# Project Report

Zhu Zhehao
zzh101596@gmail.com

May 7, 2019

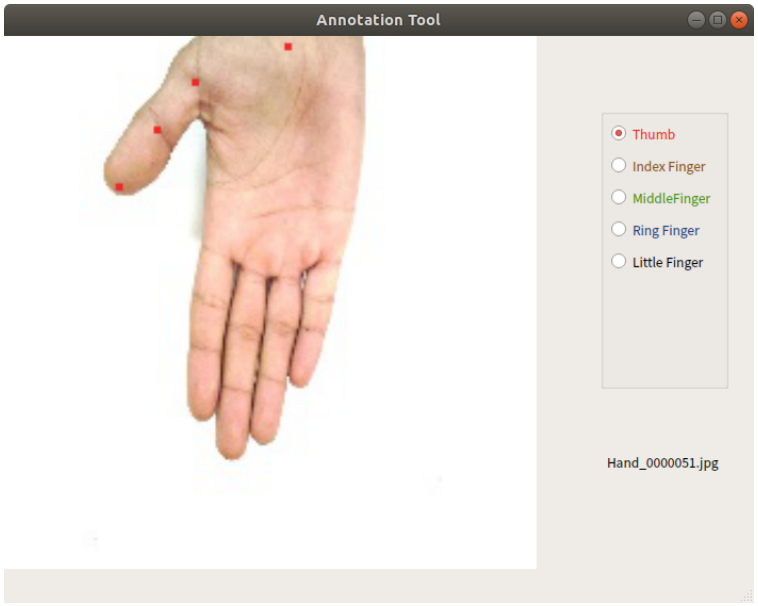
## 1 Annotation Tool

The Annotation Tool was implemented in PyQt. The following functions were inplemented:

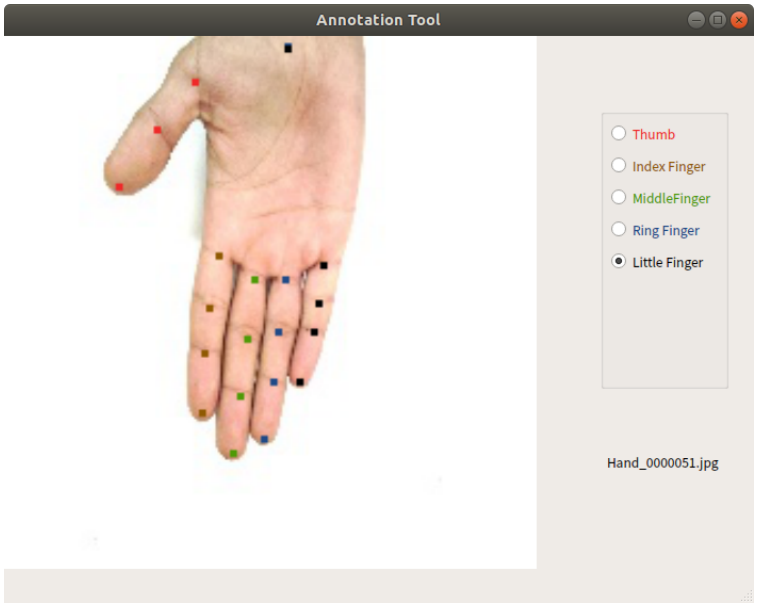
- Display a keypoint at clicked place.
- Store class information for each finger.
- Assign 's' key for saving a set positions to a file. The file format is defined as .txt file and the data is saved in json format.
- Assign 'n' key for loading next image and automatically save the last set positions.
- Assign 'r' key for refreshing the current image and clear all the points already annotated.
- Assign 'Tab' key for switching the current class of fingers.

The following image is a screenshot of the Annotation Tool.

