University of Hong Kong

Final Year Project

Final Report

Deep Learning Hand Poses

Supervisor: Dr. Dirk Schnieders

Group Members: Chen Zihao

Da Yujia

Zhang Yuqian

Author: Da Yujia

April 14, 2019
Abstract

Hand poses recognition has developed rapidly in recent years. It can be widely applied in many fields, such as robotics, artificial intelligence, and virtual reality. Previous works on the related datasets and models show the outstanding performance on hand detection and hand segmentation. This project proposes an approach that separates the process of hand pose recognition into three stages, which are hand detection, joint recognition and pose estimation. Stage 1, hand detection, aims at detecting the position of hands in images, and it can be extended into hand segmentation. Datasets for hand detection are preprocessed, and the models are trained and evaluated, giving a satisfactory precision with high efficiency. The model for segmentation was implemented and tested as well. However, the segmentation accuracy did not appear consistent on all the evaluation hand poses. Further enhancement is required to get a better trade-off between performance and time consumption.
Acknowledgements

I would like to first thank our supervisor Dr. Schnieders for his advice on the framework of our project and possible fields that we may want to focus on. Special thanks go to Mable, who provided us with help and guidance in completing this report. I would also like to show our gratitude to Vera Kang for her kind help in data preparation. Finally, I would like to express my appreciation to my groupmates, Zhang Yuqian and Chen Zihao, for their significant contribution to this project.
# Table of Contents

Abstract

Acknowledgements

List of figures

List of tables

1 Introduction
   1.1 Background ................................................. 1
   1.2 Purpose of the study ....................................... 1
   1.3 Contribution of the project ................................. 3
   1.4 Outline of report ........................................... 4

2 Literature review
   2.1 Theoretical background ..................................... 5
   2.2 Review of existing models .................................. 6
   2.3 Review of datasets .......................................... 8

3 Methodology
   3.1 Environment setting ......................................... 10
   3.2 Data preparation ............................................ 10
   3.3 Model construction .......................................... 12
   3.4 Evaluation metrics .......................................... 15

4 Experiments and Results
   4.1 Hand detection .............................................. 17
      4.1.1 Data preprocessing ..................................... 17
      4.1.2 Model Implementation and Evaluation .................... 19
   4.2 Hand segmentation .......................................... 23
      4.2.1 Data preprocessing ..................................... 24
4.2.2 Model implementation and evaluation

5 Timeline

6 Future works

7 Conclusion
# List of Figures

1. Gesture representation of the alphabet ........................................... 2
2. Selected hand poses ................................................................. 3
   2.a 5 .............................................................................. 3
   2.b A .............................................................................. 3
   2.c F .............................................................................. 3
   2.d I .............................................................................. 3
   2.e L .............................................................................. 3
   2.f S .............................................................................. 3
   2.g Y .............................................................................. 3
3. Difference between hand detection(a) and segmentation(b) ............. 5
   3.a Hand detection .................................................................. 5
   3.b Hand segmentation .......................................................... 5
4. Sample image from Rendered Handpose Dataset ............................... 8
5. Difference between hand detection(a) and segmentation(b) ............. 9
6. Original image .......................................................................... 11
7. Illumination of data augmentation approaches .................................. 12
   7.a Lossy image compression ................................................... 12
   7.b Rotation ........................................................................ 12
   7.c Cropping ........................................................................ 12
   7.d Mirroring ........................................................................ 12
   7.e Global illumination ............................................................ 12
12. Samples from Egohands Dataset and Rendered Handpose Dataset .... 17
   12.a Egohands ..................................................................... 17
   12.b Rendered ..................................................................... 17
List of Tables

1. The responsibility of each stage, as well as input and output flows ............. 3
2. Milestones and their descriptions ......................................................... 4
3. Implemented networks .......................................................................... 20
4. Datasets and Losses for different training steps ........................................ 22
5. Data augmentation methods on hand segmentation ................................. 25
6. Average IoU scores of seven poses ...................................................... 27
7. Schedule .............................................................................................. 28
1 Introduction

1.1 Background

Hand poses recognition is a process to detect the position of hands, recognize joints and estimate gestures. Recent years have witnessed a significant breakthrough in this area because of the rapid development of learning-based approaches, especially deep learning neural networks, and computer vision techniques. The rise of 3D sensing technology resulted in large datasets with annotated hand poses, along with sophisticated network structures that can cope with demanding recognizing tasks. As one of the critical technologies to achieve Human-Machine Interaction (HMI), hand poses recognition has been applied in many fields, including robotics, artificial intelligence, virtual reality (VR) and augmented reality (AR), etc. For example, the concept of VR games, where a player can experience being in a three-dimensional game space and interact with that environment, is attracting more attention in the past few years. However, it requires VR equipment, such as special gloves with sensors, to capture the gesture of your hands and then reflect your action to the system. As a result, it is inconvenient for daily use and unable to popularize because of the high prices of equipment. Hand pose recognition perfectly solves the problem since the only required hardware is an ordinary camera, which can be found everywhere on smart devices, while the hand pose can be recognized and input to the real-time control system. With technological development, the hand pose recognition will be made use of in more and more industries.

