typical face recognition framework in the background in TensorFlow and Keras to make it lightweight. It also provides mainstream face detection models [36, 55, 5] for users to use. Dlib [36] is a toolkit developed to build machine learning models with functions such as classifications and regressions. It also has several functions for FR and face detection in Python. FaceNet [55] removes the intermediate bottleneck layer used in previous deep-learning based FR and uses a trained deep convolutional network to optimize, improving the performance of the model. OpenFace [5] uses CLNF [4] as the base of OpenFace model, with some changes. These changes include: adding a validation step using CNN, using separate point distribution set for eyes, lips, and eyebrows, and additional features such as head pose and eye-gaze estimation.

2.3 Pose Estimation and Keypoints Analysis

Pose Estimation and Keypoints Analysis has been an interest in computer vision. This subject focuses on how to predict and track the position of a person, either in an image or a video. During the last few years, there have been many advances in this subject and new pose estimation algorithms are being made, either to improve the performance or accuracy of pose estimation. Currently, the most famous approach of pose estimation are top down approach and bottom up approach. We will discuss both strengths and weaknesses of both approach in this paper, and show some examples of it.

The top down approach relies on the model detecting the presence of hu-

man being in a frame, then determines the keypoints to outline the pose, with the advantage in accuracy but suffers in performance. Examples of top down approach in pose estimation [21, 20, 70, 2, 42, 26, 50, 6, 60] are reviewed in this paper. RMPE [21] utilize CNN for pose estimation and Pose NMS for pose redundancy elimination, which is implemented and improved in AlphaPose [20] by using SIKR to accurately localize keypoints and Pose-aware Identity Embedding to simultaneously track poses. HRNet [70] focuses on making a pose estimation and object detection using high-resolution images by making high-resolution convolution stream. YOLOPose [[2],[42]] build their model by using YOLO [51] as the object detection framework; however, both differ in their pose estimation. Amini, et al. [2] utilize a 6D Transformer to perform pose estimation and keypoints regression [1], whereas Maji, et al. [42] utilize loss function to perform pose and keypoints estimation. Both algorithm shows noticeable accuracy improvements compared to similar algorithms, although both are incomparable to each other due to their objective and dataset difference. DensePose [26], tackles the same issue with a different approach. Where most models focus on developing their framework first, DensePose focus on gathering a dataset, consisted of dense correspondences between SMPL model [41] and person appearing in COCO dataset, then use the resulting dataset to build a CNN object detection model, based on Mask R-CNN [27]. Improvements were made in another version of DensePose [50] where they create a lighter, less layer R-CNN model, which resulted in increase in model performance. BlazePose [6] differs from typical top down approach models by detecting the torso or face instead of whole body. To complement this approach, Bazarevsky et al. utilize a fast face detector[7] and design a pose estimation model partly inspired by Stacked Hourglass approach [46]. Pose Estimation Transformer (POET) [60] extends the work done in DETR [14] with the addition of a transformer-based

architecture to predict human poses in parallel and a set prediction loss that is a linear combination of simple sub-losses for classes, keypoint coordinates, and visibilities. It contains three main elements, a CNN backbone using ResNet [29], a encoder-decoder transformer based on [14, 67], and pose prediction head.

Bottom up approach relies on the model detecting human keypoints, then classify which keypoints belong to the same person and connect them. The method has an advantage of performance compared to top down approach, but suffers from accuracy. Examples of bottom up approach [17, 72, 3, 22, 12] are reviewed in this paper. HigherHRNet [17] uses similar backbone as HRNet [70] with addition of high-resolution feature pyramid to predict high-resolution heatmaps that are beneficial for detecting small person, which is more efficient compared to HRNet. LitePose [72] provide more improvements to HRNet [70] and HigherHRNet [17] by converting multi-branch architecture in HigherHRNet to single-branch architecture using gradual shrinking, removing redundancies found in HigherHRNet. Furthermore, LitePose also use fusion deconv head and large kernel convolution to enhance its' capacity. MoveNet divides the model into three parts: encoder, mapper, and decoder, and uses a combination of Convolution, Max Pooling, Dense, and Upsampling layers in the model. Furthermore, it utilizes LeakyReLU as the activation function after each convolution layer. Disentangled Keypoint Regression (DEKR) [22] utilize disentangled representations [8] for the representation to accurately learn the keypoint region, therefore the predicted keypoints are inside the keypoint regions. OpenPose [12] utilize Part Affinite Fields (PAF) for body parts associationin pose estimation. The research is based on [13], with the modifications on the model mainly in PAF, by increasing network depth but removing body part refinement stage, improving performance and accuracy of the algorithm.

With the massive amount of research conducted around pose estimation, it

is noted that the models made are getting increasingly more complex. Because of this, Xiao et al. [73] researched the possibility of making simple baselines for pose estimation models by using a simpler model, consisting of a few deconvolutional layers added on a backbone network, ResNet, similar to the one in [29]. The research showed that the simple model is capable of pose estimation with a promising result, managed to perform slightly better than the winner of COCO2017 keypoint Challenge's models. The result of this result showcased that a simple model can be as effective as a complex one in pose estimation, which our research will keep in mind during building our model. Furthermore, another research [45] compares two popular pose estimation model from top down and bottom up approach, BlazePose [6] and OpenPose [12], and found out that BlazePose manage to surpass OpenPose in terms of real world performance. This research demonstrate that, even though bottom up approach are generally has better performance compared to top down approach, with the right architecture, a top down model can surpass bottom up model in terms of performance.

2.4 Action Recognition

Action recognition in computer vision focuses on the ability to identify human action in videos [30]. This task is challenging due to the need to determine when the action started and ends. The interest started with the usage of holistic features [10], then progressed to local features, and finally with deep learning with [38] popularize the technique. Recent works of deep learning-based action recognition [16, 19, 39] are discussed in this paper. Channel-wise Topology Refinement Graph Convolution Network (CTR-GCN) [16] improves on Graph Convolution Network model by simultaneously learning the topolo-

gies and channel-specific correlations using channel-wise topologies. PYSKL [19] is a open-source toolbox, based of PyTorch, and designed to support different action recognition models from GCN and CNN, such as CTR-GCN [16] and 3D Convolutional Networks [63]. Hierarchically Decomposed Graph Convolutional Network (HD-CGN) [39] creates sets of joint nodes for edge extraction, highlights the dominant edge sets, and apply a new ensemble method that uses only joint and bone stream.

