Learning Hand Articulations by Hallucinating Heat Distribution - Supplementary Material

Chiho Choi Sangpil Kim Karthik Ramani
Purdue University
West Lafayette, IN 47907, USA
{chihochoi, kim2030, ramani}@purdue.edu

This document serves as supplementary materials to our paper *Learning Hand Articulations by Hallucination Heat Distribution*. We present details of our localization network and the system specifications.

1. Architecture of the Localization Network

We visualize the graph of our localization network in Figure 1. The proposed network solves two sub-tasks for hand localization: hand segmentation and hand center regression. The segmentation stream identifies pixel-wise class labels through a series of the convolution process, so the resulting probability map can effectively reconstruct the detailed hand segment. Also, the regression stream robustly estimates the centroid of the hand and thus enable us to draw the bounding box around the segmented hand.

2. Implementation Specifications

The proposed system was trained with GPUs using the Caffe framework [1]. We trained the localization network by setting the learning rate to 0.0005 and the number of epochs to 250 using the Adam optimizer[1] with $\beta_1 = 0.9$ and $\beta_2 = 0.99$. In addition, our depth network was converged after 80 epochs with the learning rate 0.01 for 60 epochs and 0.001 afterward. Our heat distribution network converged after 250 epochs. Here, we used the learning rate 0.01 for 200 epochs and then dropped it by a factor of 0.1. The hallucination network converged after 100 epochs with the learning rate 0.01. Lastly, the refinement network converged after 170 epochs by dropping the initial learning rate (0.01) in every 60 epochs by a factor of 0.1. At runtime, the computation time for each frame is split as 2 ms for processing data (*i.e.*, depth normalization, resizing, and bounding box cropping), 0.7 ms for hand localization, 1.8 ms to estimate the joint angle parameters from the proposed system. The additional hardware specifications are as follows: Intel's Core i5-4690K, 32GBs RAM, NVIDIA's Geforce GTX 1070, and Intel's RealSense SR300.

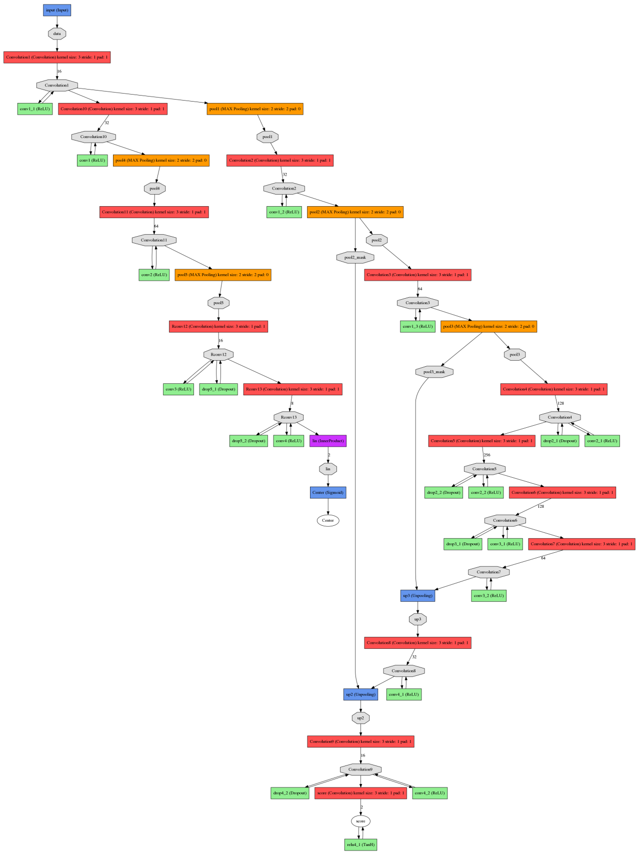
input (Input)
data
Convolution1 (Convolution) kernel size: 3 stride: 1 pad: 1
16
Convolution1
conv1_1 (ReLU)
Convolution10 (Convolution) kernel size: 3 stride: 1 pad: 1
pool1 (MAX Pooling) kernel size: 2 stride: 2 pad: 0
32
Convolution10
pool1
conv1 (ReLU)
pool4 (MAX Pooling) kernel size: 2 stride: 2 pad: 0
Convolution2 (Convolution) kernel size: 3 stride: 1 pad: 1
32
pool4
Convolution2
Convolution11 (Convolution) kernel size: 3 stride: 1 pad: 1
conv1_2 (ReLU)
pool2 (MAX Pooling) kernel size: 2 stride: 2 pad: 0
64
Convolution11
pool2
pool2_mask
conv2 (ReLU)
pool5 (MAX Pooling) kernel size: 2 stride: 2 pad: 0
Convolution3 (Convolution) kernel size: 3 stride: 1 pad: 1
64
pool5
Convolution3
conv1_3 (ReLU)
pool3 (MAX Pooling) kernel size: 2 stride: 2 pad: 0
Rconv12 (Convolution) kernel size: 3 stride: 1 pad: 1
16
Rconv12
pool3
pool3_mask
conv3 (ReLU)
drop5_1 (Dropout)
Rconv13 (Convolution) kernel size: 3 stride: 1 pad: 1
Convolution4 (Convolution) kernel size: 3 stride: 1 pad: 1
8
128
Rconv13
Convolution4
drop5_2 (Dropout)
conv4 (ReLU)
lin (InnerProduct)
Convolution5 (Convolution) kernel size: 3 stride: 1 pad: 1
drop2_1 (Dropout)
conv2_1 (ReLU)
2
256
lin
Convolution5
Center (Sigmoid)
drop2_2 (Dropout)
conv2_2 (ReLU)
Convolution6 (Convolution) kernel size: 3 stride: 1 pad: 1
Center
128
Convolution6
drop3_1 (Dropout)
conv3_1 (ReLU)
Convolution7 (Convolution) kernel size: 3 stride: 1 pad: 1
64
Convolution7
up3 (Unpooling)
conv3_2 (ReLU)
up3
Convolution8 (Convolution) kernel size: 3 stride: 1 pad: 1
32
Convolution8
up2 (Unpooling)
conv4_1 (ReLU)
up2
Convolution9 (Convolution) kernel size: 3 stride: 1 pad: 1
16
Convolution9
drop4_2 (Dropout)
score (Convolution) kernel size: 3 stride: 1 pad: 1
conv4_2 (ReLU)
2
score
relu4_1 (TanH)

Figure 1: The graph of the proposed localization network architecture. There are two streams: the segmentation stream for pixel-wise hand segmentation, and the regression stream for hand center regression.

References

[1] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014. 1

[1] The rest of the networks used the SGD optimizer.