To annotate the key points in each class of fingers, you should first click on the corresponding radio button. For example, if you want to annotate the key points of Thumb, you should click the first button in the *finger class selector* like the following screenshot shows:

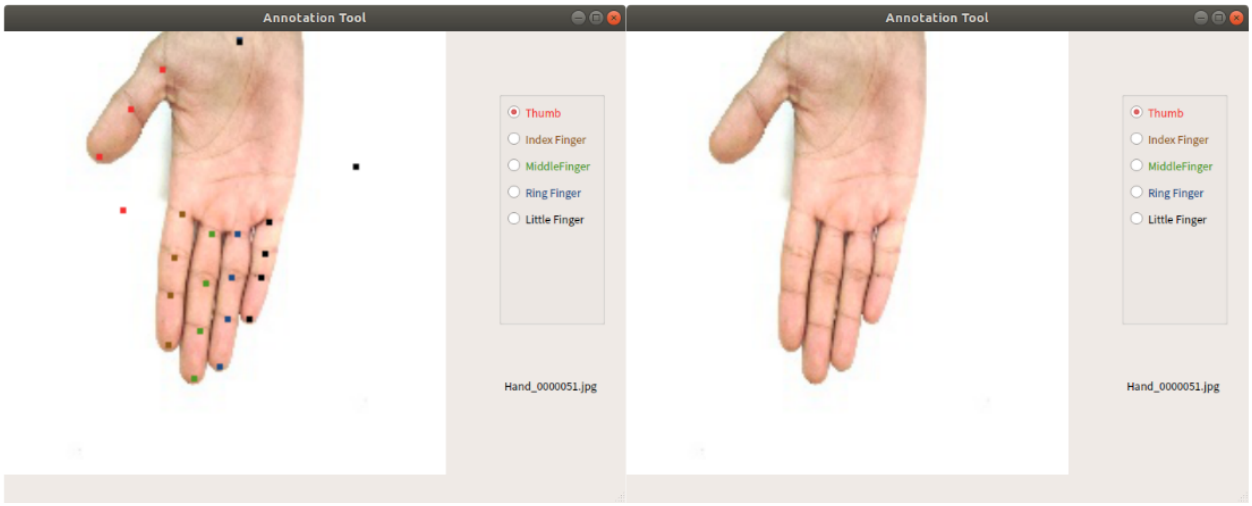



After annotating all the key points belonging to a finger, you should click the radio button of next finger you want to annotate or press Tab to switch the button.




When you have finished the whole annotation of five fingers like the above image, you should press 's' to save the annotations or press 'n' to save the current annotations automatically and load next image.

If you incorrectly click the screen and lead to wrong points annotated, you can press 'r' to refresh the current image and clear all the current points like the following screenshots:

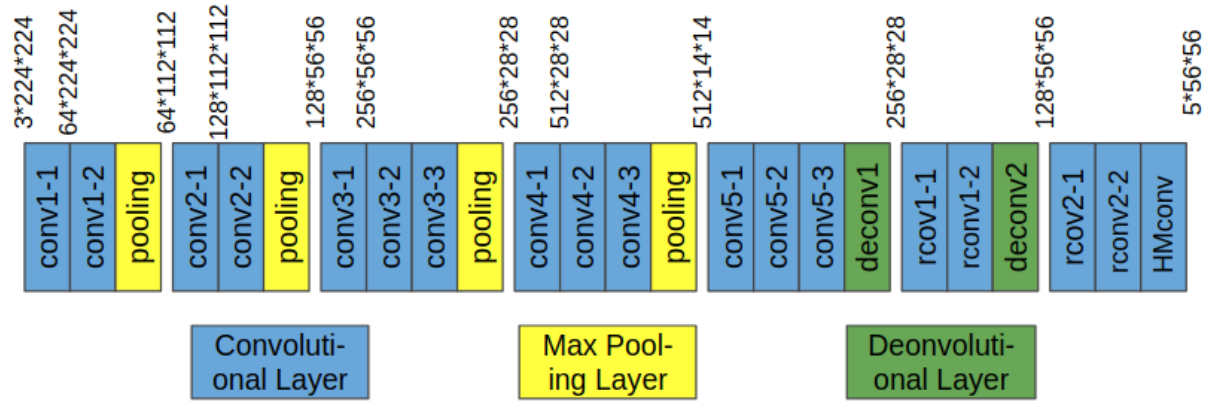



# 2 CNN Detection Model and Experiment

## 2.1 Model Architecture

The model architecture is based on VGG16 network. Instead of the original full connected layers behind the convolution layers, we added deconvolution layers to upsampling and convolution layers to reconstruct heatmaps from the deep features extracted by VGG16 network[3]. The idea refers to the SegNet[2].

