Final Project Report for CS 175, Spring 2018

Project Title: Hand Pose Estimation
Project Number: 3

Student Name(s)
Qiao He, 16765414, qiaoh3@uci.edu
Lijun Ma, 78168109, lijunm@uci.edu
Zihao Chen, 90231032, zihaoc1@uci.edu

1. Introduction and Problem Statement

This project uses Transfer Learning on Tensorflow Hand detection and then Convolutional Pose Machines[1] model to accomplish hand pose estimation, training on images from CS175 Assignment 1&3, with evaluation on the correctness of labeled joints of images and videos.

2. Related Work:

The project are implemented with an adapted version of Convolutional Pose Machine on 2D hand pose estimation model built in Tensorflow to track the hand motion in joint pattern, with the aforementioned datasets prepared to train. Divided into two phases, hand detection and hand pose estimation, the first phase focuses on detecting hands on the images. Input with images containing hands, we applied a pre-trained object detection model and generate hand bounds on each image. Cropping according to hand bounds, these processed photos that are annotated with 21 joint points each were used as the training data of second phase for training process. By executing our hand pose estimation model, hand joints can be captured and we finally track the joints reversely from cropped images to original ones.

For the first phase our earlier work was done on detecting the hands with the earliest dataset from Assignment 1&3, it works pretty well on the the hand gesture we have in the dataset, but we tested other images which may contain different hand gestures, different hand sizes or different brightness, the results were not very ideal, so that we systematically added more images that were shifted with the features of rotation, brightness and mirroring based on the original training images we have into the dataset, and ultimately achieved a better result on hand detection. For the second phase, by getting a better result from the first phase, we are able to crop the correct hand to the greatest extent and help the convolutional pose machine model we trained to label the joints more precisely.

Reference:
[1] S.Wei, V.Ramakrishna, T.Kanade, Y.Sheikh. Convolutional Pose Machine. In CVPR 2016: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/Wei_Convolutional_Pose_Machines_CVPR_2016_paper.pdf
[2] D.Victor. How to Build a Real-time Hand-Detector using Neural Networks (SSD) on Tensorflow https://towardsdatascience.com/how-to-build-a-real-time-hand-detector-using-neural-networks-ssd-on-tensorflow-d6bac0e4b2ce

3. Data Sets

The dataset we use is our self-built hand dataset from CS175 Assignment 1&3 and our self-constructed hand pose dataset. Our self-built hand dataset will play a key part in phase II, and an ancillary role in phase I.

The dataset from assignments was collected from 2-minute video generating to 57,207 images from 137 students in CS175 course in front of a Kinect camera showing hand movements. These data was later on be annotated in 21 hand joint positions on each hand total 94,126 annotated labeled data and used to train machine learning models for hand detection and pose estimation. This dataset, with high diversities in hand pose of different students and percentage of hidden part of hand, will help to aggrandize the exhaustiveness of our model.

Although the dataset including the massive amount of labeled data covers diversity in hand pose in many different people, it is still insufficient when meeting some particular scenes to do hand detection. We notice when do hand detection in some images, like there is similar color between hand and background, some more particular hand pose not covered in dataset, or the size of hand changed to bigger or smaller when nearby or far away from camera in photo taking, it is difficult to recognize the hand in image. There are several reasons could causing these scenes happen. The illumination condition and background when collecting 2-minute video is monotonous, the types in differentiate of hand pose are limited, the ratio of hand size in image is constant, and there are some images with hand joint labeled in imprecise positions.

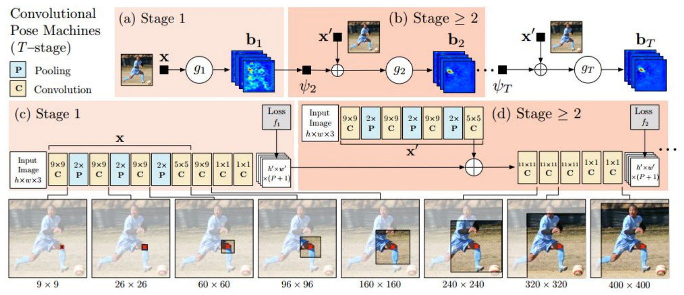
Base on previous situation we add more self-constructed hand pose dataset including 17,000 more images. We pick up images covering all students to ensure diversity of hand shape in original dataset from assignment 1&3 to generate images in rotating 90 and 270 angular and illumination adjusted, and generated 30,000 more updated annotated hand joint positions labeled data. Then combine all self-constructed hand pose dataset with original dataset.

