Hand Detection Using CNNs

Sameera Bammidi Yimeng Li

1. Abstract

In this project, we use Single Shot Detector (SSD) algorithm which utilizes Convolutional Neural Networks (CNNs) to detect all scaled hands in images. Using hand centric and complex image datasets, we train the weights for real-time hand detection in each image or a video frame. We limit ourselves to hand recognition. We do not do gesture recognition in this project. We train the dataset by supplying, formatted ground truth files, a configuration and weights file. Using these trained weights, we test the simple hand centric images and complex images which are daily life pictures and where there are many other complex features other than hands which are similar to hand features. We shoot a video where the moving hands are focused on a black background with limited noise. We extract frames from the video and test if the moving hands are detected. If there are any false positives, we analyse why those are detected.

2. Introduction and Related Work

We aim to detect and localize human hands in still images or videos. This is a challenging task since hands have multiple poses, can be closed or open. And hands vary a lot under different viewpoints. For example, hands under egocentric view and hands in human-taken photos don't have much overlapping. To build a hand detector for general usage, we intentionally chose two hand datasets hoping to cover all the general cases. We choose Oxford dataset [1] since its data comes from daily life photos. EgoHands dataset [2] is chosen since it contains images under egocentric viewpoint, which is not covered by Oxford dataset.

Hand detection is a a quit young problem in computer vision, comparing to other old problems, like pose estimation. It has been tackled many times by other researchers, but from my view, we still lack a hand detector working fine under general conditions. This is primarily due to the fact that hands are deformable in nature. Hand detection and pose estimation has quite a lot applications in robotics, human computer interaction, action recognition and sign language recognition [4].

Multiple human designed cues have been used to detect hands in the past. Kolch and Mathius [5] used haar features and cascaded classifiers, similar to techniques used in face detection. Pisharadi [6] used saliency information to help locate the hand. And to define saliency, he majorly used texture features. Zisserman [1] used SVM learned cues and sliding window techniques to detect hands and achieved state-of-the-art results on Oxford dataset.

In the past 5 years, to be more exactly, since 2012, Convolutional Neural Networks (CNN) has achieved huge success in multiple Computer Vision tasks. Apparently, someone has already tried CNN on hand detection problems. Huang [7] adapted Fast-RCNN into a more scale invariant network so it works well on complex real-life scenarios like detecting a driver's hands.

Deng [8] used a pose invariant CNN to do hand detection and pose estimation together without using depth data.

3. Technical Approach

Single Shot Detector (SSD) is one of the popular algorithms for object detection in pictures or videos. While detecting bounding boxes in a video, frames are extracted and bounding boxes in each frame are detected.

3.1 Understanding the SSD algorithm

Unlike other traditional CNNs (such as R-CNN which has Region Proposals Network + Classifier Network), SSD predicts bounding boxes and confidences for multiple categories in single network. Hence the name Single-shot multi detector.

On an input image, SSD runs a convolutional network only once and calculates a feature map by creating multiple bounding boxes and then keeping the ones with IOU (Intersection Over Union) threshold > 0.5. In different bounding boxes, small convoluting filters are used to predict object categories. Different predictors are used for different aspect ratios. It uses anchor boxes (region boxes which mostly contain the objects to be detected) at various aspect ratios and learns the off-set rather than bounding the box. SSD predicts bounding boxes after multiple convolutional layers. Each convolutional layer operates at a different scale, so, it is able to detect objects of various scales.