The whole architecture is as follows:




After final HMconv layers, I apply a sigmoid function to normalize the map. The loss function is weighted mean absolute error.

## 2.2 Dataset and Preprocessing

The hand data we used in this project is selected from the open hand dataset *11k Hands Dataset*(Mahmoud Afifi, 2019)[1].

Each image of hand is reshaped to 224*224 pixels.

## 2.3 Heatmap Generation

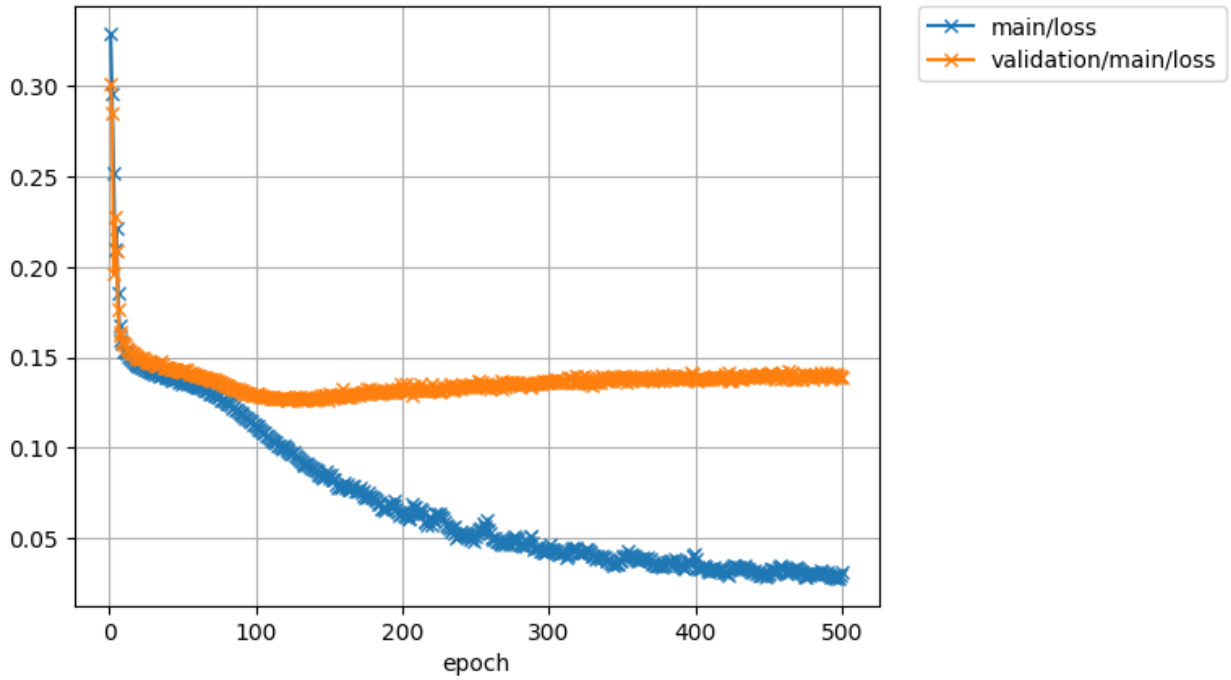
The heatmap from the annotation data was generated in 3 steps:

- Generate binary map where the pixels of key points are set to 1 while others are set to 0;
- Apply a 2D Guassian filter whose kernel size is 3 to the map;
- Apply data normalization.

By applying Guassian filting, we turn the exact coordinate of key points to a possibility distribution of severate coordinates, which avoids the model from falling into local minimum.