Some researches [37, 61] focus on hand gesture detection and recognition as part of action recognition. Kopuklu et al. [37] create a hand gesture detection using lightweight CNN and hand gesture classification using deep CNN. Sung et al. [61] improves on an existing hand skeleton tracker model, *M*ediaPipe Hands [77], with modifications to improve keypoint accuracy and the ability to do a 3D keypoints estimation. And for classification, it combines heuristic methods and neural networks to classify gestures.

2.5 Long Short-Term Memory

Unlike the other sections discussed in this material, Long short-term Memory (LSTM) is not a part of computer vision. LSTM is a neural network, part of AI. Rather than continuously passing input that is processed in the previous layer, LSTM as the part of Recurrent Neural Network, allows feedback connection. This feedback connection allows the network to process entire sequences of data, with example of speech or video. The architecture of this neural network also allows the unit to protect memory content, while also processing activation pattern step-by-step and add them to the memory after processing [31]. This behaviour allows the network to process short-term memory that can last for a lot of steps. LSTM network are well suited for issues that can be solved

by gradient-based approaches, such as classification, prediction on time series and many else. Despite it's older age compared to most techniques, LSTM is proved to be effective, even for the recent implementations. In 2018, one team consisted of five neural network trained on a single layer, 1024-units LSTM model is trained to play a game of Dota 2, a game that is well known for a complex mechanics and strategies and is able to beat teams of amateur human teams [47, 9]. In 2019, an Artificial Intelligence called AlphaStar, built on LSTM network beats one of the world strongest professional player in one of the most complex video game, Starcraft 2, won convincingly with the score of 5-0 [68]. This prove that this network is still well used in the industry. Not only that this model is able to be used in time series prediction, but it is also used in human action recognition, speech recognition, rhythm learning and music composition, handwriting recognition, market prediction and more tasks that is able to be solved through artificial intelligence, especially healthcare.

Chapter 3

Implementation

3.1 General Framework

With the object detection and pose estimation model sorted, we have decided on a general framework for the model we use on our project, shown in *Figure 3.1.* The flow of the model will go as follow

1. A person enters the Field of View of a camera-equipped smart home device, making them the input for our model

2. The model will detect the person and using object detection, it will create a bounding box around them

3. The model will scan the image inside the bounding box and using facial recognition, it will search for a face inside and match it to a database

4. When the person's face match with the face in database, the model will use pose estimation to extract keypoints

5. With the extracted keypoints, the model uses LSTM for action recognition

6. With the action recognize, the program will do the command related to the action

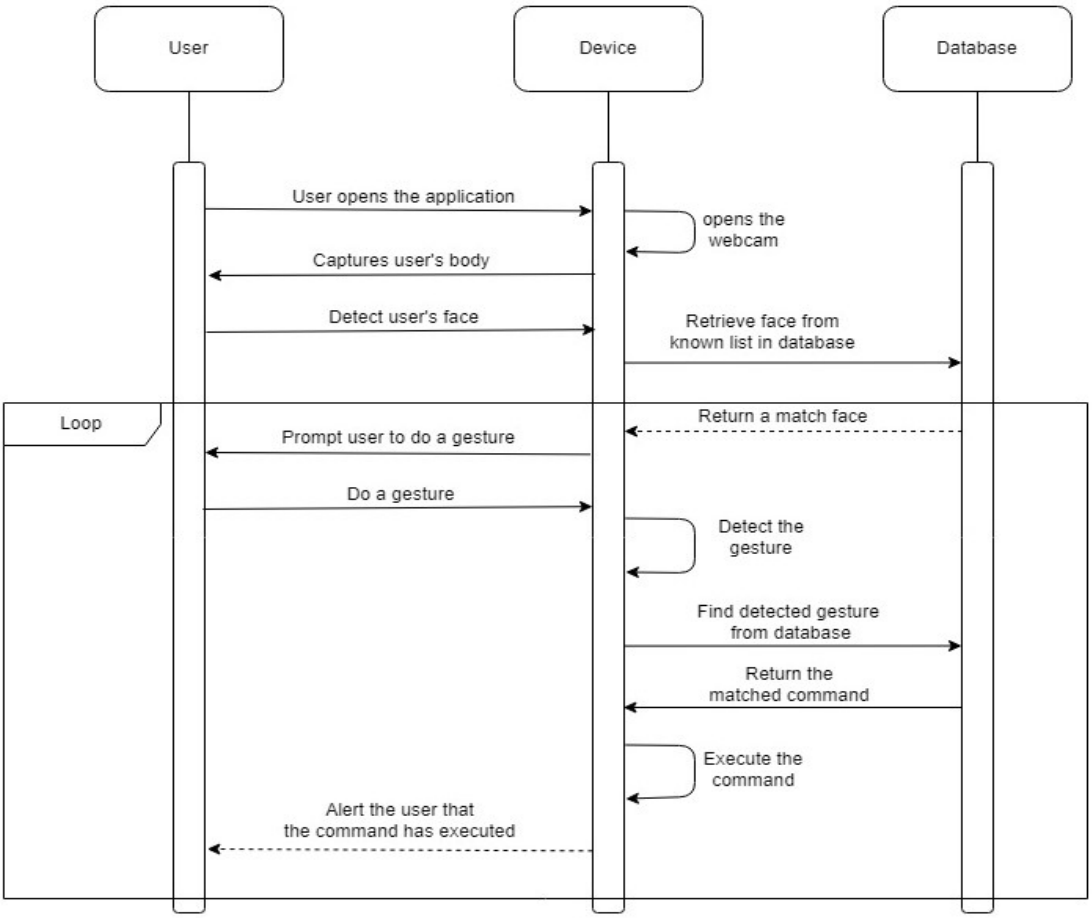


Figure 3.1: Sequence Diagram of the General Framework

From *Figure 3.1*, initially, we are planning to create a single stage model, with all components of the model (Object detection, Face recognition, Pose estimation) in one, streamline pipeline. The problem comes during combining of these components. Previous researches combines either object detection and face recognition [49] or object detection and pose estimation [[42],[2]], but not both of them at the same time. Here, we began the research more on multi-task

learning, as a part that is often used by different researcher to solve this issue.

