Hand Segmentation Using RefineNet (In Python)

Research Papers-

1. Analysis of Hand Segmentation in the Wild (CVPR 2018)
2. RefineNet: Multipath Refinement Networks for High-Resolution Semantic Segmentation (CVPR 2017)

Github: https://github.com/adarsh1001/Hand_Segmentation_RefineNet

Adarsh Pal Singh
Ishan Bansal
Paawan Gupta

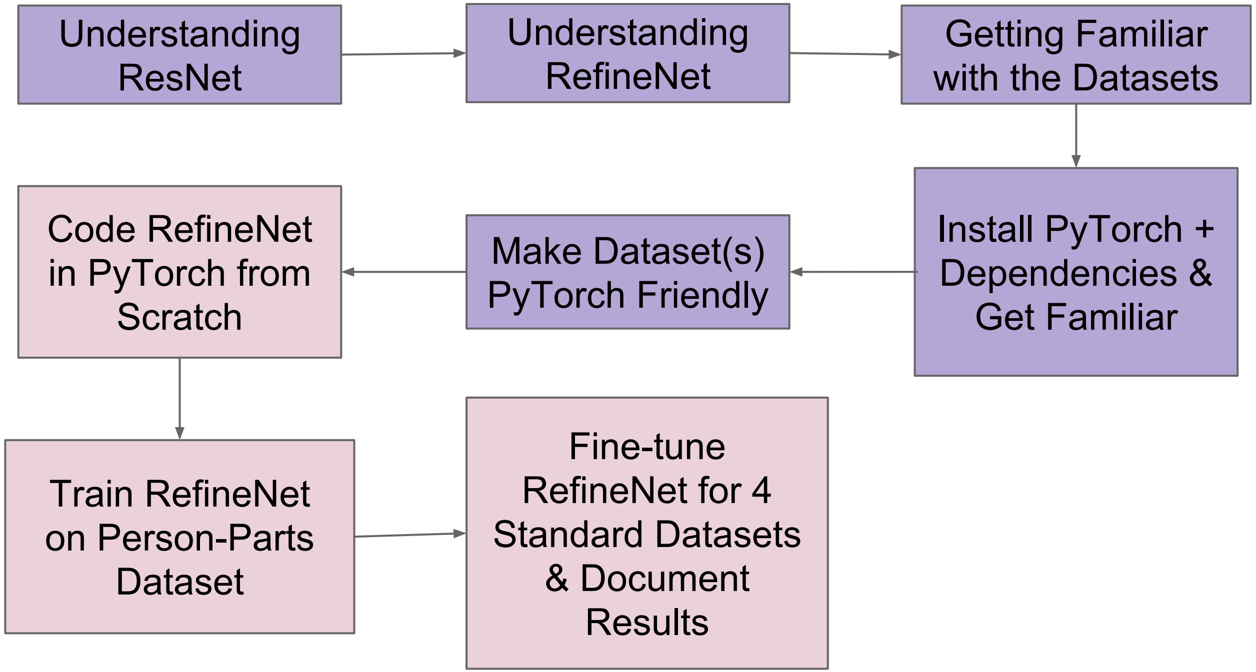
In a Nutshell…

The main goal of this project is to develop an egocentric hand segmentation model using RefineNet in Python.

Motivation for Hand Segmentation

1. Hand pose and configuration tell a lot about what we plan to do or what we pay attention to.
2. Applications in robotics, human-machine interaction, computer vision, augmented reality, etc.
3. Extracting hand regions in egocentric videos is a critical step for understanding fine motor skills such as hand-object manipulation and hand-eye coordination.