3.2 Architecture of SSD

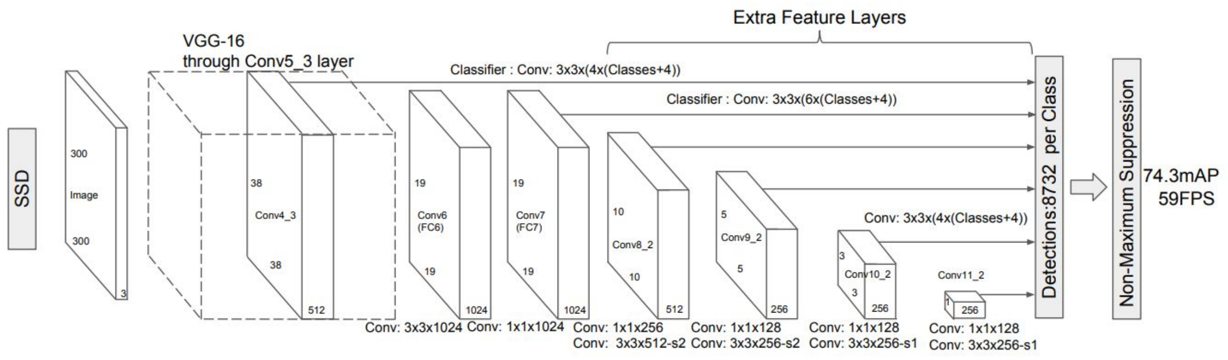


Fig reference: https://arxiv.org/pdf/1512.02325.pdf

Here the VGG-16 network (https://arxiv.org/pdf/1409.1556.pdf) is used as a base. Each added feature layer produces a fixed set of detection predictions using a set of convolutional filters. These are indicated on top of the SSD network architecture in above figure.

For a 3 channel, m x n size feature layer, a small 3 × 3 × p kernel is the basic element for predicting parameters of a potential bounding box. This kernel produces either a category score, or a shape offset relative to the default bounding box coordinates. An output value is produced at each of the m x n locations. Relative to each feature map location, bounding box offset output values are measured(compared to a default box position). [3]

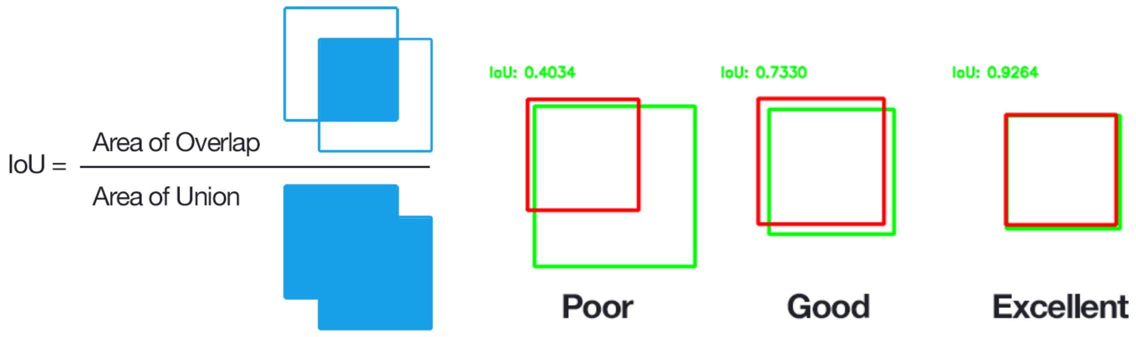


Diagram explaining IoU (picture reference: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/)

Default boxes and aspect ratios

There are four parameters associated to a bounding box: bounding box center coordinates(x,y), its width and height are needed for an object detection.

Each detector outputs a single value so, (Number of classes + 4) are needed for detection. But there are different aspect ratios for detection bounding boxes. Hence,

Number of detectors = (Number of classes + 4) x total number of default bounding boxes

For each box at a given location, class scores and the 4 offsets relative to the original default box shape are computed. This results in a total of [(Number of classes + 4)x Number of boxes] filters. These filters are applied around each location in the feature map, resulting [(Number of classes + 4) x Number of boxes x m x n] outputs for a feature map of size m × n.

3.3 Training objective

The SSD training objective is to handle multiple object categories. To match the ith bounding box to jth ground truth box of class P,

$$\text{Let } x^{p}_{ij}=\{1,0\}$$

In the matching strategy above, we can have $\sum_i x^{p}_{ij} \geq 1$.