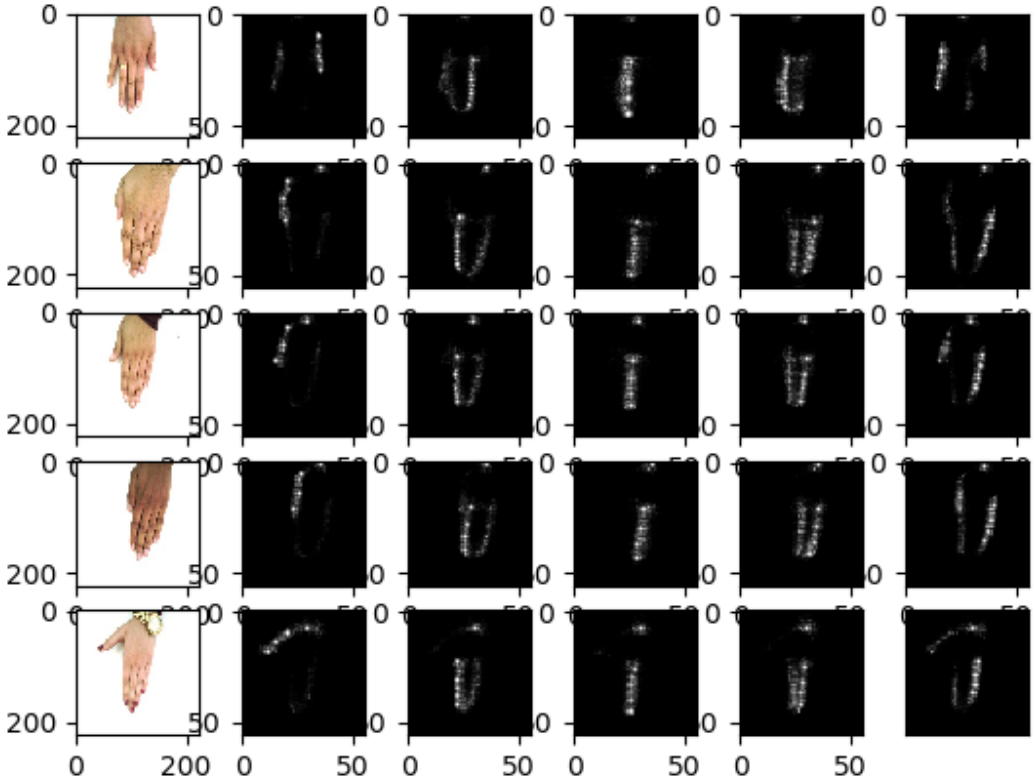
## 2.4 Training Result

Limited by the size of data I annotated by myself, The final performance of my model didn't work quite well on validation set. Therefore, in order to get some results worthy of discussing and show the result of keypoint extraction in next subsection, I made the model a little overfitting on the training set. The loss plot is shown in the following figure.




5 selected prediction results is shown in the following figure. The first column is the original hand images, and the second to sixth columns are predicted heat maps for thumb, point finger, middle finger, ring finger and little finger.

From the figure, we can observe that the network works well on hand keypoint detection overall. It can extract finger features from the background and other part of the hand effectively.




However, there also exits some problems. First, the network works not quite well on comparing between two positional symmetric fingers, e.g. point finger and ring finger. We can see that in almost all the heatmaps of ring finger, the parts of point finger are also highlighted, which are only a little darker than the ring finger's parts. I think this is because we didn't design a structure in the network that encourages the network to generate mutually exclusive highlighted parts for five heatmaps.

Second, even though the network detects the area of fingers successfully, there is still some difficulties for it to detect the specific positions of knuckles. I consider that the reason is the lack of data and that the network cannot learn enough information of relative positions between knuckles and textures.

I think the network may work better by solving the above problems.

## 2.5 Keypoints Prediction

After generating heatmaps, the final keypoints coordinates are detected in two steps:

- Apply a 2D Guassian filter whose kernel size is 3 to the maps to eliminate noises;
- Find specified number of maximums in the filtered heatmaps, e.g. 4 for thumb and 5 for the others.

# 3 Conclusion

In this project, I implemented an annotation tool used to generate hand keypoint labels. After generating a small dataset by myself, I designed a basic CNN network to detect keypoints of hands by training it on the dataset. The implementation of network used Chainer. During this process, I handled basic knowledge and skills of designing networks based on Chainer.

I may continue to improve my work after submitting this project.

# References


[1] Mahmoud Afifi. 11k hands: gender recognition and biometric identification using a large dataset of hand images. *Multimedia Tools and Applications*, 2019.

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[3] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.