Plan

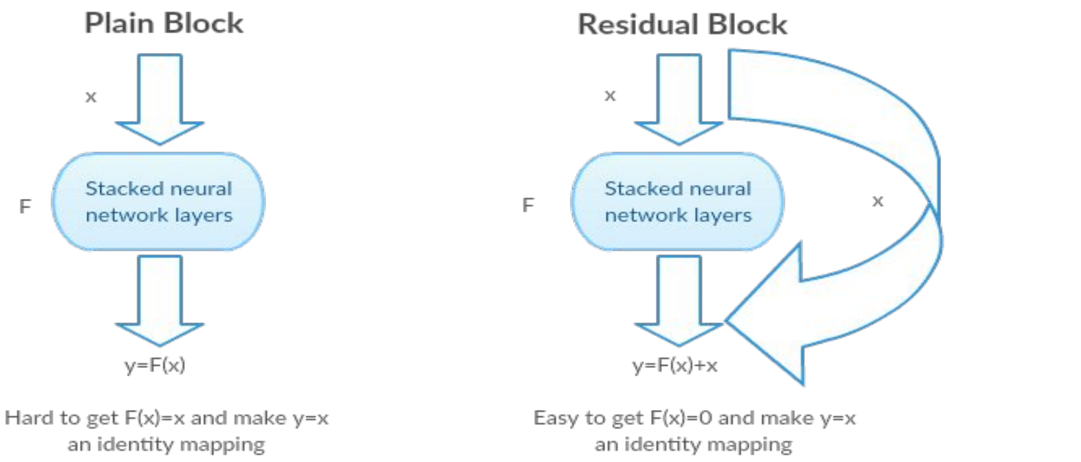

Understanding ResNet
Understanding RefineNet
Getting Familiar with the Datasets
Install PyTorch + Dependencies & Get Familiar
Make Dataset(s) PyTorch Friendly
Code RefineNet in PyTorch from Scratch
Train RefineNet on Person-Parts Dataset
Fine-tune RefineNet for 4 Standard Datasets & Document Results

ResNet

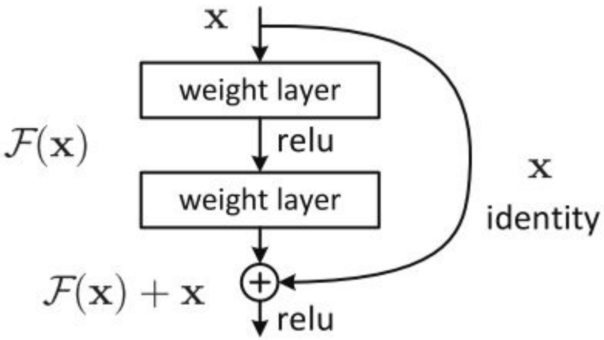
Understanding ResNet

- Feedforward network with a single layer is sufficient to represent any function.
- However, the layer might be massive and the network is prone to overfitting the data.
- Common trend in research to make networks deeper!
- However, increasing network depth does not work by simply stacking layers together.
- Deep networks are hard to train because of the notorious vanishing gradient problem.

- Another Problem: Performing optimization on huge parameter space and naively adding layers leads to higher training error (Degradation Problem).
- Residual networks allow training of such deep networks by constructing the network through modules called residual model.



- The core idea of ResNet is introducing “identity shortcut connection” that skips one or more layers
- These parameterized gates control how much information is allowed to flow across the shortcut