Overall objective loss function = weighted sum of the localization loss (loc) + the confidence loss (conf)

$$L(x,c,l,g)=(1/N) * (L_{conf}(x,c)+\alpha L_{loc}(x,l,g)) \text{ ---------(1)}$$

N is the number of matched default boxes. Localization loss is the loss between the predicted box (l) and the ground truth box (g) parameters. For the four bounding box parameters regress to offsets. Confidence loss is the loss over multiple classes confidences (c). Weight term α is set to 1 by cross validation. X is the indicator as mentioned above. [3]

3.4 Configuration file and weights

We used SSD with Mobilenet v1, configured for Oxford-IIIT Pets Dataset. We configured the fine_tune_checkpoint field in the train config

```
fine_tune_checkpoint: "ssd_mobilenet_v1_coco_2017_11_17/model.ckpt"
```

Model.ckpt are the weights trained on coco dataset.

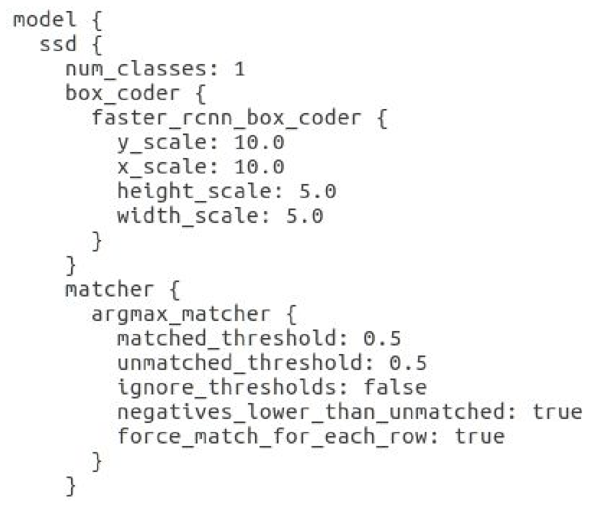
We modified all the label_map_path, input_path fields in train_input_reader as below:

```
label_map_path: "training/hand_detection.pbtxt"

input_path: "data/train.record"
```

In ssd, we changed the number of classes: 1

This is how model and matcher in configuration file look like:



```
model {
  ssd {
    num_classes: 1
    box_coder {
      faster_rcnn_box_coder {
        y_scale: 10.0
        x_scale: 10.0
        height_scale: 5.0
        width_scale: 5.0
      }
    }
    matcher {
      argmax_matcher {
        matched_threshold: 0.5
        unmatched_threshold: 0.5
        ignore_thresholds: false
        negatives_lower_than_unmatched: true
        force_match_for_each_row: true
      }
    }
```

We used weights pretrained on Common Objects in COntext (COCO) dataset. It has 330K images out of which >200K images are labeled. There are 1.5 million object instances and 80 object categories.

3.5 Datasets Used

1. Oxford dataset: This is a comprehensive dataset of hand images collected from various public image data set sources. A total of 13050 annotated hand instances are available. [1] http://www.robots.ox.ac.uk/~vgg/data/hands/
2. EgoHands dataset: This dataset contains 48 Google Glass videos of complex, first-person interactions between two people. The dataset offers 15,053 pixel-level labeled hands. [2] http://vision.soic.indiana.edu/projects/egohands/

3.6 Preprocessing

We downloaded these two datasets which came with annotations in .mat format and then preprocessed the data to make the labels compatible with the Tensorflow architecture.
Following is the procedure:
For Oxford hands dataset,

1. Read each .mat file associated with the corresponding image.
2. Get the minx, miny, maxx, maxy values of each box in the image.
3. Create a new rectangular bounding box with the above obtained values and write them to a .txt file

4. Get all the text files and print image filename, image width, image height, class, xmin, ymin, xmax, ymax for each bounding box in to a single csv file.

For EgoHands dataset,

1. Read the contours drawn around each hand segment associated to a hand and computing rectangular bounding boxes from those.
2. Do the steps 2 to 7 same as above.

3.7 Training steps

1. First we need to create tensorflow records based on the images and annotations. We cannot just use images and annotation text files as input for tensorflow. Everything need to be converted into tensorflow understandable data structures. In the data structure, all kinds of data are converted into binary codes and they have a map like structure to extract required data very efficiently.