Multi-task Learning in itself is a subsidiary of machine learning, where multiple task are learnt together in parallel through a shared model. This approach reduces the data that are needed to train machine learning tasks, reducing overfitting and allows optimization on training on related tasks. Not only that, but multi-task learning also allows model for different tasks to be trained faster, and has the capability to reuse the previously known to augment the model for more complex tasks. There are different methods of multi-task learning, with some of the most used one named task grouping and overlap, and transfer of knowledge. Because of its flexibility, multi-task learning are used in different tasks that are solvable through deep learning. In Computer Vision, multi-task learning are mostly used in changing the architecture, for the architecture to partition the network into task-specific and allows generalization while minimizing negative transfer [18].

In some cases, multi-task learning can hinder the performance of the model. Depending on the tasks, multi-task learning can be outperformed by single-task learning. Furthermore, tackling a large and diverse task at the same time can be a challenge for multi-task learning, shown in [66]. This meant that depending on the tasks, using multi-task learning might be less efficient than just relying on single-task learning. Furthermore, research that involve multi-task learning applies it only on the the head of the architecture, consisting of at most 2 or 3 tasks. Because of that, it is not yet possible for us to build a whole architecture using multi-task learning, that is capable to accommodate every task inside the architecture.

With multi-task learning not as efficient and more time-consuming than we first thought, we decided to split the framework into 2 stages, as shown in *Figure 3.2*. The 2 stages mentioned are: Object Detection and Face Recognition in

stage 1, and Pose Estimation and Action Recognition in stage 2.

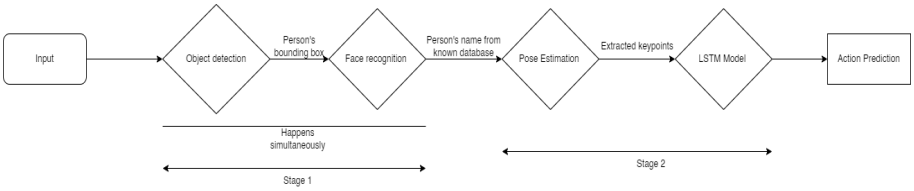


Figure 3.2: General framework of our project

3.2 Stage 1: Object detection and Face Recognition

3.2.1 Object Detection

With the goal of human detection, we have selected potential object detection models that is suitable for our research, which are YOLO, Faster R-CNN, and SSD. Furthermore, since our objective is to implement the model on smart home devices, our main goal is to find the most efficient model that is available for real-time detection. Based on previous research, we have decided to utilize YOLO for object detection due to the its' performance. Compared to R-CNN, YOLO is much more suitable and provides much better real-time performance. Whereas with SSD, although SSD provides slightly higher accuracy compared to YOLO, we ultimately chose YOLO due to better performance. In the middle of our research time, we also discovered that there is a newly published paper and repository on the newest iteration of YOLO, YOLOv7. Before, we have been doing trials on YOLOv5, which is the current iteration of YOLO with the most update and community support, despite not having an official paper. This means there are 2 versions of YOLO that we can use in this project and because of that, we have decided to do inference runs on both models, YOLOv5 and

YOLOv7, and we will evaluate which model suits our project the most based on speed and accuracy.

3.2.2 Face Recognition

Face recognition is necessary to identify the person. There are a few face recognition models we are able to choose for our project [55, 49, 36, 5]. In the end, we decided to implement dlib [36] to our framework as the face recognition model. We implement face recognition system that is available in dlib due to its' simple model, which is beneficial for our performance-minded framework, and the built-in face recognition functions is sufficient for our project.

3.3 Stage 2: Pose Estimation and Action Recognition

3.3.1 Pose Estimation

Another crucial part of our model is pose estimation. Similar to object detection, we began to find different papers and implementations that are available on GitHub. Our criteria in choosing the libraries of implementations and papers are

1. Performance in Low Performance Device (using CPU)
2. Accuracy and Precision
3. Maturity of the model
4. Existing paper and conference entry of the paper
5. Community support on the model

Ultimately, we managed to get several pose estimation models that suits our criteria, such as AlphaPose, OpenPose, LitePose, Yolo-Pose, BlazePose, HR-Net, HigherHRNet, Mo0veNet, PoseResnet, and Lightweight Openpose. The models chosen have different arch9itecture, with a good combination of top-down approach and bottom-up approach, as shown in *Table 3.1*. This proves that, contrary to popular belief regarding bottom-up approach has better performance than top-down approach, there are top-down approaches that can match the performance of bottom-up approaches. To mimic the goal of this research, we have decided to run inference of the aforementioned models on our machines. However, in that stage, we encounter some issues about some of the models.

Table 3.1: Approaches of different pose estimation tools

Method	**Model architecture**
HRNet	Top-down
HigherHRNet	Bottom-up
BlazePose	Bottom-up
AlphaPose	Top-down
LitePose	Top-down
Openpose	Bottom-up
PoseResnet	Top-down
YOLO-Pose	Top-down
Lightweight Openpose	Bottom-up
Movenet	Bottom-up

3.4 Problem encountered

During our inference testing, we encounter several issues with some of the pose estimation models. HRNet, HigherHRNet, and LitePose utilize similar archi-

tecture for their models, and from their research paper and GitHub page, they develop their models using multiple NVidia GPUs. This meant it is impossible for us to do an inference testing of their models on our machine. We have tried to do inference run of the models on CUHK CSE GPU Server, and the issue still persists. Although LitePose has a mobile version, we could not find a source code for it, hence we could not replicate it to run on our machine. Furthermore, some models are unable to run inference on our machine, hence we remove it from our inference testing. For models that manage to run, we have done an inference run and use their pre-trained model to measure the performance. From these constraints, we have decided in testing the performance of 6 models [12, 6, 20, 72, 48, 3].

3.4.1 Action Recognition

Lastly, action recognition is important for our project since it gives the machine ability to understand human's movement. Although there are several action recognition models researched and available to use on GitHub, we found that it is insufficient to fulfill our main objective due to the limitations in the models. Because of that, we have decided to build our own action recognition model, using LSTM to accurately track and predict the movement. Besides that, we also provide some pre-determined poses for LSTM to predict it, using numpy file to store the pre-determined poses.

Chapter 4

Evaluation

4.1 Stage 1

4.1.1 Object Detection

With our concerns regarding YOLOv5 and YOLOv7, we decided to do inference testing on both models to determine which model suits the best for our use case.