RefineNet

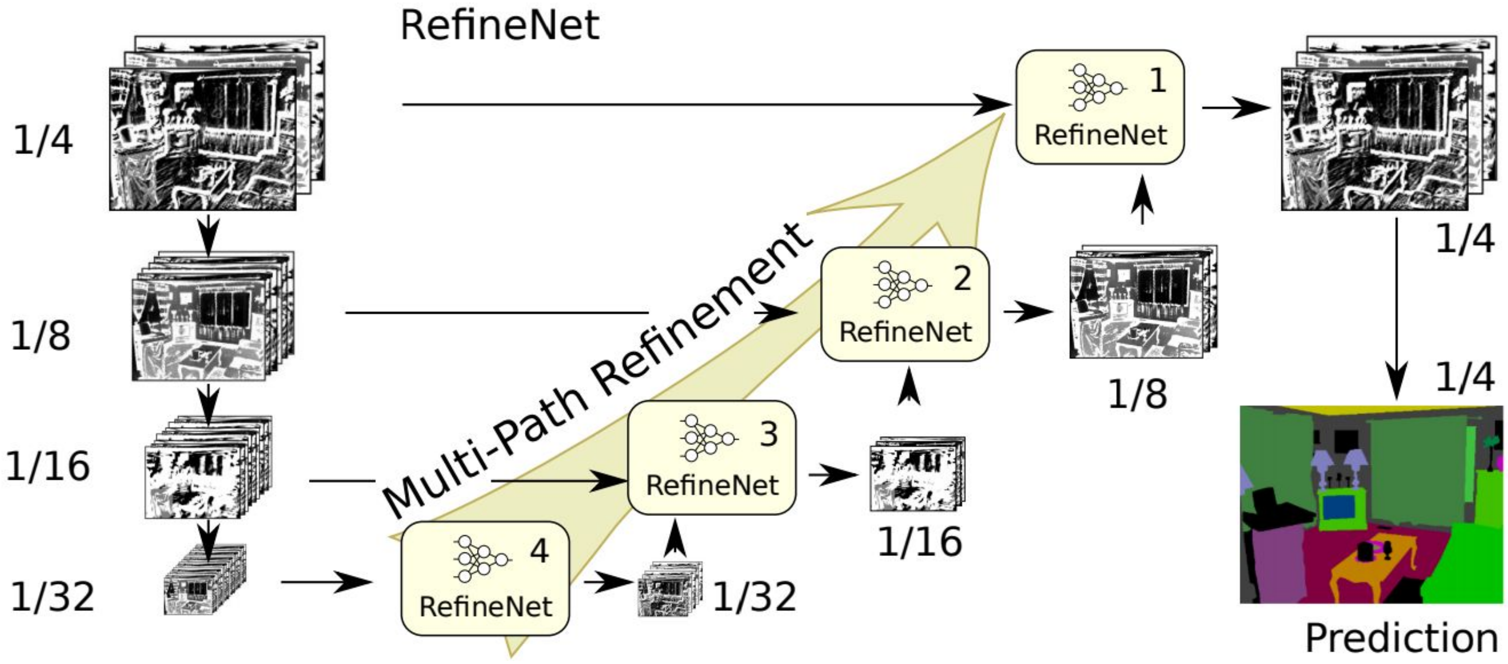
Why RefineNet?

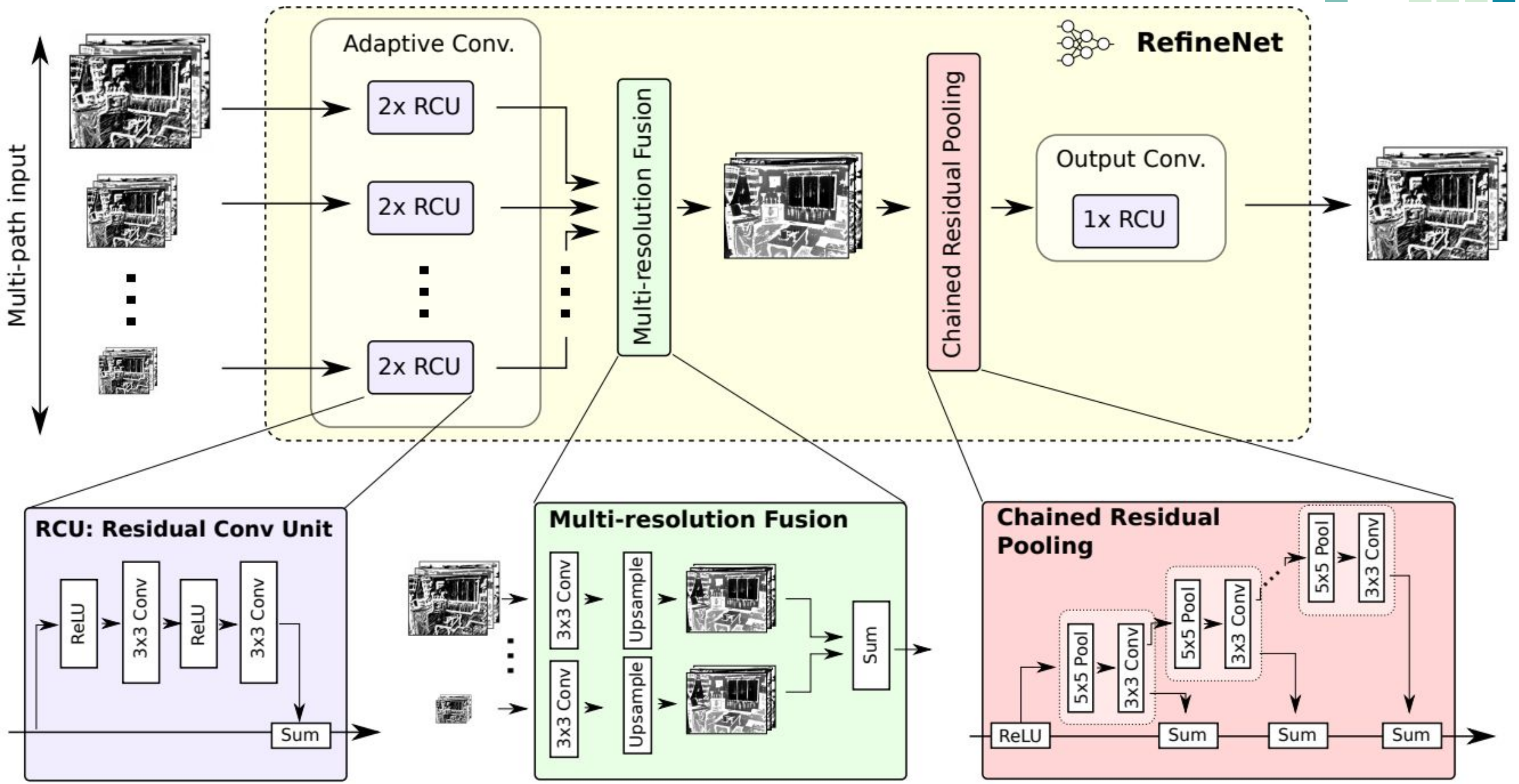
- RefineNet is a multi-path refinement network which exploits all the features at multiple levels along the down sampling path
- Authors performed off-the-shelf evaluation of leading semantic segmentation methods on the EgoHands dataset and found that RefineNet gives better results than other models.
- On EgoHands dataset, RefineNet significantly outperformed the baseline.