1.2 Purpose of the Study

This project aims to achieve highly accurate hand detection, joints recognition and pose estimation with deep learning approaches, focusing on several specific hand gestures, and constructing three deep learning models corresponding to the above three functions. The model’s capability is limited to detect only full hands shown in the image. More specifically, occluded joints are permitted from the view of the camera, but the actual location of each joint should not be out
of the image; detect only hands without disability that, each hand in the image has 21 joints; the hands in the image should be at least clear enough for human beings to understand. With these restrictions, given an input visual resource with hands included, our model will return the location of each joint and categorize each hand pose into a predefined type.

Since there exist a large number of hand poses with various shapes and structures, for example, the OK pose and the Victory pose looks somewhat different, and it is impossible to train the models on all kinds of gestures, several particular hand poses are selected to be trained and evaluated in the models. Figure 1 shows the pose representations of the alphabet, and we propose to use six of them, which are illustrated in Figure 2a-f, including the representation of letter 'A', 'F', 'I', 'L', 'S', and 'Y'. Besides, a frequently-used pose '5' (see Figure 2a below) is also considered. In total, we took seven poses into account.

![Figure 1: Gesture representation of the alphabet](image)

To reduce the project complexity, we define three stages to divide a massive task into small subtasks. Each stage can be regarded as an independent block with a model, which means that they can be trained independently and simultaneously with prepared training data and label, and hence this approach reduces the workload and enhances project efficiency. Input and output of each stage are shown in Table 1.

Since there are several types of visual resources that we plan to train and test on, including images, videos, and live demo, we determine to set three milestones for the project step by step. Remarkably, a video with 20 FPS (Frame Per Second), can be considered as 20 sequential
Figure 2: Selected hand poses

Table 1: The responsibility of each stage, as well as input and output flows

<table>
<thead>
<tr>
<th>Stage</th>
<th>task</th>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1 Hand Detection</td>
<td>locate all human hands in the images and crop the hands out</td>
<td>Full size images containing hands</td>
<td>Cropped hand-centered images</td>
</tr>
<tr>
<td>S2 Joint Recognitiation</td>
<td>Mark the 21 joints in each hand image</td>
<td>Output of Stage 1</td>
<td>2D coordinates of hand joints</td>
</tr>
<tr>
<td>S3 Pose Estimation</td>
<td>Associate the input data with a predefined hand pose</td>
<td>Output of Stage 1 and Stage 2</td>
<td>Hand Pose Categories</td>
</tr>
</tbody>
</table>

images per second; live demo, meaning a video in real time, can as well be decomposed to an image sequence. Considering this logical dependency and level of intricacy in ascending order, the milestones are defined in Table 2. M2 and M3 are based on M1, and they share almost identical algorithm as M1. We regard M1 as the trunk, while M2 and M3 are like the branches extended from M1.

### 1.3 Contribution of the project

There are three group members in our final year project group, each responsible for onestage of the whole project. Da Yujia, Zhang Yuqian and Chen Zihao focus on work related to Stages 1, 2 and 3 respectively. The initial research on existing datasets, evaluation metrics and framework etc., the creation of an original high-quality dataset, and the ultimate combination of the three stages are done by all three members of the group.
### Table 2: Milestones and their descriptions

<table>
<thead>
<tr>
<th>Milestone</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M1</td>
<td>Accurately identify a static hand pose and its joints in a single RGB Image</td>
</tr>
<tr>
<td>M2</td>
<td>Reduce the oscillation in joint labelling on the sequential images and identify poses on each static image with accordant accuracy from M1</td>
</tr>
<tr>
<td>M3</td>
<td>Improve the prediction speed so that it can support live demo with equivalent accuracy from M2</td>
</tr>
</tbody>
</table>

In all, the work distribution is approximately balanced between each group member.

### 1.4 Outline of Report

This report contains six major parts. Section One introduces the topic Deep Learning Hand Poses and illustrates project objects and the pipeline. The contribution is presented in this section as well.

Section two reviews the literature in this area of study. State-of-the-art models and high-quality datasets for stage 1 of the project are discussed in this part.

Section Three presented the methodology of the first stage used in environment setting, data augmentation, model construction, and evaluation metrics.

Section Four shows all the experiments and results of this project. Work in stage 1 is separated into two parts: hand detection and hand segmentation. Data preprocessing completed, model implementation and evaluation are completed for both parts. This section also illustrates the progress of connecting models from all the three stages together. The difficulties encountered during the process are explained afterward, followed by suggested solutions.

Section Five demonstrates what can be improved in the future for stage 1. A timeline starting from September to the end of the project is provided as well.

Section Six summarizes the findings and conclusion of the project.
2 Literature Review

2.1 Theoretical background

The task in stage 1 can be categorized into two main processes: hand detection and hand segmentation. Hand detection locates the hands and output boundary boxes to show their position, while hand segmentation gives hand masks. Figure 5 illustrates the difference between hand detection and segmentation. The green square in Figure 3a is the boundary box and the red polygons in Figure 3b is the hand mask of the sample image.