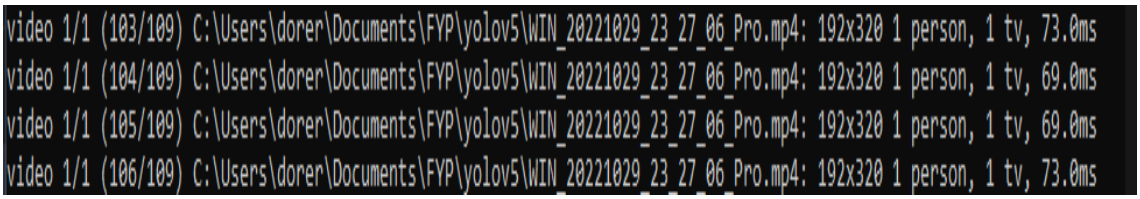


Figure 4.1: Inference testing using YOLOv5

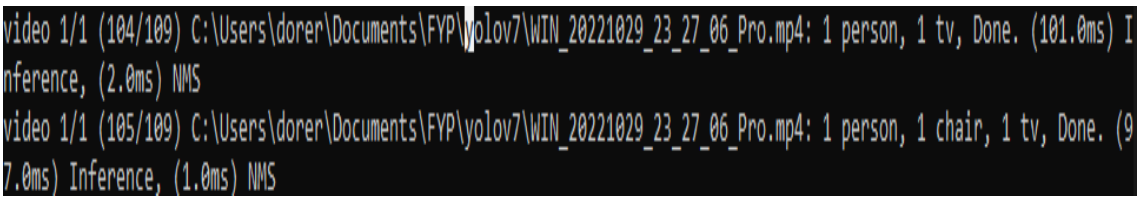


Figure 4.2: Inference testing using YOLOv7

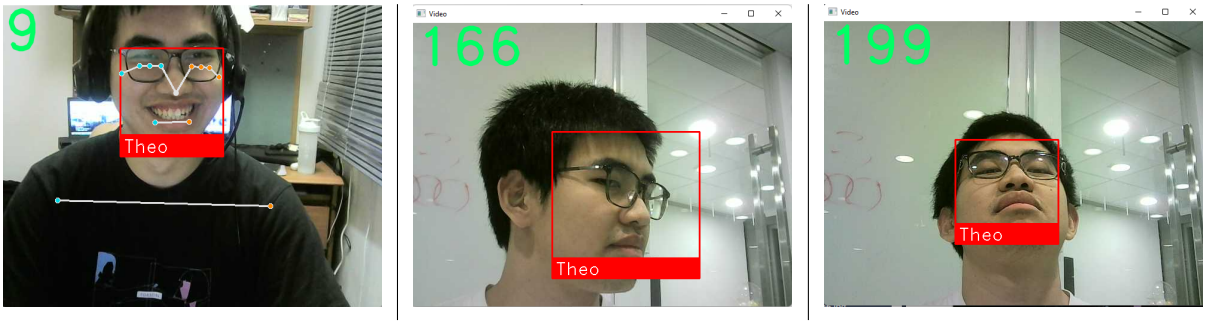
From *Figure 4.1* and *Figure 4.2*, we can see that YOLOv7 manages to

pick up a detail that YOLOv5 miss (A chair), which indicates that YOLOv7 has higher precision than YOLOv5. However, it is slower when compared to YOLOv5 (70ms vs 100ms, both models are using smallest possible pre-trained model) during inference testing. Because our project mainly focus on the performance of the models instead of precision or accuracy, we decided to use YOLOv5 for our project.

4.1.2 Face recognition

Early on, we figured out that the face recognition that is based on dlib is really easy to use. What we need to do is only to put a picture of a person into a known list, then the algorithm will draw similarities that is read on the camera through OpenCV. From there, it is able to the compare it and give annotations on the face that is detected on the camera.

Table 4.1: Sample of Face Recognition



One issue that we found on this part is that the face encoding that is detected might delay the performance of the framework in stage 1, since it takes some time to detect the face. Not only that, but this face detection is pretty inconsistent in time given too. One idea that might be viable for the next development is to just take a picture of someone's face and return the name directly rather than giving bounding box to someone's face, as it would

lighten the weight of stage 1.

4.2 Stage 2

4.2.1 Pose Estimation

For models that manage to run, we have done an inference run and use their pretrained model to measure the performance. *Table 4.2* shows the performance of these models, with Frames per Second (FPS) as the measurement of performance (Higher FPS equals to better performance). We notice that OpenPose, Lightweight OpenPose, and AlphaPose are not suitable for our use case due to the low real-life performance (0 and 3-4 FPS) and all three models are more suitable for devices with a dedicated GPU, which is not the intended target of our research. Because of that, for our research, we have decided to focus on the remaining 3 models, LitePose, BlazePose, and MoveNet.