Understanding RefineNet

- Dilated convolutions are computationally expensive and take a lot of memory because they have to be applied on large number of high resolution feature maps.
- This hampers the computation of high-res predictions.
- RefineNet uses encoder-decoder architecture.
- Encoder part is ResNet-101 blocks.
- Decoder has RefineNet blocks which concatenate/fuse high res features from encoder and low res features from previous RefineNet block.

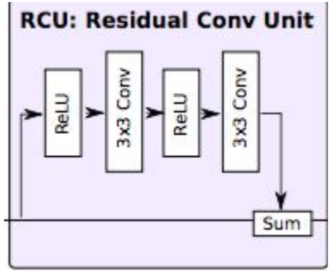
- RefineNet provides a generic means to fuse coarse high-level semantic features with finer-grained low-level features to generate high-resolution semantic feature maps
- It ensures that the gradient can be effortlessly propagated backwards through the network all the way to early low-level layers over long range residual connections, ensuring that the entire network can be trained end-to-end


RefineNet
1/4
1/8
1/16
1/32
Multi-Path Refinement
1
RefineNet
2
RefineNet
3
RefineNet
4
RefineNet
1/32
1/16
1/8
1/4
1/4
Prediction


Multi-path input
Adaptive Conv.
2x RCU
2x RCU
2x RCU
Multi-resolution Fusion
Chained Residual Pooling
RefineNet
Output Conv.
1x RCU
RCU: Residual Conv Unit
ReLU
3x3 Conv
ReLU
3x3 Conv
Sum
Multi-resolution Fusion
3x3 Conv
Upsample
3x3 Conv
Upsample
Sum
Chained Residual Pooling
ReLU
5x5 Pool
3x3 Conv
5x5 Pool
3x3 Conv
5x5 Pool
3x3 Conv
Sum
Sum
Sum

Residual Convolution Unit (RCU)

- Adaptive Convolution set that fine tunes the pretrained ResNet weights for the task.
- Each Input is passed sequentially through 2 RCU where Batch Normalization is removed from the original ResNet.



Multi-Resolution Fusion

- All path inputs are then fused into a high-resolution feature map by the multi-resolution fusion block.
- First applies convolutions for input adaptation, which generates feature maps of the same feature dimension.
- Up-samples all feature maps to the largest resolution of the inputs.
- Finally, all features maps are fused by summation.
- The input adaptation in this block also helps to re-scale the feature values appropriately along different paths.

Chain Residual Pooling

- Aims to capture background context from a large image region.
- It is able to efficiently pool features with multiple window sizes and fuse them together using learnable weights.
- In particular, this component is built as a chain of multiple pooling blocks, each consisting of one max-pooling layer and one convolution layer.
- The current pooling block is able to re-use the result from the previous pooling operation and thus access the features from a large region without using a large pooling window.