(a) Hand detection

(b) Hand segmentation

Figure 3: Difference between hand detection(a) and segmentation(b)

In the past few years, many papers on these two processes have been published as a result of the development of deep learning neural networks and computer vision technologies. At the beginning of the exploration on hand detection and hand segmentation decades ago, before the rise of learning based approaches, computer vision techniques were widely accepted and applied in use. These methods mainly took advantages of edge detection and other detection routines, for example, recognizing hands by the color of skins. Edge detection aims at finding the edges of an object and locate its position according to the detected edges. More specifically, the images should be processed by several filters in advance and then edge points, which are usually located in a sharp change in image brightness, are identified by a large number of mathematical functions. It can be observed that edge detection is extremely complicated, and the accuracy of the detection result is not satisfactory because of the complexity of images.
By contrast, deep learning approach does not have any human-made rules to identify points but uses images with labels to train models and detects objects by trained models. Not only it reduces the human work on processing images and dealing with mathematical methods, but also significantly improve the precision of detection. Meanwhile, the speed does not decrease, or even faster than that of computer vision methods because of the reduction of the image processing procedure. However, the traditional CV is still in use to do data preprocessing. Nowadays, most studies on hand detection make use of transfer learning to enhance the accuracy of predefined object detectors, for example, models based on Convolutional Neural networks (CNN) such as MobileNet and ResNet50. Transfer learning means that a model can make use of another model as its starting point. CNN is an algorithm of deep learning, and have reported remarkable performance of static recognition. Therefore, it is the most popular and state-of-art network used in the field of object detection.

Similar to hand detection, the implementation of hand segmentation is based on CNN models and applies transfer learning. It is intuitive that hand segmentation is more complicated than detection and as a result, the speed and accuracy decrease. However, the masks of hands, rather than boundary boxes, will improve the performance of models in stage 2 and 3.

Moreover, there exists a large number of studies for real-time applications on the recognition of dynamic hand poses based on the progress made on static estimation. These studies focus on improving the speed of models while keeping an acceptable precision. With the development of researches on lightweight models, it is possible to develop applications that can detect hand poses in real time.

### 2.2 Review of existing models

In the aspect of stage 1, R-FCN[8], Mask-RCNN[10] and RetinaNet[11] worth researching, as they rank relatively high on the leaderboard of COCO 2017 Object Detection Task[12]. COCO
Object Detection Task is an annual object detection competition, where all the models with different structures are trained by the same COCO Dataset[12], which is a labeled dataset with 80 classes of daily objects such as animals, vehicles, and commodities. After training the performance of models will be evaluated in two dimensions: speed and accuracy, and then the models will be ranked according to their time-accuracy efficiency. Although there is no hand class in COCO Dataset, the winners in this competition show their well-designed structures for our hand detection. In these three models, R-FCN and RetinaNet are the models for hand detection with boundary boxes as the output, while Mask-RCNN is designed for hand segmentation. Especially for ResNet, which represents deep residual network, it was firstly presented in 2015 and won the first place in both MS COCO and ImageNet in the same year. Its performance on the detection, classification, localization, and segmentation beat the model in the second place more than 10%.

In the same year, a class of models called MobileNets with efficient time and accuracy tradeoff due to its light weight was presented [2]. The new model makes it possible to develop mobile vision applications and achieve real-time hand detection. As shown in the tensorflow model zoo[6], which is a library containing pretrained object detection models, MobileNets combined with SSD algorithm has the best time-accuracy efficiency because of its extremely fast speed. SSD refers to Single Shot MultiBox Detector, a model that predicts bounding box by regression and object class by classification directly[14]. Compared with Fast-RCNN[7], which contains two networks for region proposal and object classification, it would be highly advantageous to apply SSD and MobileNet in live detection.

Apart from models, feature extractors play a significant role in the performance of networks. Feature Pyramid Network (FPN) shows significant improvement as a generic feature extractor in several applications[13]. In addition, innovation in selecting specific training data will also be considered in this project, including focal loss[11] and Online Hard Example Mining[1]. A series of models and ideas will be applied to compare and evaluate before we determine a robust model suitable for our hand case.
2.3 Review of datasets

There are many public hand pose datasets with ground-truth annotations for the first stages, such as Rendered Handpose Dataset[5], NYU Hand Pose Dataset[4] and EgoHands Dataset[3].

Rendered Handpose Dataset[5] contains 41258 training and 2728 testing samples, and each sample has one RGB image, one depth image, a segmentation mask and the coordinates of 21 joints. It provides sufficient training and evaluation data with all the formats of labels and various hand poses. However, images in this dataset are not real photos and are generated by software (see Figure 4). Because of this small difference, it is doubtful whether the model trained by this dataset can work correctly on real pictures.

![Sample image from Rendered Handpose Dataset](image)

Figure 4: Sample image from Rendered Handpose Dataset

Egohands Dataset[3] consists of 48 short videos with hand masks, which can be extended to 4800 labeled images. Those videos record scenes that two players are playing cards or chess. As a result, images in this dataset contain hands of both first-person view and the third view. The first view hand is marked as purple and yellow while the third view ones are red and green in Figure ???. However, there exist limitations of this dataset since the size is not large enough, and hand poses in some images are similar to each other.