Table 4.2: 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

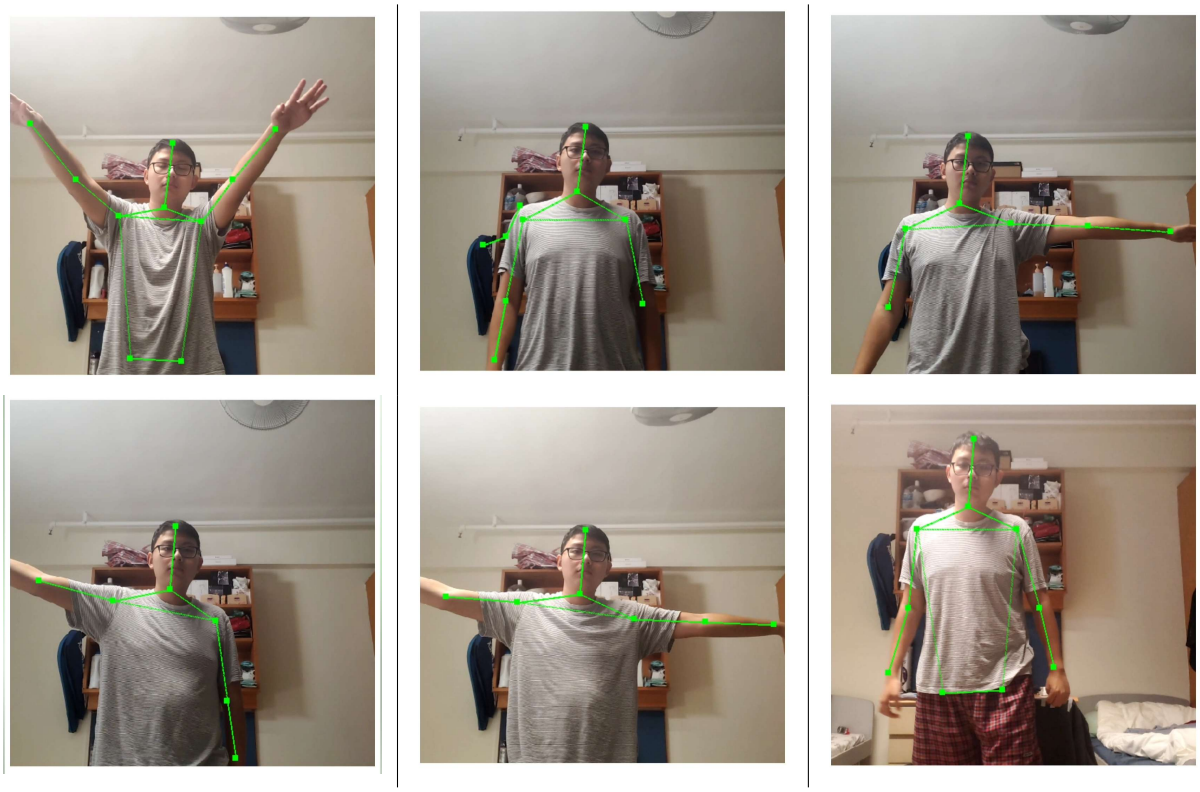
CPU 1: AMD Ryzen 5 Pro 3500U with Radeon Vega 8

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

4.2.2 LitePose

LitePose [72] is the most efficient architecture when it comees to pose estimation. Originally inspired by HRNet, this pose estimation architecture is able to remove redundant parts and improving the overall architecture, through Fusion Deconv and Large Kernel Convs, enabling it to perform in low computing devices. This also allows the architecture to perform with much lower latency compared to its predecessor while not sacrificing its accuracy.

Table 4.3: Litepose inference run on mobile



From *Table 4.3*, we can see that Litepose manages to estimate the poses correctly. However, with Litepose, there are moments where the model inaccurately estimate the poses, especially in regards to detecting background objects as part of a pose, shown in *Table 4.3*.

4.2.3 Movenet

MoveNet, released in 2022, is another architecture that is suitable for our project. The model offers great performance for real-time video, easy to use (offered as part of TensorFlow Hub), and it has different models for different goals. MoveNet uses a combination of Convolution layer, Max Pooling, Dense layer, and Up sampling layer.With Movenet, we manage to do an inference run on local device, and it manages to locate the keypoints correctly, as shown in *Table 4.4*. Compared to Litepose, it doesn't perform as smooth; however, pose estimation accuracy is better in Movenet compared to Litepose.

Table 4.4: Movenet inference run on CPU

4.2.4 BlazePose

Built for low end devices, BlazePose is another architecture fit for our project. Not only that it doesn't require a high computational device to operate it, it also has some perks, which are customizable through python, supports more keypoints and many others. However, we also find that there are times where this tool fall short, especially when handling pollution, where it is not able to determine correctly the keypoints on the body. This happens because of its nature of approach on bottom-up.

Table 4.5: BlazePose inference run on CPU

In conclusion, we found that LitePose, although have the best performance from *Table 4.2*, it suffers from slightly in accuracy, mentioning that the model is designed to be simpler than the original HRNet in favour of performance.

However, the biggest problem that LitePose have is the lack of support for non NVidia GPU machine, which makes it not suitable for our project. Meanwhile, BlazePose and Movenet perform very similar between each other; but, BlazePose manage to do this with more keypoints plotted (33 in BlazePose vs 17 in Movenet). Because of that, for our project, we are leaning towards using BlazePose as our pose estimation model, although we still keep Movenet as a second option.

4.2.5 LSTM Model

Succeeding in finding different pose estimation tools that are available to use, we began to find more on how to process those poses in a sequence where the poses detected connect together to form an action. Different researches points out that we are able to use and integrate Long Short-term Memory to our existing pose estimation. This is possible since the pose estimation system that we use enables us to only extract the keypoints from different body parts that are detected in the frame. In addition to that, for each and every single one of keypoints that are used in BlazePose are in form of arrays, this enables us to do concatenation for unification through numpy and thus enables us to have the structure to do training.

Data Collection & Preparation

No model is able to be run in anything without the data. That is the realization that we had as we thought of this solution. Now that the keypoints are collected in form of numpy array, it is easier for us to collect sequences of image. Initially there are several datasets that are available and well known for action recognition, namely UCF101[59], Kinetics[35], Moments [44] and many more. However, we decided not to use those datasets for several reasons. First,

those dataset takes very long time to train, as the length / sequence that each dataset have are varying from 1 sec to 15 minutes. With the amount of data in one dataset, we believe that it will take some time to train all of them. Second, we envision that the actions that are to be detected in the model is simple gestures used in our daily lives, such as raising hand, swiping and rotating. As we look at the datasets, most of them contains obscure actions that we don't need and would be weird instead if implemented, such as breakdancing, high kick, tai chi.

Because of this reasons, we decided to take our own data. This could be done by recording 30 videos of 30 frames for each movements. This way, we would have sufficient data to train the activity for it to be recognizable. In a single frame, it contains the coordinates of keypoints extracted from that frame, in total 258 coordinates (x, y, z) for keypoints on pose, right hand and left hand captured through BlazePose. This frame by frame recording will imitate as if there are a certain movement done through the keypoints.

There is not much to be done for the data preparation before the model training, apart from splitting the data collected to be training and testing data.

