# Nitte Meenakshi Institute of Technology

## Department of Computer Science and Engineering

## 18CSE751 Introduction to Machine Learning

## Learning Activity Proposal

Guide : Dr. Vani V

## AI Stylist

Khushi Dubey - 1NT18CS075
Mohammad Shadaab - 1NT18CS099
Simran Maurya - 1NT18CS160

## Abstract

The idea is to utilize machine learning models for image segmentation and classifying a picture of clothing, according to a set of predetermined features (Example: tops, trouser, skirt, etc). We plan on building a ML model for automatic categorization of fashion apparels. In short ,we propose fashion apparel detection and feature tagging.

## Introduction

In Machine Learning (ML) and AI – Computer vision is used to train the model to recognize certain patterns and store the data into their artificial memory to utilize the same for predicting the results in real-life use[2].

Methods that we will be using to accomplish this task are -

1. Semantic Segmentation : The task of assigning a class to every pixel in a given image. Note here that this is significantly different from classification. Classification assigns a single class to the whole image whereas semantic segmentation classifies every pixel of the image to one of the classes.
   Why Semantic Segmentation?
   To detect the human body or the cloth in the input image.

2. Background Segmentation : Background separation is a segmentation task, where the goal is to split the image into foreground and background. In semi-interactive settings,

the user marks some pixels as “foreground”, a few others as “background”, and it’s up to the algorithm to classify the rest of the pixels.
Why Background Segmentation?
To remove the background noise which will disrupt the accuracy of the image classification model.

3. Image Classification : Image classification is where a computer can analyse an image and identify the 'class' the image falls under. (Or a probability of the image being part of a 'class'.) A class is essentially a label, for instance, 'car', 'animal', 'building' and so on.
Why Image Classification?
This method is the final step to predict the labels of the input image.

## Dataset

We are using the DeepFashion2[4] dataset, it has labels and annotations much larger than any other dataset. DeepFashion2 is a comprehensive fashion dataset. It contains 491K diverse images of 13 popular clothing categories from both commercial shopping stores and consumers.

| AP | AP50 | AP75 |
|---|---|---|
| 0.638 | 0.789 | 0.745 |

Clothes detection trained with released DeepFashion2 Dataset evaluated on validation set.

**1**. **Attributes**
It totally has 801K clothing items, where each image is attributed with : scale, occlusion, zoom-in, viewpoint, category, style, bounding box, dense landmarks and per-pixel mask.

Annotations are in this format -

- source: a string, where 'shop' indicates that the image is from a commercial store while 'user' indicates that the image is taken by users.
- pair_id: a number. Images from the same shop and their corresponding consumer-taken images have the same pair id.
- item 1
  - category_name: a string which indicates the category of the item.
  - category_id: a number which corresponds to the category name. In category_id, 1 represents short sleeve top, 2 represents long sleeve top, 3 represents short sleeve outwear, 4 represents long sleeve outwear, 5 represents vest, 6 represents sling, 7 represents shorts, 8 represents trousers, 9 represents skirt, 10 represents short sleeve dress, 11 represents long sleeve dress, 12 represents vest dress and 13 represents sling dress.
  - style: a number to distinguish between clothing items from images with the same pair id.
  - bounding_box: [x1,y1,x2,y2], where x1 and y_1 represent the upper left point coordinate of the bounding box, x_2 and y_2 represent the lower right point coordinate of the bounding box. (width=x2-x1;height=y2-y1)

- landmarks: [x1,y1,v1,...,xn,yn,vn], where v represents the visibility: v=2 visible; v=1 occlusion; v=0 not labeled. We have different definitions of landmarks for different categories. The orders of landmark annotations are listed in figure 2.
- segmentation: [[x1,y1,...xn,yn],[ ]], where [x1,y1,xn,yn] represents a polygon and a single clothing item may contain more than one polygon.
- scale: a number, where 1 represents small scale, 2 represents modest scale and 3 represents large scale.
- occlusion: a number, where 1 represents slight occlusion(including no occlusion), 2 represents medium occlusion and 3 represents heavy occlusion.
- zoom_in: a number, where 1 represents no zoom-in, 2 represents medium zoom-in and 3 represents large zoom-in.
- viewpoint: a number, where 1 represents no wear, 2 represents frontal viewpoint and 3 represents side or back viewpoint.