Output Convolutions

- The final step of each RefineNet block is another residual convolution unit (RCU).
- This results in a sequence of three RCUs between each block. To reflect this behavior in the last RefineNet-1 block, we place two additional RCUs before the final softmax prediction step.
- The goal here is to employ non-linearity operations on the multi-path fused feature maps to generate features for further processing or for final prediction.
- The feature dimension remains the same after going through this block.

How is RefineNet used?

- RefineNet-Res101 pre-trained on Pascal Person-Part dataset used in all experiments.
- A new classification layer added with 2 classes: hand and no hand.
- Fine-tuned the model on EgoHands, EYTH, GTEA, and HOF datasets.
- RefineNet-Res101 uses feature maps from ResNet101.
- After fine tuning, performed multi-scale evaluation for scales: [0.6, 0.8, 1.0] which gives consistently better results than single scale evaluation.

Datasets

PASCAL Person-Parts Dataset

- A subset of the Parts-dataset that is present with VOC 2010 dataset.
- 24 different human body parts annotated!
- Mostly third person photos.
- Link: http://www.stat.ucla.edu/~xianjie.chen/pascal_part_dataset/pascal_part.html

EgoHands

- 48 videos recorded with Google glass.
- Videos are recorded in 3 different environments: office, courtyard and living room.
- Each video has two actors doing one of the 4 activities: playing puzzle, cards, jenga or chess.
- Pixel-level ground truth for over 15000 hand instances.
- Link: http://vision.soic.indiana.edu/projects/egohands/

EgoYouTubeHands (EYTH)

- Pixel-level hand annotations in real world images and/or videos obtained from YouTube.
- Users perform different activities and are interacting with others.
- This dataset has 2600 hand instances, with approx. 1800 first-person hand instances and approx. 800 third-person hands.
- Link: https://github.com/aurooj/Hand-Segmentation-in-the-Wild

Georgia Tech Egocentric Activity (GTEA)

- 7 daily activities performed by 4 subjects.
- Videos are collected in the same environment for the purpose of activity recognition.
- Does not capture social interactions and is collected under static illumination conditions annotated at 15 fps for 61 action classes.
- 663 images with pixel-level hand annotations.
- Link: http://www.cbi.gatech.edu/fpv/

HandOverFace (HOF)

- Contains 300 images obtained from the web in which faces are occluded by hands.
- Useful to study how skin similarity can affect hand segmentation.
- Has images for people from different ethnicities, age, and gender.
- Pixel-level annotations for hands along with the hand type: left or right.
- Link: https://github.com/aurooj/Hand-Segmentation-in-the-Wild

(a) EgoHands (b) EgoYouTubeHands (c) GTEA (d) HandOverFace

Figure 1: Sample images from 4 hand segmentation datasets including EgoHands, EYTH, GTEA and HOF, used in this paper.

Results

Initial Testing with RefineNet

An example code written to test and get familiar with RefineNet in PyTorch. Pre-trained VOC weight file directly used.