Model Designing

For us to have the machine learning model that has the capability to use every frame, it means that we need to use an algorithm that allows the training to go through the whole sequence that are available in the dataset we collected. Hence, we decided to use LSTM as the machine learning algorithm. Not only to just use one layer in the infrastructure, but also to use two layers to imitate multi-step forecasting for the movements. The rest of the architecture is explained through the *Figure 4.3*. Also, the architecture explained through Tensorboard would also help explain on the more detail on the graph, which

is shown in figure 4.4. For this model, Adam is utilized in this model as the optimization technique since it can give one that can manage difficulties with sparse gradients and noise.

```
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 4.3: LSTM Model

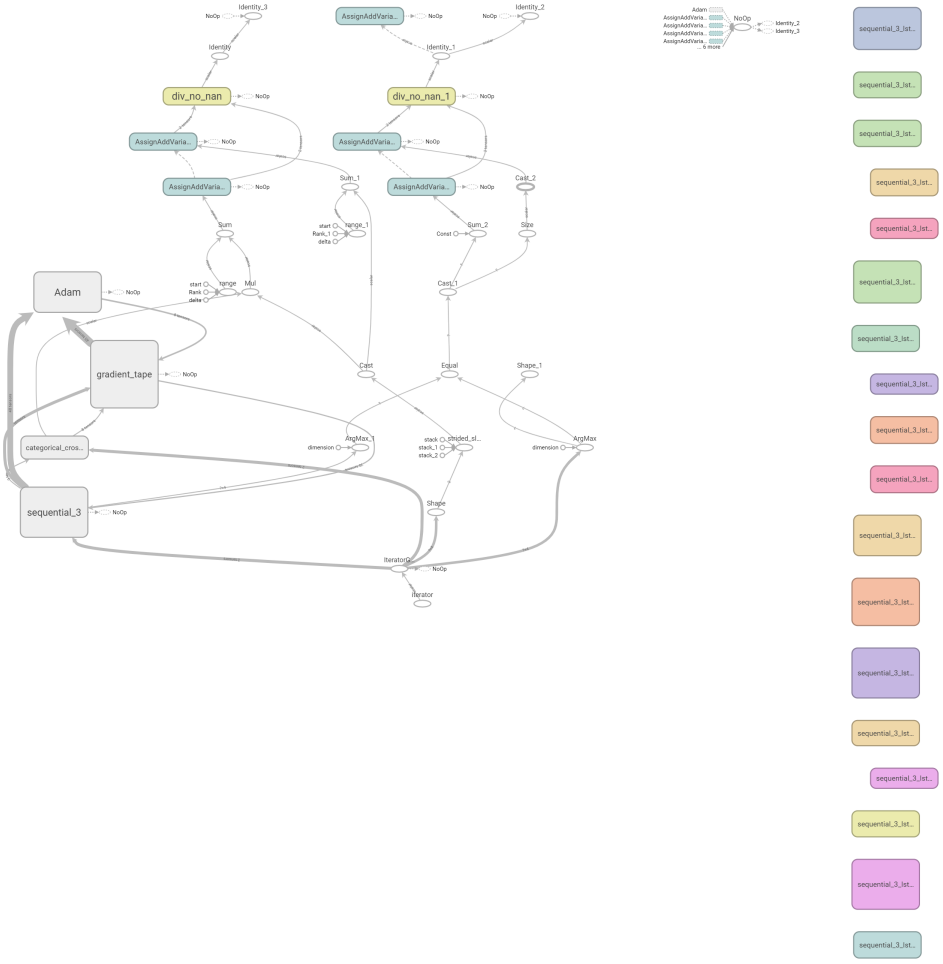


Figure 4.4: Architecture of the LSTM model

4.3 Overall evaluation

4.3.1 Accuracy evaluation

On the training for the model, we thought that we need a lot of epochs for the model to learn., so we set it to around 1000 epochs. However, the training categorical accuracy reaches 99% accuracy pretty quickly. We decided to finish the training early in 60 epochs, so the model won't overfit. This decision is

not just a hunch, as it is revealed in more iterations of model building that overfitting prone to happen quite quickly, and thus it is wise to use enough epoch for it to train. *Figure 4.5* and *Figure 4.6* describe the graph produced on the accuracy and loss on the training.

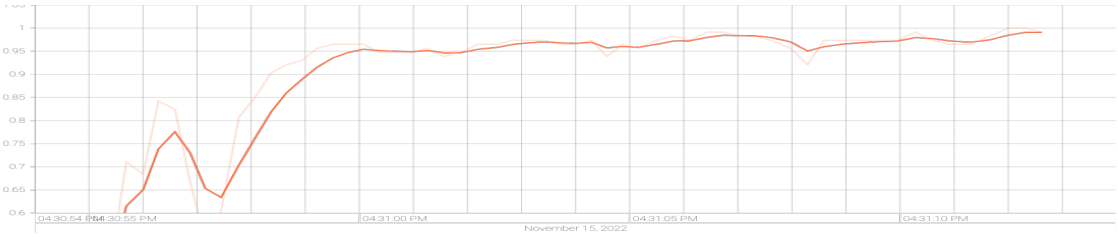


Figure 4.5: Training statistics on accuracy

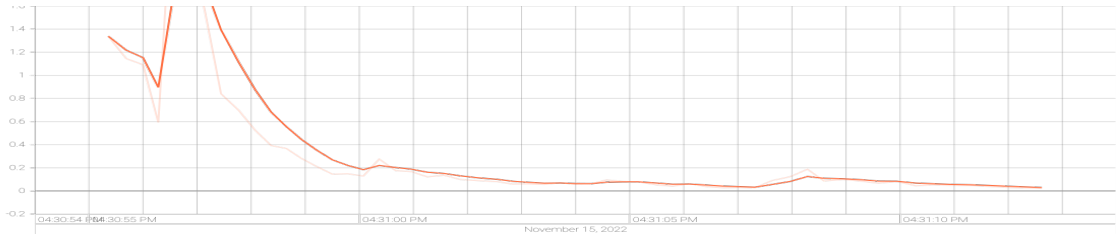


Figure 4.6: Training statistics on loss

As seen on the figures above, we can see that there are a significant drop in the accuracy in around 20 epochs. This drop is however, without any reason whatsoever since from there it will increase with stability and hitting constant accuracy above 95 percent.

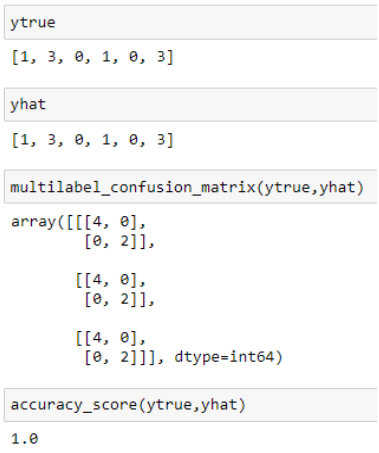


Figure 4.7: Confusion Matrix on the testing data

After the model is built, the thing that is needed to test is the accuracy. The way to test this accuracy is through a method called confusion matrix, to test whether the result of the model and expected value are the same. As we can see on *Figure 4.7*, our model is able to hit accuracy score of 1.0. This prove that the model that was built had a good performance in detecting the gesutres correctly.

4.3.2 Speed evaluation

Now that the part of stage 2 is ready, then the speed or the performance of the integration is able to be tested. Initially, it was pretty hard to determine accurately on the speed, since the speed varies a lot on whether a certain part is detected or not (it varies if only keypoints are detected, or when a motion that form the gesture trained in the model is detected). Most of the time a motion is registered in the camera, the system will print out the latest gesture that the user made. So in *Figure 4.7*, it can be seen that the latest motion that is made by the user is Swipe Right (SwipeR), and the FPS shown on user's CPU (AMD Ryzen 5 PRO 3500U) is showing 4 FPS. Despite its low number,

the interaction with camera shows that it is smoother than 4 FPS that was achieved in comparison done in *Table 4.2*. Trial on another CPU (Intel Core i5 7300HQ) shows that the performance of the model improves a lot, reaching 12 FPS.

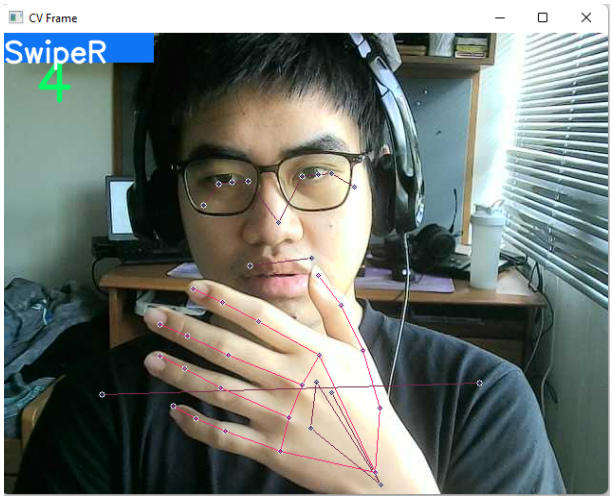


Figure 4.8: Stage 2 evaluation on speed (FPS is shown on top left corner)

Chapter 5

Conclusion

5.1 Summary

In this semester, we have done research on object detection, face recognition, pose estimation, and action recognition. We have chosen several models for those components and do inference run on these models to determine the model best fitted to our purpose. Furthermore, for action recognition, we have built an LSTM-based action recognition with 4 poses as the trial poses. Our LSTM-based action recognition manages to achieve accuracy of 95 percent and average of 8 FPS, when combined with BlazePose. This indicates how efficient our model is despite the limitations of our devices.

5.2 Future Work

In the next term, there are 2 main goals that we would like to achieve for this project:

1. We would like to combine both stages of the model into one code and optimize the models such that we can obtain a good performance and accuracy result

2. Add more poses to be recognized and if possible, the ability for users to add their own poses and faces

Bibliography

[1] A. Amini, A. S. Periyasamy, and S. Behnke. T6d-direct: Transformers for multi-object 6d pose direct regression. *CoRR*, abs/2109.10948, 2021.

[2] A. Amini, A. S. Periyasamy, and S. Behnke. Yolopose: Transformer-based multi-object 6d pose estimation using keypoint regression, 2022.

[3] R. Bajpai and D. Joshi. Movenet: A deep neural network for joint profile prediction across variable walking speeds and slopes. *IEEE Transactions on Instrumentation and Measurement*, 70:1–11, 2021.