   To do the conversion, we wrote a generate_tfrecord.py script by following an example. Then we run the generation scipt twice, once for the train TFRecord and once for the test TFRecord.

```
python generate_tfrecord.py --csv_input=data/train_labels.csv
--output_path=data/train.record
python generate_tfrecord.py --csv_input=data/test_labels.csv
--output_path=data/test.record
```

We use the above two commands to run the script. Parameter csv_script refers to the csv file. Parameter output path refers to where you want to save the generated tensor records file and how you want to name it.

2. Now we have the tensor records files, we can start the training process

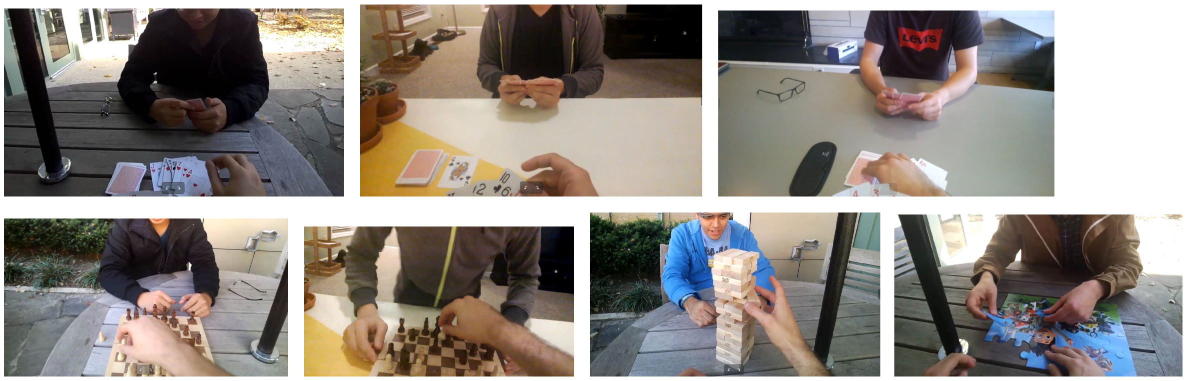
```
python research/object_detection/train.py --logtostderr --train_dir=training/
--pipeline_config_path=training/ssd_mobilenet_v1_pets.config
```

This is the command we use to start the train script under tensorflow object detection module. Train.py is the script, it starts the training session, reads in the tensor records and network configuration. Parameter train_dir refers to the directory we put intermediate weights and checkpoint files in. Parameter pipeline_config_path refers to the initial network configuration. Here we used ssd mobile net architecture.

3. When the training process starts, we can use tensorboard to monitor it. But since we are running tensorflow on ARGO, which is a cluster and it doesn't provide GUI for us to view the images. So instead of using tensorboard, we just follow the training log. It shows the loss after every step. We waited for approximately 10 hours until the loss became stably below 3. Then we stopped the training process and started the test session. [10]

Following are few examples of Oxford data set images we used for training:

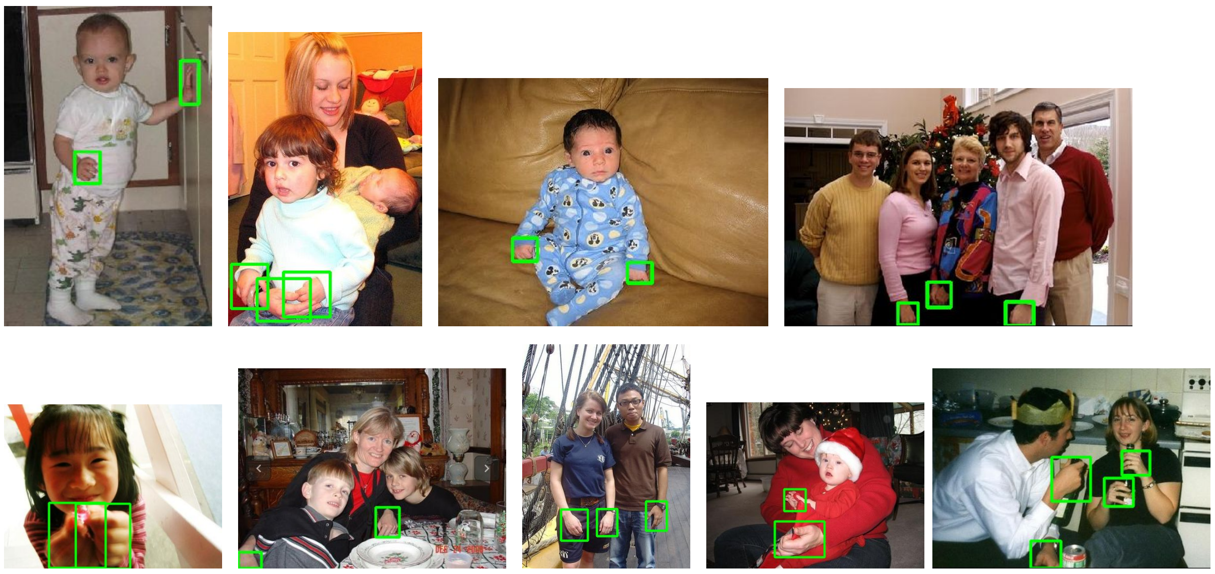
Following are few examples of EgoHands dataset images we used for training:



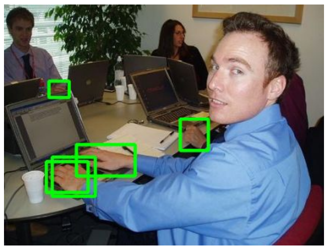
4. Results and Discussion

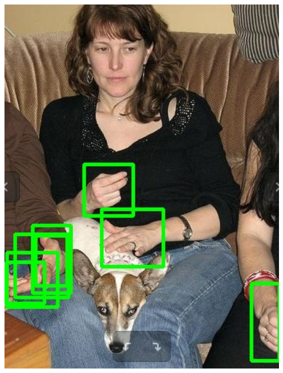
4.1 Oxford Dataset

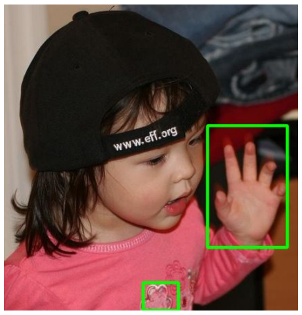
Following are the results of testing on Oxford data test set:



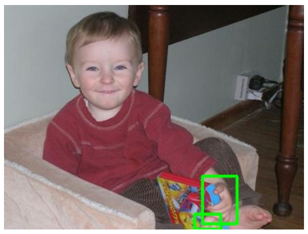
There are few false positives and false negatives in this training process. Following are few examples of these:

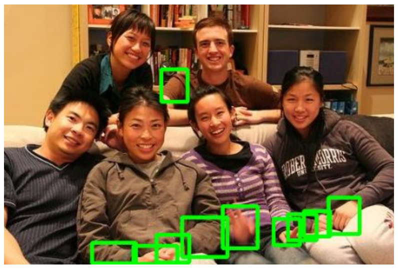

(a)

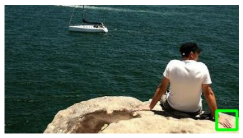

(b)

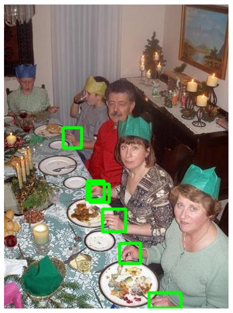


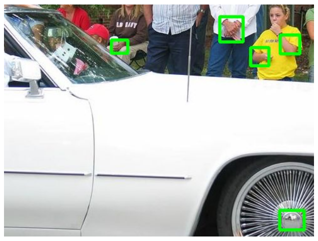
(c)


(d)


(e)


(f)


(g)


(h)

This is because,

1. The data set size is small (about 4000 images). This could work much better if the network is trained on more than 100000 complex images and used those weights to test.
2. The data is complex and noisy as shown in figures (e), (g) and (h). There are plenty of other features other than hands in these images.
3. The resolution is low. For example, image (g) has very low resolution. It is difficult for a human eye to predict these. Hence, more difficult for a machine to predict them.
4. Multiple bounding box's IOU value similar (say ~ 0.9). That is why in figures (a), (b) and (e) there are multiple predictions on a single hand.