Original Image

Res101

Original Image

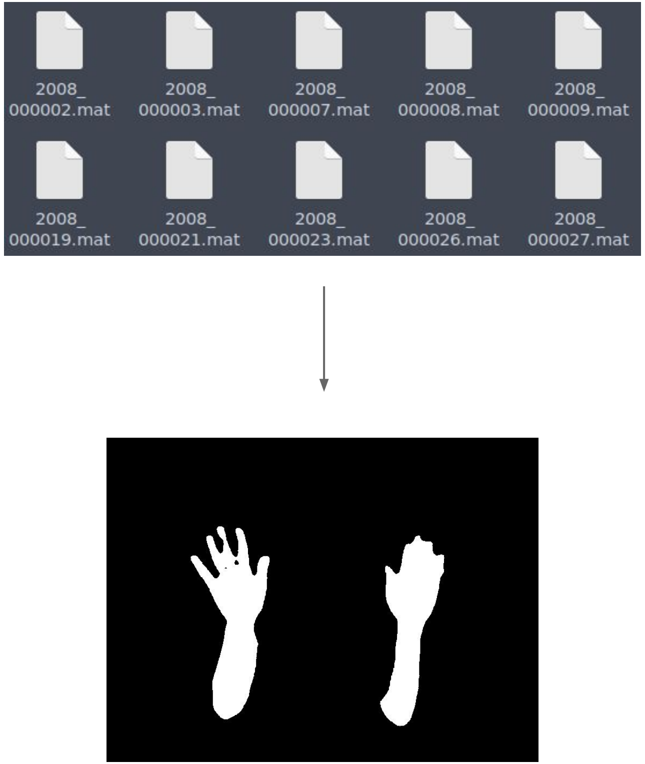
Res101

Codes

Problem: Pre-trained weight for VOC-parts dataset incompatible with PyTorch! Moreover, the parts dataset has .mat files for image labels which PyTorch's RefineNet can't use natively.

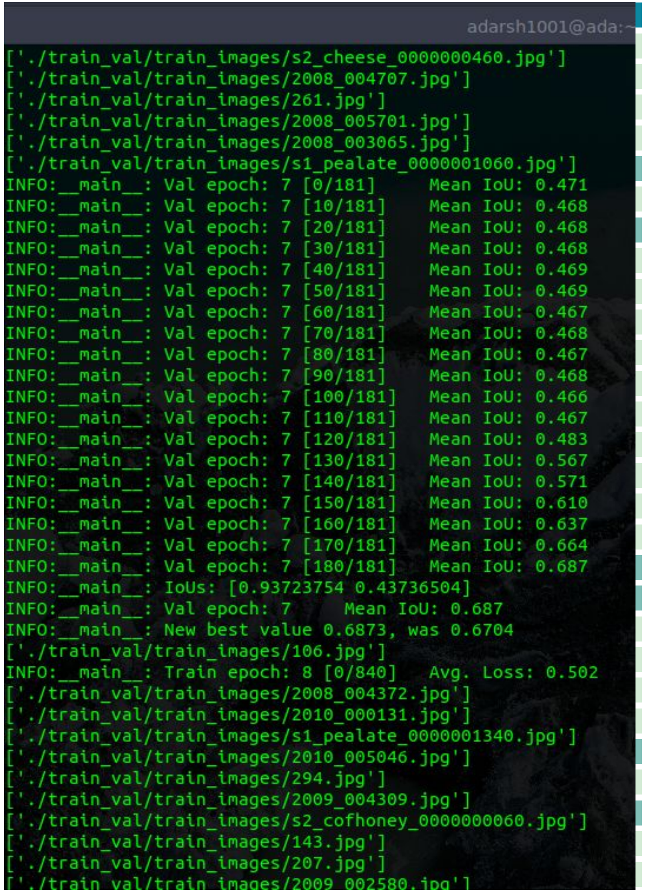
Dataset Cleaning: Script to process the Parts dataset. Converts .mat files to .jpg segmentations for all pictures containing "persons" => Person-parts dataset!

Codes


2008_000002.mat
2008_000003.mat
2008_000007.mat
2008_000008.mat
2008_000009.mat
2008_000019.mat
2008_000021.mat
2008_000023.mat
2008_000026.mat
2008_000027.mat

Training

1. Train on PersonParts (for Hands).
2. Fine tune for other datasets.
3. Learning 5e-5
4. Scales: [0.6, 0.8, 1.0]



mIoU

PersonParts (Hand): 0.61

EgoHands: 0.662

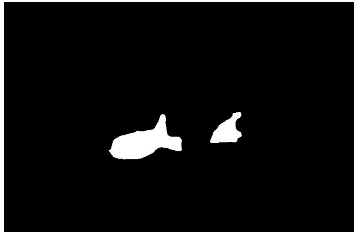
EYTH: 0.492

GTEA: 0.637

HOF: 0.612

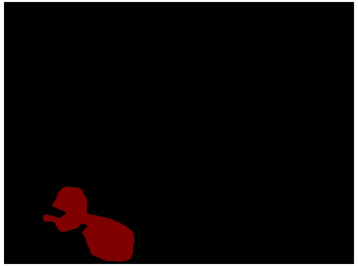
** On their respective train-test splits.