* item 2
  ...
* item n

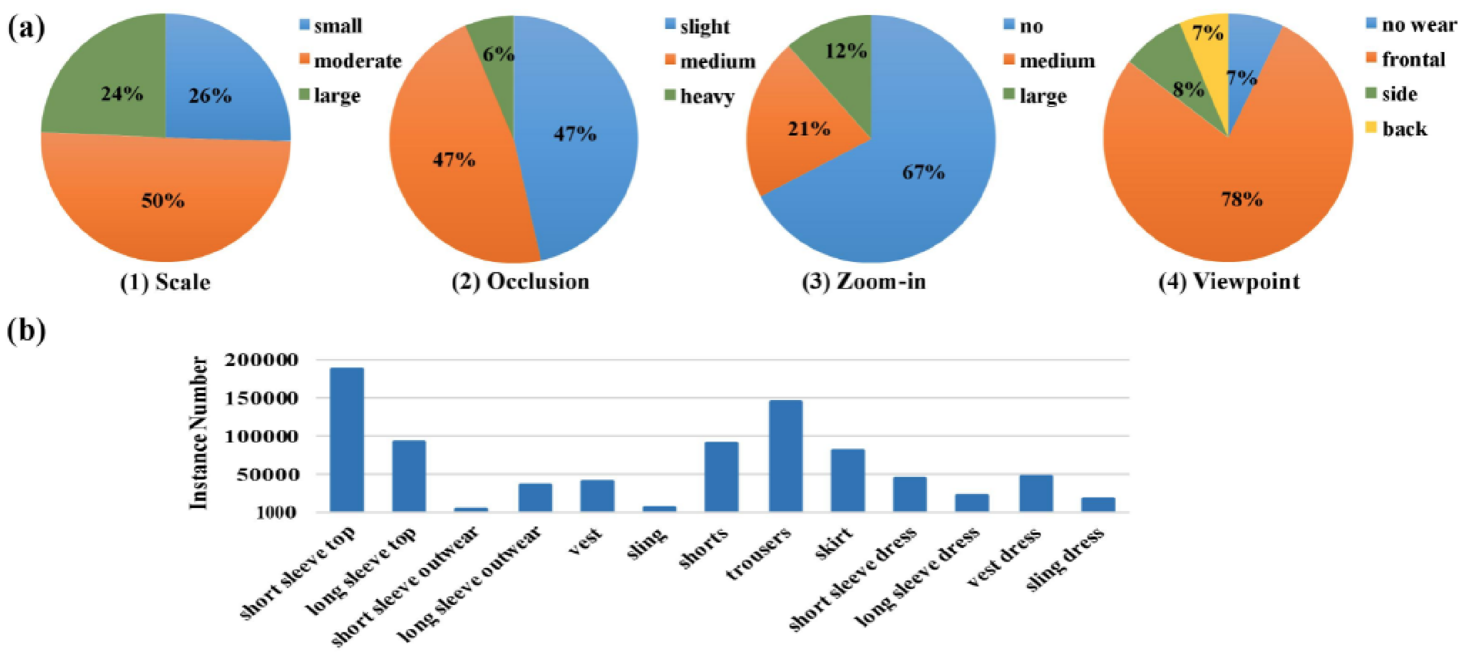
**2. Dataset Organization**
The dataset is split into a training set (391K images), a validation set (34k images), and a test set (67k images).

Training images: train/image Training annotations: train/annos
Validation images: validation/image Validation annotations: validation/annos
Test images: test/image

Each image in seperate image set has a unique six-digit number such as 000001.jpg. A corresponding annotation file in json format is provided in an annotation set such as 000001.json.

**3. Dataset Statistics**

| | Train | Validation | Test | Overall |
|---|---|---|---|---|
| images | 390,884 | 33,669 | 67,342 | 491,895 |
| bboxes | 636,624 | 54,910 | 109,198 | 800,732 |
| landmarks | 636,624 | 54,910 | 109,198 | 800,732 |
| masks | 636,624 | 54,910 | 109,198 | 800,732 |
| pairs | 685,584 | query: 12,550<br>gallery: 37183 | query: 24,402<br>gallery: 75,347 | 873,234 |