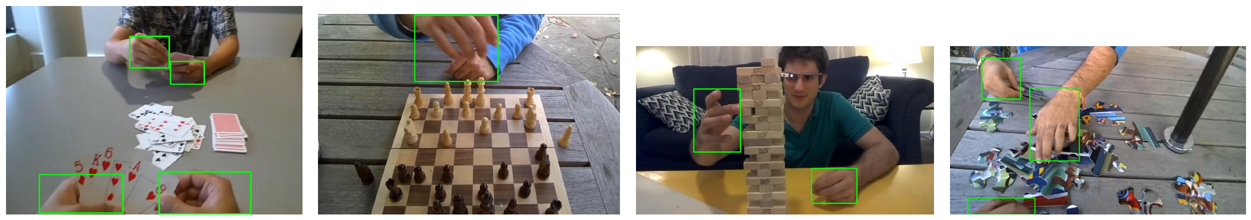
Also, in figure (h), the car wheel axel's tip is detected as hand In figure (c) the flower on the little girl's shirt is detected a hand. This is because of training on a small dataset. Though the iterations are more, it does not work in few cases.

In figure (d) the little boy's right foot is detected as hand but the left foot is not. This is because only the right foot's fingers are captured in the picture. This can be avoided by training on sufficient(1000 images) similar data.

In figure (f) the man's left hand is not detected which is a false negative. In figure (g) the left hand of last kid with blue cap is not detected. This is because the pictures have low resolution. This problem persists in SSD. It works better on the high resolution images. The background is also noisy. The can be reduced by sharpening the object and smoothening the background. Changing contrast also helps.

4.2 Egohands Dataset

Following are the results of testing on EgoHands data test set:



The results of this test data are more accurate when compared to the Oxford dataset results because the images are hand centric and the image data is less complex. Also, there is less noise.

After training SSD on the EgoHands dataset, we used those weights to test few images of the Oxford dataset which resulted in way less number of correct predictions.

Video test:

We shot a video of moving hands doing origami with paper and on a black background with less noise. We extracted frames from the video and ran the test. The SSD detected fair number of true positives.

4.3 Contribution of Sameera:

Initial attempt with YOLO framework:

For this project I initially preprocessed the ".mat" annotation files related to Oxford dataset and polygon mat files related to egohands dataset to obtain bounding box data in text file format. Then I converted them into yolo's input format text files.

Example original annotation:

0

232.652113093 271.58883655 260.758890143 289.746382686

0

258.650281868 264.670363851 287.265575737 290.192436714

Example yolo annotation:

0 0.593042071197 0.674681753889 0.0675643679087 0.0436479474423

0 0.656148867314 0.666902404525 0.0687867641082 0.0613511366899

Then I wrote script to recursively extract all the images from 48 different folders of egohands dataset rename them and place them in a single folder for training. Next, I wrote script to compress the images into size 416 x 416. Then, I used weights pre-trained on imagenet dataset

for over 100000 images and configuration file provided by darknet. The configuration file has 30 layers. Since hand detection is a single class detection problem, I modified the "class number" attribute of the configuration file to 1. Next, I changed parameter values *batch size* to 32 and *subdivisions* to 16 in the configuration file. In each iteration 32 images are processed in CNN with 2 images in parallel. This is done to ensure that the GPU machine does not run out of memory after empirical trials.
I trained oxford and egohands data sets separately on these. Each of the data sets contain about 4000 images. I stopped training after 10000 iterations because the error value dropped to under 12 from an initial error of over 400. My understanding of why this approach did not work as expected in the trials- It is possible that all my experiments were done with weights that are pretrained on multi-class detection and the resulting combination of CNN structure and pretrained weights is unsuitable for the single class problem at hand. Also, if I had more time I would have experimented with tuning the final convolutional layer of the configuration file to suit the single class detection problem.

Due to time constraints and lack of availability of a good GPU machine, I chose to not pursue this approach further.