Person Parts



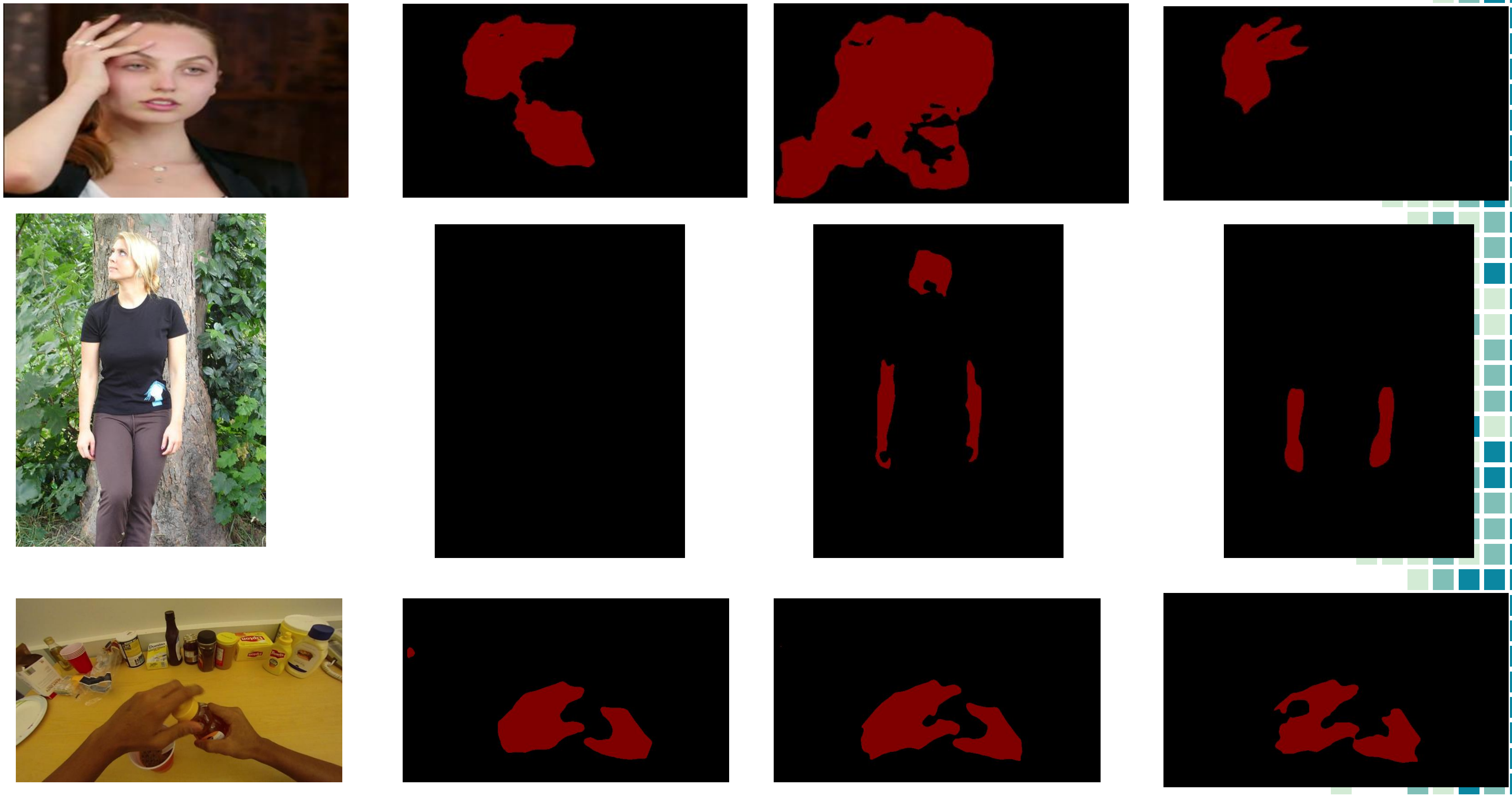


EgoHands

EYTH GTEA HOF

Person Parts

EgoHands



EYTH | GTEA | HOF

Conclusion

- We got different results for fined tuned models on different datasets.
- Best result was from PersonParts & EgoHands based model
- HoF based model is useful to study similar appearance occlusions like hand-to-skin occlusions
- Cross-dataset testing revealed segmentation faults with other body parts like in the case of GTEA.

Failure Cases

- Motion Blur

- Occlusion
- Similar Appearance Occlusion
- Small Hands
- Lightning Conditions

THANKS!

Any questions?