For NYU Dataset[4], there are not the only frontal view of each hand pose but also 2 side views and depth information, which promise a higher data variety. This dataset is significant in the number of images and is widely used and tested in the field of hand pose recognition. Since many state-of-the-art works set baselines according to these datasets, using the same datasets
Figure 5: Difference between hand detection(a) and segmentation(b) provides a convenient platform for horizontal comparison and evaluation.
3 Methodology

3.1 Environment setting

Tensorflow, one of the opensource deep learning frameworks, is adopted in this project mainly for three reasons. To begin with, it provides a large number of APIs and library functions for us to call directly, which remarkably improve the efficiency of the project. Especially for stage 1, tensorflow object detection API contains not only scripts for processing data, training, testing and visualizing, but many pretrained models and their configuration files as well, significantly saving our time and resources. Secondly, compared with other opensource frameworks such as PyTorch and Keras, tensorflow is the most popular framework among all of its competitors, thus there are sufficient resources and tutorials for beginners like us to start with. TensorFlow is such a versatile framework that many models which are initially implemented in other frameworks like Caffe and Caffe2 are re-implemented with TensorFlow by developers. Because of its comprehensive functions, it is convenient to compare the performance of existing models in this platform and in other frameworks. Thirdly, tensorflow support parallel computing and has many related functions, which is friendly to our time-consuming project. For all the above reasons, tensorflow framework becomes our prudent choice and version 1.13.0 will be installed.

Apart from frameworks, we have some other hardware settings. Since there is a huge amount of computation in this project, Graphics Processing Units (GPU) are required, otherwise, with only Central Processing Units (CPU), all the resources and the time will be occupied by the training process. Google Cloud Platform provides us with NVIDIA® Tesla® K8 GPUs, and corresponding NVIDIA Driver and CUDA 10.0 will be installed on the cloud.

3.2 Data preparation

For the first stage, the project will take two existing public datasets as the training and testing data, which are Rendered Handpose Dataset and EgoHands. This is because there are hand masks in both two datasets and as a result, images can be applied to hand detection and segmen-
tation after processing. Besides, since the images in Rendered Handpose Dataset are not photos taken by the cameras in reality, but the generated images produced by computer software, the predicted results on real pictures are likely to be unsatisfactory if the model were only trained on this dataset. Therefore, Egohands Dataset will serve as a supportive dataset that gives more realistic details to the model.

Moreover, with regard to the procedure of hand detection, the coordinates of bounding boxes, the final annotations of images that feed into networks, should be generated from existing labels in the datasets. However, the formats of labels in Rendered Handpose Dataset and Egohands Dataset are PNG masks and bounding points respectively. In this situation, two different methods will be applied to create labels of hand detection: from joints and from mask. For Rendered Handpose Dataset, although there are masks in the dataset, joints annotations will be taken advantage of because it costs too many computational resources to find the edge of hands by pixel-wise comparison. The details of generating will be discussed in Section 4.1.1.

Meanwhile, data augmentation will be applied to all datasets to increase the number of images and enhance the comprehensiveness of training data. More specifically, the variety of datasets will be improved by data augmentation, and as a result, the detection of models will remain precise in unexpected situations such as images containing hands with different sizes of hands, views and illumination.

![Original image](image)

Figure 6: Original image

As shown in Figure 6, there are various approaches to augmenting images, such as lossy image compression, random cropping and rotation, global illumination and mirroring, etc. Lossy image compression means to reduce resolutions of images, in order to simulate the blur frames extracted from a video (Figure 7a). Thus, it attaches great importance to apply lossy image
compression in real-time detection, especially for quickly-moving objects. Random crop and rotation aim at training models with hands of different angles and sizes (see Figure 7b and 7c). Moreover, models are capable of detecting hands accurately under any image brightness and color temperature if changing the global illumination of training data, while mirroring will solve the problem that the model can only recognize one side hands (left hands or right hands). It is notable that there is no need to apply every data augmentation method to all images in the datasets considering the limited computational resources.

### 3.3 Model construction

In stage 1, hand detection, two state-of-the-art Object Detection models, SSD combined with MobileNets and ResNet50 will be implemented and compared. Aiming at building a real-time hand detection application, we are expected to focus on the structure of SSD and MobileNets since these two models contribute to the improvement of prediction speed. SSD model is based on a feed-forward convolutional network that generates bounding boxes and classification scores of each class[14]. There exist two main features in the SSD model, multi-scale feature maps and default bounding boxes, by which the model keeps the high accuracy while promotes the speed of prediction.

Compared with ordinary convolutional networks like YOLO, where only the feature maps from the last feature layer can be passed to the final detection, feature maps in each layer with different scales are outputted to the final detector for prediction in SSD (see Figure 8). Since the objects with larger scales lie on lower feature layers, it is challenging for YOLO to localize the large hands without the feature maps from the lower layers. Therefore, with the gradual decrease in the sizes of feature layers, SSD gains advantages on the detection of objects with multiple
scales, resulting in higher accuracy than networks with the feature of single-scale feature maps.

Another remarkable feature is the approach of default bounding boxes, which means generating several bounding boxes with particular aspect ratios and the final predicted boxes are among these default boxes (see Figure 9). To be specific, we evaluate the shape offsets and confidence for the hand class on every default bounding box, and the box with the highest score on both localization and classification becomes the final prediction of the model. By contrast, YOLO needs to build another neural network for bounding boxes proposal, costing much more time and computational resources than SSD to achieve the same goal.
Combined with SSD, MobileNet is an important network to cut down the prediction time[2]. As illustrated in Figure ?, the crucial point in the MobileNet is to replace the standard convolutional filter by depthwise filters and pointwise filters but the shape of output feature maps remains unchanged. The computational cost of N standard convolutional filters with the shape of (DK, DK, M) is: $DK \cdot DK \cdot M \cdot N \cdot DF \cdot DF$, where DF indicates the size of the feature map of this filter. If the standard filters are replaced in the MobileNet, the computational cost of the depthwise and 1 x 1 pointwise convolutions is: $DK \cdot DK \cdot M \cdot DF \cdot DF + M \cdot N \cdot DF \cdot DF$, which is considered to be smaller than that of using standard convolutions. To conclude, MobileNet will be added into networks to improve the time performance of the whole model.