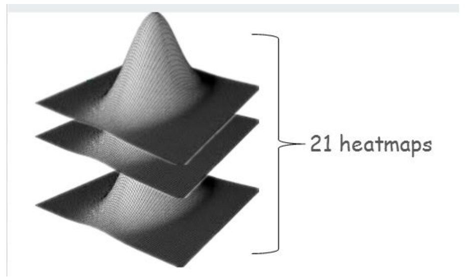
The super dataset including more than 75,000 images, around 125,000 labeled data splitting into train (80%), test (20%) folders will be trained together with annotations will be crucial in hand pose training process.

4. Description of Technical Approach

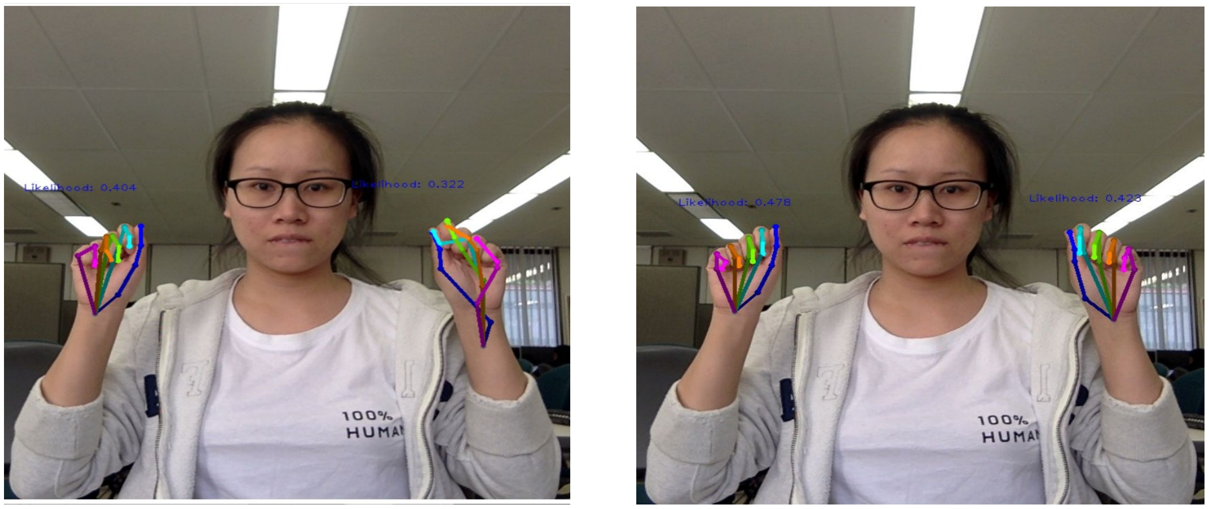
In the first phase, hand detection, we use Tensorflow, object detection API. It is possible to use transfer learning process to shorten the amount of time needed to train the entire model with neural networks.Transfer learning is an optimization that allows rapid progress or improved performance when modeling the next task. Tensorflow does offer a few models and we chose to use the `ssd_mobilenet_v1_coco` model as our start point given it is currently one of the fastest models. The training process would be done on Google Cloud GPU machine. With transfer learning from the Tensorflow detection model, *ssd_mobilenet_v1_pets.config,* we are going to retrain with our dataset, on this model to build a hand detector. As the training process progresses, the expectation is that total loss gets reduced to its possible minimum.

In the second phase, hand pose estimation, the model we have applied is the Convolutional Pose Machine, which is a state-of-the-art 2D model for body pose estimation. The model divides the layers in to N stages, where each stage takes inputs and predict outputs that are of equivalent size, in which way the a latter stage can take anterior stage' output as input. From any i-th stage from the second to the N-th, a fixed sequence of layers, *X'* , will also make an equal-size prediction concatenating to (i-1)-th stage's output to generate the input of i-th stage. This methodology is demonstrated with the left figure.