[4] T. Baltrusaitis, P. Robinson, and L.-P. Morency. Constrained local neural fields for robust facial landmark detection in the wild. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 354–361, 2013.

[5] T. Baltrušaitis, P. Robinson, and L.-P. Morency. Openface: an open source facial behavior analysis toolkit. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–10. IEEE, 2016.

[6] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann. Blazepose: On-device real-time body pose tracking, 2020.

[7] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grund-

mann. Blazeface: Sub-millisecond neural face detection on mobile gpus, 2019.

[8] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[9] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[10] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *IEEE Transactions on pattern analysis and machine intelligence*, 23(3):257–267, 2001.

[11] A. Bowes, A. Dawson, and D. Bell. Ethical implications of lifestyle monitoring data in ageing research. *Information, Communication & Society*, 15(1):5–22, 2012.

[12] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields, 2018.

[13] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017.

[14] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[15] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2017.

[16] Y. Chen, Z. Zhang, C. Yuan, B. Li, Y. Deng, and W. Hu. Channel-wise topology refinement graph convolution for skeleton-based action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13359–13368, 2021.

[17] B. Cheng, B. Xiao, J. Wang, H. Shi, T. S. Huang, and L. Zhang. Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation, 2019.

[18] M. Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

[19] H. Duan, J. Wang, K. Chen, and D. Lin. Pyskl: Towards good practices for skeleton action recognition. *arXiv preprint arXiv:2205.09443*, 2022.

[20] H.-S. Fang, J. Li, H. Tang, C. Xu, H. Zhu, Y. Xiu, Y.-L. Li, and C. Lu. Alphapose: Whole-body regional multi-person pose estimation and tracking in real-time, 2022.

[21] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu. Rmpe: Regional multi-person pose estimation, 2016.

[22] Z. Geng, K. Sun, B. Xiao, Z. Zhang, and J. Wang. Bottom-up human pose estimation via disentangled keypoint regression, 2021.

[23] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[25] T. Grel. Region of interest pooling explained. `https://deepsense.ai/region-of-interest-pooling-explained/`, 2017.

[26] R. A. Güler, N. Neverova, and I. Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7297–7306, 2018.

[27] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn, 2017.

[28] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Computer Vision – ECCV 2014*, pages 346–361. Springer International Publishing, 2014.

[29] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[30] S. Herath, M. Harandi, and F. Porikli. Going deeper into action recognition: A survey. *Image and vision computing*, 60:4–21, 2017.

[31] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.

[32] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural

networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[33] P. Jin, V. Rathod, and X. Zhu. Pooling pyramid network for object detection. *arXiv preprint arXiv:1807.03284*, 2018.

[34] K. Kang, H. Li, J. Yan, X. Zeng, B. Yang, T. Xiao, C. Zhang, Z. Wang, R. Wang, X. Wang, and W. Ouyang. T-CNN: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2896–2907, oct 2018.

[35] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[36] D. E. King. Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, 10:1755–1758, 2009.

[37] O. Köpüklü, A. Gunduz, N. Kose, and G. Rigoll. Real-time hand gesture detection and classification using convolutional neural networks. In *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, pages 1–8. IEEE, 2019.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[39] J. Lee, M. Lee, D. Lee, and S. Lee. Hierarchically decomposed graph convolutional networks for skeleton-based action recognition. *arXiv preprint arXiv:2208.10741*, 2022.

[40] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. SSD: Single shot MultiBox detector. In *Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing, 2016.

[41] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: a skinned multi-person linear model. *ACM Trans. Graph.*, 34:248:1–248:16, 2015.

[42] D. Maji, S. Nagori, M. Mathew, and D. Poddar. Yolo-pose: Enhancing yolo for multi person pose estimation using object keypoint similarity loss, 2022.

[43] D. Marikyan, S. Papagiannidis, and E. Alamanos. A systematic review of the smart home literature: A user perspective. *Technological Forecasting and Social Change*, 138:139–154, 2019.

[44] M. Monfort, A. Andonian, B. Zhou, K. Ramakrishnan, S. A. Bargal, T. Yan, L. Brown, Q. Fan, D. Gutfruend, C. Vondrick, et al. Moments in time dataset: one million videos for event understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–8, 2019.