![Diagram](image)

For the hand segmentation part of stage 1, MASK-RCNN from the tensorflow model zoo is expected to be implemented and evaluated. All of the above models can predict the position of many different objects by their pretrained weights. Therefore, we can apply transfer learning to load their best-provided weights in the model and train with our hand datasets to make the
model specialized on hand detection. The reason for not designing a new model is that there are many effective and impressive models in the object detection field, and thus we determine not to waste resource and time on the redundant work. We will also make attempts to apply the FPN feature extractor to achieve better accuracy.

### 3.4 Evaluation metrics

To evaluate the performance of hand detection in stage 1 quantitatively, we use mean Average Precision (mAP), a metric of the accuracy of object detection. As shown in Figure 11, Average Precision (AP) refers to the area under the Precision vs. Recall diagram and fall within the range of (0, 1). The mAP is the average AP of every class, and since there is only one class, hand, in stage 1, mAP equals to AP. The precision of the model increases with the increase of mAP. Considering the real-time analysis, the efficiency of models is as important as accuracy. Therefore, we will also evaluate the inference speed of models by Frame per Second (FPS), which refers to the number of images predicted in a second.

![Precision vs. Recall Diagram](image)

**Figure 11: Precision vs. Recall Diagram[9]**

For the evaluation of hand segmentation, we use Intersection-over-Union (IoU) function to show the intersected area between the estimated hand mask and the ground-truth mask over the union area of two masks.
Except for quantitative methods, visual judgment by observation will be applied to the model, since this will give intuitive information on the accuracy and can help to filter out serious mistakes that yield small loss.
4 Experiments and Results

4.1 Hand detection

4.1.1 Data preprocessing

For Stage 1 hand detection, Egohands Dataset and Rendered Handpose Dataset were prepared for training and evaluation. As shown in Figure 12a and 12b, images from Egohands Dataset and Rendered Hand Pose Dataset are labeled as mask and coordinates of joints respectively. Since the inputs of Stage 1 should be images labeled with four vertices of detection box, we did the transformation according to two different types of labels to images in the above two datasets. The position of a single hand is fixed by the square with four points \((x_1, y_1), (x_1, y_2), (x_2, y_1)\) and \((x_2, y_2)\). With regard to the coordinate of joints, the new coordinates were generated by equation (1):

\[
\begin{align*}
    x_{\text{min}} & = \min(x_1, x_2, \ldots, x_i, \ldots, x_{21}) - p & \quad (1) \\
    y_{\text{min}} & = \min(y_1, y_2, \ldots, y_i, \ldots, y_{21}) - p & \quad (3) \\
    x_{\text{max}} & = \max(x_1, x_2, \ldots, x_i, \ldots, x_{21}) + p & \quad (2) \\
    y_{\text{max}} & = \max(y_1, y_2, \ldots, y_i, \ldots, y_{21}) + p & \quad (4)
\end{align*}
\]

\((x_i, y_i)\) represents the coordinate of i-th joints and \(p\) indicates the padding index, which was set to 15px.

Figure 14 shows the fact that padding can improve the probability that the entire hand is put inside the boundary box. Figure 14a is the original image with joint coordinates annotations.
while b illustrates two boundary boxes generated with and without paddings. It can be easily observed that the outer box describes the position of hand more precisely (14c). Therefore, a proper number of padding is of great significance in the approach of generating boxes by joints.

Another method is to generate boundary boxes by hand masks. In Egohands Dataset, hand masks are stored in a file called ‘polygons.mat’ in the form of coordinates of edge points (Figure 4.2(a)). It is convenient for us to calculate the $x_1$, $y_1$, $x_2$ and $y_2$ simply by these points and the generated boxed are the strict boundaries of hands. However, for the Rendered Handpose Dataset, their masks are in stored as PNG images, leading to pixel-wise search on the image. As a result, it is time and resources consuming to do preprocessing, which is considered to be unnecessary.

To solve this problem, we applied the method of joint coordinates to Rendered Handpose Dataset and hand masks to Egohands Dataset. The crucial information of points $(x_1, y_1)$ and $(x_2, y_2)$, together with the image name, width and height, is kept in CSV files and finally transformed to tf.records by python scripts, which is the standard form of data to be trained in the network.
After repeated attempts, we found that the above two datasets are not sufficient for training a network capable of distinguishing human faces and hands (details will be discussed in section 4.2). Therefore, the other two datasets were added to reduce false positive (FP) detection when dealing with images with faces inside. The first dataset was made by ourselves, containing 4,500 real photos with a human face and a single hand in it. We refer to this dataset as Face-Hand Dataset and Figure 15a shows a newly generated sample in Face-Hand Dataset. The hands are randomly printed on the image by scripts and thus the annotations were generated during printing. Moreover, a special dataset Caltech Human Face (Front) Dataset was used to train a hand class of the model. The structure of this dataset is the same as normal hand dataset, consisting of 450 images and an annotation file, but the labeled object is a human face (see Figure 15b). These two datasets are put into the final dataset and a script was written to separate all the images into the training set and testing set.