With each stage's input defined, the output of stages are in the form of heatmaps. Each stage outputs a 22-layer heatmap, including one heatmap per joint and a background. The heatmap stores a 2D Gaussian Distribution on the joint's probability to be located at each pixel in the cropped image, and therefore we can visualize at in 3D graph as the right figure.
Using stages have several pregnant benefits: Firstly, the second to the N-th stages can be implemented with a for-loop, which aggrandize the efficiency of layer modification; secondly, intermediate results can be supervised by presenting the heatmaps at each stage; last but not least, the aggregated loss of all stages is used for back propagation, which solves the significant problem of vanishing gradient, by replenishing gradient with *X'* at each stages, and thus overcomes the bottleneck that snags deep nets.
Moreover, for both phase 1 and 2, we attach great importance to data preprocessing. The intuition is straightforward while with great improvement. We reuse the existing datasets by rotating the images, mirroring images and adjusting the illumination condition. With rotated image as input,our model is capable of estimating hand joints in different directions; with illumination variation, hand can be recognized in places that is darker or brighter than standard. Remarkably, as we observe that in some gestures like fists, right hands are well-labelled while left hands are relatively pool, we do mirroring so that left and right hands can train with each others' dataset. The improvement is significant, as illustrated below, before mirroring and after mirroring.



We can see with mirroring, left hand estimation has been largely improved, to similar performance of right hand

5. Software

a) Code we have written

Name of the script	Description
phase1_json_to_csv.ipynb (preprocessing)	Convert Json file to CSV file; Contains function to draw the bounding box according to the maximum value of x, y for the 21 joints labeled on the hands, to check the accuracy of the joint labels.
phase1_split_partition.ipynb (preprocessing)	Randomly split the dataset into training data and test data with the partition of 8 : 2
phase1_illumination.ipynb (preprocessing)	Change the brightness of input images which covering all the people as the new dataset to be added into the CSV file
phase1_rotate_270.ipynb (preprocessing)	Rotate the input images by degree of 270 where covering all the people as the new dataset to be added into the CSV file
phase1_rotate_90.ipynb (preprocessing)	Rotate the input images by degree of 90 where covering all the people as the new dataset to be added into the CSV file
phase1_prediction.ipynb (prediction)	Give a image, find the hand and draw the bounding box, show the result of phase 1
Detector.py (tool for phase 2)	Contain functions for prediction for phase 1 and return the x, y limit of the bounding box to let the phase 2 crop the image to be able to get the image which contains hand in the middle area. Contain tool function for phase 2 that convert each joint into gaussian heat map during the training process

phase2_tfrecord.ipynb (preprocessing)	Generate tfrecord from json file. Given the original image from the json file, use the Detector.py to find the bounding box for the image, crop it, add some paddings and resize it to 256x256 pixels, then save the cropped image and the relative 21 joints on the cropped image into the tfrecord. And split the record into training and test tfrecords.
Ensemble_data_generator.py (training)	Load the tfrecord, iterate the data and generate batch of data as the input for training process for phase 2
training.py (training)	Train the tfrecord with the cmp_model to get the optimized hyper parameters for phase 2
cpm_model.py (training)	Constructing first stage layer contains the stage_x to be concat with output layer of each stage as the input layer of next stage. Create the loss function to train the loss and optimize the hyper parameters for phase 2
config.py (training)	Contains all the pre-set variables to be used in the training process for phase 2
project.ipynb (prediction)	Predict test images on hand joints. Provide the training result with the cropped image and session will return 22(21 joints and 1 background) gaussian heat maps, find the location of the highest value in the gaussian to be the position of each joint, then calculate the position of the 21 joints in the original image, then label all the joints on the graph. By changing FLAGS.DEMO_TYPE in config.py, can support predict image, some or all images of a folder, or predict a video
video2jpg.py	Convert the video into series of images
jpg2video.py	Convert the series of predicted images back into video

b) Code adapted from other people

Name of the script	Description
phase1_tfrecord.ipynb (preprocessing)	Convert CSV file to tfrecord
ssd_mobilenet_v1_pets.config (phase 1 object detection model)	Pretrained model from tensorflow object detection to be our starting point of training for the phase 1
train.py (phase 1 training)	Load training and test tfrecords, ssd_mobilenet_v1_pets.config.