[45] S. Mroz, N. Baddour, C. McGuirk, P. Juneau, A. Tu, K. Cheung, and E. Lemaire. Comparing the quality of human pose estimation with blazepose or openpose. In *2021 4th International Conference on Bio-Engineering for Smart Technologies (BioSMART)*, pages 1–4, 2021.

[46] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation, 2016.

[47] OpenAI. Openai five. `https://blog.openai.com/openai-five/`, 2018.

[48] D. Osokin. Real-time 2d multi-person pose estimation on cpu: Lightweight openpose. *arXiv preprint arXiv:1811.12004*, 2018.

[49] D. Qi, W. Tan, Q. Yao, and J. Liu. Yolo5face: why reinventing a face detector. *arXiv preprint arXiv:2105.12931*, 2021.

[50] R. Rakhimov, E. Bogomolov, A. Notchenko, F. Mao, A. Artemov, D. Zorin, and E. Burnaev. Making densepose fast and light, 2020.

[51] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[52] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015.

[53] S. Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[54] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[55] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[56] S. Sepasgozar, R. Karimi, L. Farahzadi, F. Moezzi, S. Shirowzhan, S. M. Ebrahimzadeh, F. Hui, and L. Aye. A systematic content review

of artificial intelligence and the internet of things applications in smart home. *Applied Sciences*, 10(9), 2020.

[57] S. I. Serengil and A. Ozpinar. Lightface: A hybrid deep face recognition framework. In *2020 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pages 23–27. IEEE, 2020.

[58] M. Shafiq, Z. Gu, O. Cheikhrouhou, W. Alhakami, and H. Hamam. The rise of "internet of things" review and open research issues related to detection and prevention of iot-based security attacks. *Wireless Communications and Mobile Computing*, 2022:12, 08 2022.

[59] K. Soomro, A. R. Zamir, and M. Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[60] L. Stoffl, M. Vidal, and A. Mathis. End-to-end trainable multi-instance pose estimation with transformers. *arXiv preprint arXiv:2103.12115*, 2021.

[61] G. Sung, K. Sokal, E. Uboweja, V. Bazarevsky, J. Baccash, E. G. Bazavan, C.-L. Chang, and M. Grundmann. On-device real-time hand gesture recognition. *arXiv preprint arXiv:2111.00038*, 2021.

[62] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1701–1708, 2014.

[63] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of*

the IEEE international conference on computer vision, pages 4489–4497, 2015.

[64] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of cognitive neuroscience*, 3(1):71–86, 1991.

[65] J. Uijlings, K. Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 09 2013.

[66] S. Vandenhende, S. Georgoulis, W. Van Gansbeke, M. Proesmans, D. Dai, and L. Van Gool. Multi-task learning for dense prediction tasks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2021.

[67] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[68] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, T. Ewalds, D. Horgan, M. Kroiss, I. Danihelka, J. Agapiou, J. Oh, V. Dalibard, D. Choi, L. Sifre, Y. Sulsky, S. Vezhnevets, J. Molloy, T. Cai, D. Budden, T. Paine, C. Gulcehre, Z. Wang, T. Pfaff, T. Pohlen, D. Yogatama, J. Cohen, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, C. Apps, K. Kavukcuoglu, D. Hassabis, and D. Silver. AlphaStar: Mastering the Real-Time Strategy Game StarCraft II. `https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/`, 2019.

[69] J. Walsh, N. O' Mahony, S. Campbell, A. Carvalho, L. Krpalkova,

G. Velasco-Hernandez, S. Harapanahalli, and D. Riordan. Deep learning vs. traditional computer vision. 04 2019.

[70] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang, W. Liu, and B. Xiao. Deep high-resolution representation learning for visual recognition, 2019.

[71] M. Wang and W. Deng. Deep face recognition: A survey. *Neurocomputing*, 429:215–244, 2021.

[72] Y. Wang, M. Li, H. Cai, W.-M. Chen, and S. Han. Lite pose: Efficient architecture design for 2d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13126–13136, 2022.

[73] B. Xiao, H. Wu, and Y. Wei. Simple baselines for human pose estimation and tracking, 2018.

[74] L. Yang, L. Lou, X. Song, J. Chen, and X. Zhou. An improved object detection of image based on multi-task learning. In *2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications (CVIDL & ICCEA)*, pages 453–457, 2022.

[75] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu, and N. Sang. Bisenet: Bilateral segmentation network for real-time semantic segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, pages 325–341, 2018.

[76] C. Z. Yue and S. Ping. Voice activated smart home design and imple-

mentation. In *2017 2nd International Conference on Frontiers of Sensors Technologies (ICFST)*, pages 489–492, 2017.

[77] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann. Mediapipe hands: On-device real-time hand tracking. *arXiv preprint arXiv:2006.10214*, 2020.

[78] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890, 2017.