In total, the final training dataset consists of four datasets, Egohands Dataset, Rendered Hand-pose Dataset, Face-Hand Dataset and Caltech Human Face (Front) Dataset, providing 51,008 images for training and 2728 images for testing. All images come with annotations of four maximum and minimum coordinates of x-axis and y-axis.

4.1.2 Model Implementation and Evaluation

Model Comparison SSD combined with MobileNet and ResNet50, which are pre-trained object detection models and are capable of distinguishing different types of objects, was constructed and evaluated in the first stage. Feature Pyramids Network (FPN) was also tried to
improve the accuracy of detection.

Table 3 shows the network structures used in this project.

**Table 3: Implemented networks**

<table>
<thead>
<tr>
<th>Model Name</th>
<th>Speed(ms)</th>
<th>mAP</th>
<th>Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssd mobilenet</td>
<td>20</td>
<td>0.52</td>
<td>25,000</td>
</tr>
<tr>
<td>ssd mobilenet fpn</td>
<td>45</td>
<td>0.53</td>
<td>25,000</td>
</tr>
<tr>
<td>ssd resnet50</td>
<td>59</td>
<td>0.59</td>
<td>25,000</td>
</tr>
<tr>
<td>ssd resnet50 fpn</td>
<td>78</td>
<td>0.60</td>
<td>25,000</td>
</tr>
</tbody>
</table>

The four models were downloaded from the official tensorflow model zoo and were trained by COCO Dataset in advance. In order to train our own models by prepared datasets, tensorflow object detection API was utilized. It provides config samples, training and evaluating scripts and extracting tools that are capable of exporting weights to runnable graphs. Each model was trained for 25,000 steps by Egohands Dataset, which takes and the loss decreased to less than 2.5 and all the measurements of each model are done on a single NVIDIA® Tesla® K8 GPU. The speed refers to the average time to do hand detection on a 320 x 320 image and mAP represents the accuracy of detection. We evaluated the trained models by the same testing dataset and got the speed of SSD together with MobileNet model was about 20ms while that of ResNet50 was 59ms. After FPN algorithm was applied to each network, the speed decreased significantly to 45ms and 78ms respectively. We can draw a conclusion that ssd_mobilenet has the best performance on detection speed, which is consistent with the result in official tensorflow detection model zoo (see Figure 16 below)[6]. For the accuracy, the mAP of the ssd_mobilenet model was 0.52, which is smaller than 0.59 of resnet50. The performance was improved by adding FPN to models since the mAP of ssd_mobilenet and ssd_resnet50 are 0.53 and 0.60 respectively.

In addition to quantitative evaluation of the accuracy, a real-time application was implemented to test the speed and accuracy of each model. The application reads input images from the camera on the laptop and draws predicted boxes on the screen. As mentioned in section 3.4, the speed of a model can be represented by frames per second (FPS). Figure 17 shows the FPS of those models in the application. Since the laptop does not have a GPU, we found that the
FPS of pure MobileNet was about 8 while that of the model with FPN was only 1, and the FPS of ResNet50 was less than 1. Besides, it was observed that the detection performance of three models in the application was all acceptable, which means that almost every hand with a common pose can be accurately detected. Since speed is regarded as the bottleneck of better performance, ssd_mobilenet is the most efficient model for building a live application.

By the comparison of the above four models, we can draw the conclusion that ssd_mobilenet is the most suitable model with the highest speed and acceptable accuracy and should be further trained and evaluated.

**SSD_MobileNet Implementation** In order to decrease the total loss and improve the performance of the ssd_mobilenet model, more preprocessed datasets are added to train more steps (see Table 4). We used Egohands Dataset and Rendered Handpose Dataset to train the
model for 30,000 more steps. After that, as shown in the total loss of the model reduced to less than 1.8 and the mAP of detection was 0.52, reaching high time-accuracy efficiency.

Table 4: Datasets and Losses for different training steps

<table>
<thead>
<tr>
<th>Steps</th>
<th>Datasets</th>
<th>Loss</th>
</tr>
</thead>
<tbody>
<tr>
<td>0-25K</td>
<td>Egohands Dataset</td>
<td>2.5</td>
</tr>
<tr>
<td>25K-55K</td>
<td>Egohands Dataset, Rendered Handpose Dataset, Face-Hand Dataset</td>
<td>1.8</td>
</tr>
<tr>
<td>55K-82K</td>
<td>Egohands Dataset, Rendered Handpose Dataset, Face-Hand Dataset</td>
<td>1.8</td>
</tr>
<tr>
<td>82K-107K</td>
<td>Egohands Dataset, Rendered Handpose Dataset, Face-Hand Dataset, Caltech Human Face Dataset</td>
<td>1.8</td>
</tr>
</tbody>
</table>

Figure 18: Loss of ssd mobilenet

However, in the process of non-quantitive evaluation, which means real-time test in the application, faces appeared in the images were easy to be recognized as hands and be detected out. After analyzing the similarity of these two different classes, we came with three possible explanations: (i) same skin appearance of the two objects; and (ii) the high-level features of faces are similar to those of hands; and (iii) there was no sufficient images in the training set that contained faces in the background. The first two reasons took the features of faces and hands in consideration and it is challenging to change them, thus we attempted to solve this problem by adding more faces in the background. Face-Hand Dataset was utilized to train the model, in
which there were faces with different expressions in the background and a labeled hand. After 27,000 steps’ training, the loss and mAP remained almost the same as those before and the number of faces being detected in the test dataset decreased. Meanwhile, incorrect detection of faces appeared less when using the evaluation application, leading to a better intuitive performance of the model.