Approach with SSD framework:

Next I joined Yimeng in using SSD network with Tensorflow architecture. While he was working on training and testing Hand detection on EgoHands dataset, I worked on training and testing Oxford hands dataset. I used the preprocessed ground truth text files in the previous attempt and converted them to the format compatible with tensorflow architecture and produced the CSV files for train and test data. [10]

Example CSV file format:

```
filename,width,height,class,xmin,ymin,xmax,ymax
VOC2010_74.jpg,500,375,hand,37,59,76,96
VOC2010_74.jpg,500,375,hand,192,133,211,158
VOC2010_74.jpg,500,375,hand,280,209,300,228
VOC2010_74.jpg,500,375,hand,297,57,316,75
```

I followed all the steps previously described in section 3.7 of this document. As described in the section 4.1, there were few false positives and false negatives obtained. To improve the performance of this network, if time permits, I would train it on a bigger dataset with higher resolution and specific error prone images which cause false positives. And perhaps, categorize common false positives such as cropped feet with toes in the image frame.
The code I wrote is attached to the email.

4.4 Contribution of Yimeng:

I did the literature review and designed the project. Then I finished the experiment using SSD. I did not contribute much to the writing of final report.

Conclusions

After our experiments, we found out that Single Shot Detector(SSD) algorithm produced good results. Using hand centric and complex image datasets, we trained the weights for real-time hand detection in each image or a video frame. We analysed why it produced some false positives and false negatives. A bigger dataset with over 10000 images and better training would produce lesser number of false positives and false negatives. We did not get to to the part to observe numerical test results such as average precision. If time permits, we would do this and also, train the skin segmentation network and hand detection network to get better results.

References

[1] http://www.robots.ox.ac.uk/~vgg/data/hands/
[2] http://vision.soic.indiana.edu/projects/egohands/
[3] Buehler, Patrick, et al. "Long term arm and hand tracking for continuous sign language TV broadcasts." *Proceedings of the 19th British Machine Vision Conference*. BMVA Press, 2008.
[4] Liu W. et al. (2016) SSD: Single Shot MultiBox Detector. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9905. Springer, Cham
[5] Kölsch, Mathias, and Matthew Turk. "Robust Hand Detection." *FGR*. 2004.
[6] Pisharady, Pramod Kumar, Prahlad Vadakkepat, and Ai Poh Loh. "Attention based detection and recognition of hand postures against complex backgrounds." *International Journal of Computer Vision* 101.3 (2013): 403-419.
[7] Le, T. Hoang Ngan, et al. "Multiple scale faster-rcnn approach to driver's cell-phone usage and hands on steering wheel detection." *Computer Vision and Pattern Recognition Workshops (CVPRW), 2016 IEEE Conference on*. IEEE, 2016.
[8] Deng, Xiaoming, et al. "Joint Hand Detection and Rotation Estimation Using CNN." *IEEE Transactions on Image Processing* 27.4 (2018): 1888-1900.
[9] https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/
[10]https://towardsdatascience.com/how-to-build-a-real-time-hand-detector-using-neural-networks-ssd-on-tensorflow-d6bac0e4b2ce
[11] https://pjreddie.com/darknet/yolo/
https://timebutt.github.io/static/how-to-train-yolov2-to-detect-custom-objects/
[12]https://github.com/AlexeyAB/darknet#you-only-look-once-unified-real-time-object-detection-version-2
[13]https://pjreddie.com/darknet/
[14]https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/
[15]https://github.com/thtrieu/darkflow
[16]https://github.com/tzutalin/labelImg
[17]https://cosmosmagazine.com/technology/what-is-deep-learning-and-how-does-it-work

[18]https://github.com/Guanghan/darknet/blob/master/scripts/convert.py
[19]https://stackoverflow.com/questions/874461/read-mat-files-in-python
[20]http://cmdlinetips.com/2012/09/three-ways-to-write-text-to-a-file-in-python/
[21]https://stackoverflow.com/questions/678236/how-to-get-the-filename-without-the-extension-from-a-path-in-python?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa
[22]https://github.com/tensorflow/models/blob/master/object_detection/samples/configs/ssd_mobilenet_v1_pets.config