	Train through all the records
export_inference_graph.py (phase 1 training)	Convert the training result into frozen graph as the input for the prediction of phase 1

6. Experiments and Evaluation

For the hand detection phase, we expect to achieve higher accuracy on hand recognizing and drawing the bounding box around the hands. For the hand pose estimation phase, we will evaluate the accuracy of hand joint labeling based on human visual judgement in sequential images input with diversities in hand pose, background and illumination. We decided to have fixed train-test partition on phase I with training sets 80% and validation sets 20% folders. With transfer learning from the Tensorflow detection model, *ssd_mobilenet_v1_pets.config,* we are going to retrain with our dataset, on this model to build a hand detector. In hand detection process, the detection scores represents the likelihood of an object to be of that class with only greater than 0.5 and pick up the top 2 range in detection scores list will be shown in visualized boxes.
Before adding updated self-constructed hand pose dataset including rotation and illumination adjusted, we observed the hand detection and found the accuracy was 70% around. The object detection would be confused in similar color between hand pose and other objects, the different hand shape never shown in dataset and recognition in face instead of hand. After adding hand pose dataset in rotation and illumination adjusted with same ratio fixed train-test partition, we did training process with transfer learning continually. We observed the hand detection and found the accuracy was changed to 90% around. The hand detection could recognize the hand precisely with particular hand pose in any direction and did not confuse in illumination. There are still a few tricky hand pose could not be recognized because of obviously different with dataset and could be overcome and get improvement if we add more types of self-constructed hand pose future.

For the hand pose estimation phase, we use three distinct approaches to comprehensively evaluate and justify the predictions. Firstly, we keep track of training loss for each iteration, which validate the correctness of our model. Validation loss is calculated for every 100 training iterations, which takes average loss of 100 validation records, checking the potentiality of over-fitting. Different from classification or regression problems, the loss cannot be used as the only method for evaluation, as same magnitude of loss may tend to be disparate gestures when visualized and the real performance of model is hard to quantify. Therefore, apartment from validation set, we also apply the approach of user studies. We build a test sets as the metrics, which takes different illumination conditions, rotations of hands and gestures into consideration, so as to exhaustively research the strengths and weaknesses of our model. Thirdly, we take a standard test video and predict it as a sequence of photos. With this approach, we are capable of evaluating the fluctuation of the current model in time series.

With the aforementioned methodologies, we obtain several form of result for comparisons. We keep a table of validation losses after each training, which helps us to monitor the significance of parameters and layers, and determine which model transcends the others and would be kept training. Additionally, we keep the predictions result of the standard test set at each training time, by comparing which we will know which training sets of what gestures or illumination level should be underscored in the next step correspondingly. With regard to the standard video, it helps to compare the stabilities of models that are relatively accurate in single photos, and further winnow a model for sequential prediction.

7. Discussion and Conclusion

Throughout this project, we gain insights from several perspectives. Firstly, we comprehend the full cycle of a deep learning project by designing the pipeline, and then step by step accomplishing data collection and preprocessing, model building and training, and eventually evaluation. Besides, we acquire experience of reading academic papers, by which we also understand the intuitions of designing deep learning models.Last but not least, we learn how to balance the time between running model and coding, as well as cooperate in a relatively long term project as a team.

Basically, we are pretty satisfied with the result we got, for the most test image we predicted, we able to observe relatively correct labelling of the joints. But there is still space that we can improve, such as the hand gesture of fist, since some joints are hidden, so the predicted result of labelling did not really achieved our expectation as the other simpler hand gestures. If we have the future opportunity to continue with this project, we might want to improve the model to be able to detect the hand joints with any type of occlusion, we hope the model can be more powerful and smart enough to be able to predict the approximate positions of the joints that are hidden in the images. What is more, we may want to learn how to train the model with 3D depth images, which could better reduce the noisy factors and acquire more features for training. At last, we again underline the importance of dataset that, if we can improve the label accuracy of dataset by manually cross checking, the precision of the model will be promised and the oscillation in video prediction will be minimized.