Unfortunately, although the usage of Face-Hand Dataset reduced the number of false positive detection of faces, incorrect cases were still found in the evaluation, especially for some particular expressions. Thus, we decided to train an extra detector of faces in the same model. Since the original model was trained by COCO Dataset, which contains 80 different object classes, it is not difficult to add a face class to the model. Caltech Human Face Dataset was taken advantages of as the training dataset for the new class, while Egohands Dataset, Rendered Handpose Dataset and Face-Hand Dataset were still in use to improve the performance of the hand detector. We trained the model for 25,000 more steps until the loss decreased to 1.8(see Figure 18). Then the accuracy of the two classes was evaluated, but mAP was not a suitable metrics to because there was more than one class. As a result, we used Average Precision (AP) to represent the detection performance. Although AP of hand class was only around 0.25, it is sufficient for the model to distinguish hands from faces and the false positive detection seldom appeared any more.

After the above implementation steps, the ssd_mobilenet model was capable of detecting the positions of hands in an image, providing the precisely cropped image for the second and third stages.

4.2 Hand segmentation

Compared with hand detection, models recognized the rough outlines of hands and output images with hands and black background (see Figure ?). Similar with hand detection, main process contains two steps: (i)data preprocessing; and (ii)model implementation and evaluation.

4.2.1 Data preprocessing
For the hand segmentation, Egohands Dataset and Rendered Handpose Dataset were applied since these two are both annotated by hand masks. As illustrated in Figure 19, the format of masks is one-channel PNG images in Rendered Handpose Datasets, and the grey level of masks in PNG images is 76 while that of background is set to 0. Masks in Egohands Dataset are stored as the coordinates of bounding points showing the outlines of hands in MAT files. Red crosses shown in Figure 20 represent the bounding points, and the polygons surrounded by points are masks of hands.

![Sample image](a) Sample image ![Sample mask](b) Sample mask

Figure 19: Images and masks in Rendered Handpose Dataset

![Sample image](a) Sample image ![Sample mask](b) Sample mask

Figure 20: Images and masks in Egohands Dataset

In order to train the segmentation model with the two datasets simultaneously, the masks need to be transformed to the same type, which is PNG images here. Therefore, data preprocessing should be done on Egohands Dataset, including reading MAT files to get coordinates of bounding points of hands in each image, filling the polygons surrounded by those points by matplotlib library and generating mask images. After the above steps, masks of all the images became to the same format.

Furthermore, to avoid the problem that the trained model can only detect single side hand, which means left or right hand, and to increase the size and complexity of datasets, several approaches of data augmentation were applied to the overall dataset. Table 5 show the augmented percent-
age of all images and the augmentation methods used.

<table>
<thead>
<tr>
<th>Data Augmentation Method</th>
<th>Percentage(%)</th>
<th>Output number of images</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mirror</td>
<td>20</td>
<td>9,200</td>
</tr>
<tr>
<td>Random rotation</td>
<td>10</td>
<td>9,200</td>
</tr>
<tr>
<td>Random Crop</td>
<td>10</td>
<td>4,600</td>
</tr>
</tbody>
</table>

Overall, since the size of Egohands Dataset and Rendered Handpose Dataset is 41,256 and 4,800 respectively, the total number of images is 64,900. Then the whole dataset was separated into a training set (58,500 images) and an evaluation set (6,400 images).

### 4.2.2 Model implementation and evaluation

The implementation of Mask RCNN model is similar to that of models in hand detection, which made use of the tensorflow object detection API to train, test and extract graphs. In total, the model was trained for about 40,000 steps. The total loss function is illustrated in Figure 21, where we can find that the loss kept fluctuating between 0.6 and 1.2.

![Figure 21: Loss of Mask RCNN](image)

To evaluate the performance of the model, both intuitive and quantitative evaluation methods are applied to the trained model. With regard to the intuitive evaluation, a python script was written to paint the estimated masks on the image. Figure 22 shows sample predictions on the seven selected gestures. It can be observed from the figures that the segmentation for '5', 'S' and 'Y'
was relatively precise while the masks of rest gestures ignored details of fingers or cannot cover the whole hand at all. Although this evaluation approach cannot provide statistic information on the performance of the model, it is capable of suggesting specific weaknesses of the model, such as the mask cannot recognize fingers in some poses.

![Figure 22: Estimated masks of seven poses](image)

The quantitative evaluation of Mask-RCNN is represented by the value of IoU. Table 6 shows the average IoU of seven hand poses. There were 100 test images per pose and the result of IoU reflect the accuracy of segmentation. It is not surprising that the gesture ‘5’ gained the highest IoU score, 0.6559, since it is the most common poses appeared in daily life and datasets. Gesture ‘Y’ won second place with the score 0.5713 while the IoU scores of ‘A’ and ‘S’ equal to 0.5403 and 0.5012 respectively. Considering the complexity of background, it can be regarded as a good segmentation case if the IoU score is larger than 0.5. However, the scores of ‘F’, ‘I’ and ‘L’ are lower than our standard. After analyzing the images in datasets and the similarity of these three hand poses, we came with two possible explanation: (i) the number of samples that contains ‘F’, ‘I’ and ‘L’ gestures is smaller than that of gesture ‘5’, ‘Y’, ‘A’, ‘S’; and (ii) the structures of low-IoU poses are more complicated than those with high scores. The first solution to this low-accuracy problem is related to the expanding of training datasets. More training images with gestures, of which the segmentation performance cannot fulfill our expectations, should be put into the training set. However, after searching in public resources, we found that it is challenging to gain the datasets that are precisely suitable for our hand poses. As a result, our own datasets, which consists of images of seven hand poses and their masks, are supposed to be created for the training of hand segmentation procedure. Since the process of
Table 6: Average IoU scores of seven poses

<table>
<thead>
<tr>
<th>Gesture Name</th>
<th>Average IoU Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.6559</td>
</tr>
<tr>
<td>Y</td>
<td>0.5713</td>
</tr>
<tr>
<td>A</td>
<td>0.5403</td>
</tr>
<tr>
<td>S</td>
<td>0.5012</td>
</tr>
<tr>
<td>F</td>
<td>0.4500</td>
</tr>
<tr>
<td>I</td>
<td>0.4327</td>
</tr>
<tr>
<td>L</td>
<td>0.3788</td>
</tr>
</tbody>
</table>

Labeling will cost a large number of resources and the time is limited, this task has not been done yet, but it becomes one of the research directions in the future. Except for adding more data into datasets, it is possible to improve the precision by training more steps. Currently, the batch size was set to 64, and the model has been trained for 40,000 steps, things may be changed if the model can be trained for 100,000 or more steps. Furthermore, some other models like Mask RCNN combined with ResNet50 and ResNet101 have a significant increase in precision with a much slower speed. Figure 23 shows the comparison of segmentation models in the tensorflow model zoo.

Figure 23: Segmentation models in the tensorflow model zoo

Apart from the accuracy, it attaches great importance to the efficiency of the model. On a Nvidia K80 GPU, it took about 5 seconds to do segmentation on a 1280 * 720 images, which is acceptable for image processing but far from the requirement of the real-time demo. The pretrained model we used, Mask RCNN, is the segmentation model with the highest speed under the sacrifice of precision. To enhance the efficiency of segmentation, we need to consider some state-of-art segmentation models that are not included in the model zoo or models implemented under other frameworks such as Keras and PyTorch.
# 5 Timeline

Table 7: Schedule

<table>
<thead>
<tr>
<th>Time period</th>
<th>Events</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.30</td>
<td>Detailed project plan and project website</td>
<td>Completed</td>
</tr>
<tr>
<td>10.1 - 10.15</td>
<td>Further research on papers</td>
<td>Completed</td>
</tr>
<tr>
<td>10.15 - 10.30</td>
<td>Data preparation</td>
<td>Completed</td>
</tr>
<tr>
<td>11.1 - 12.15</td>
<td>Model implementation</td>
<td>Completed</td>
</tr>
<tr>
<td>12.15 - 12.31</td>
<td>Model evaluation</td>
<td>Completed</td>
</tr>
<tr>
<td>1.7</td>
<td>First Presentation</td>
<td>Completed</td>
</tr>
<tr>
<td>1.20 - 3.15</td>
<td>Performance optimization</td>
<td>Completed</td>
</tr>
<tr>
<td>3.15 - 4.15</td>
<td>Live demo</td>
<td>Pending</td>
</tr>
</tbody>
</table>
6 Future works

More work can be done to enhance the performance of stage 1 hand segmentation model. Currently, since the precision and efficiency of Mask RCNN cannot reach the bottom line, it cannot be used to segment hands for stage 2 and stage 3. Thus, we are expected to train and evaluate other segmentation models, such as Mask RCNN combined with ResNet, to gain a better time-accuracy trade-off. Meanwhile, the corresponding datasets focusing on the seven selected poses are supposed to be built as the supplementary of current segmentation datasets. In this process, some devices can be utilized to reduce the workload of making annotations of masks. For example, gloves with sensors on the edges and joints are able to reflect the positions of joints and the boundary points of hands to the computer. As a result, we do not need to label images by ourselves.

Apart from improving the performance of models, applications made use of our project can be developed in the future. For instance, a drawing board application will recognize the change of hand poses and the path of hand movement by the models and draw relative patterns on the screen. The transformation from model researches to the applications in practice will be our major direction in the future.
7 Conclusion

Hand pose recognition is of great significance since it can be widely applied in human-machine interaction fields such as AI, VR, and AR. The hand recognition process is separated into three stages: hand detection, joint recognition, and pose estimation. Stage 1 aims at detecting the position of hands in images and segmenting the detected hands.

For hand detection, several models were trained and compared. The model with the best time-accuracy tradeoff, SSD combined with MobileNet, was implemented and evaluated. A real-time application built based on this model. For hand segmentation, precision on some hand poses is satisfactory while that of others cannot reach the requirement. To enhance the performance of the segmentation model, we should create additional datasets and review other models.

In the future, models are expected to be used in application development.
References