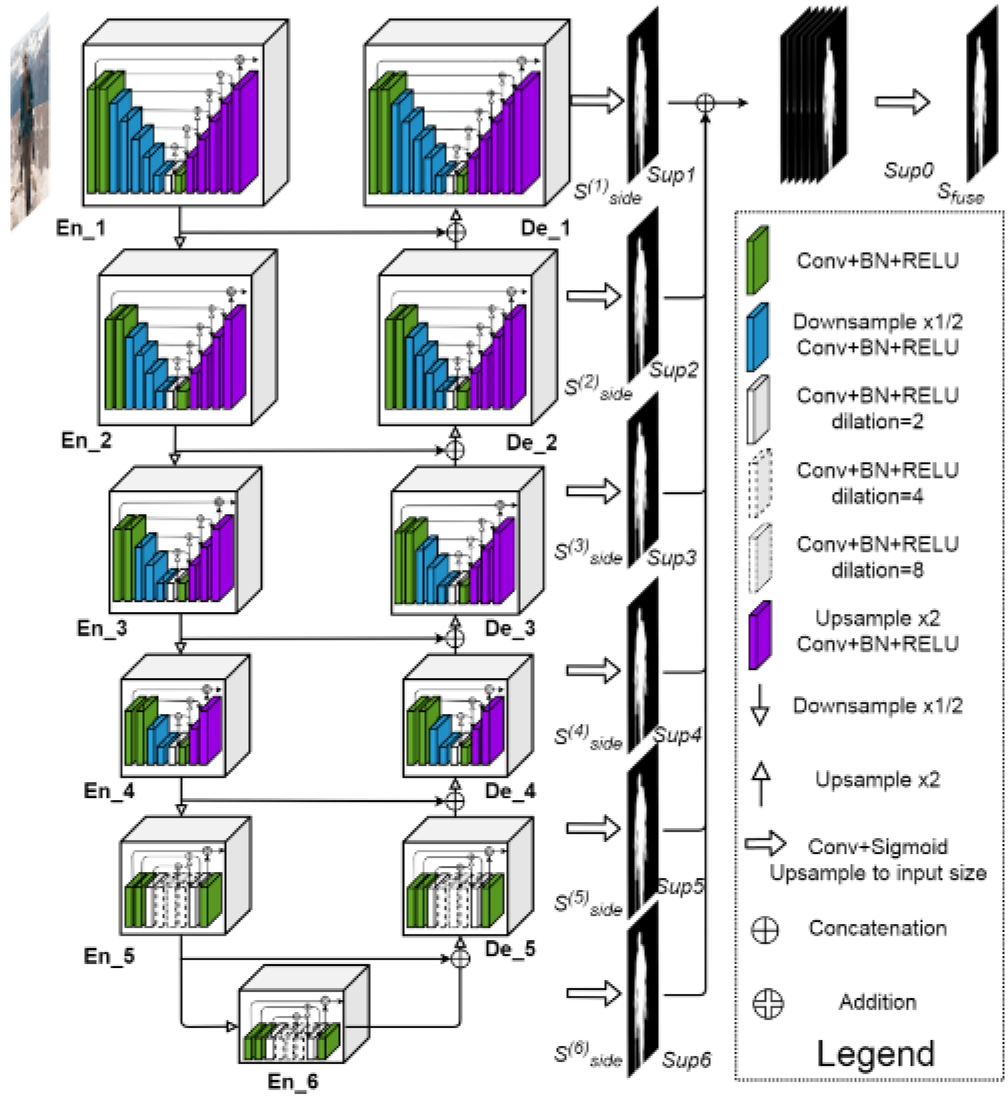



### 4. Dataset Challenges

The only challenge we are facing is that we require a GPU with high memory and cuda cores for faster computation because of the dataset being so vast.

## Deep Learning Methods

1. Semantic Segmentation[1]
   For Semantic Segmentation we are using the U^2NET[5] model. The model has been trained and tested on the DUTS and ECSSD dataset using tf v2.4.1 and keras v2.4.3.
   U^2NET : The architecture of the U2-Net is a two-level nested U-structure. The design has the following advantages: (1) it is able to capture more contextual information from different scales thanks to the mixture of receptive fields of different sizes in our proposed ReSidual U-blocks (RSU), (2) it increases the depth of the whole architecture without significantly increasing the computational cost because of the pooling operations used in these RSU blocks. This architecture enables you to train a deep network from scratch without using backbones from image classification tasks.

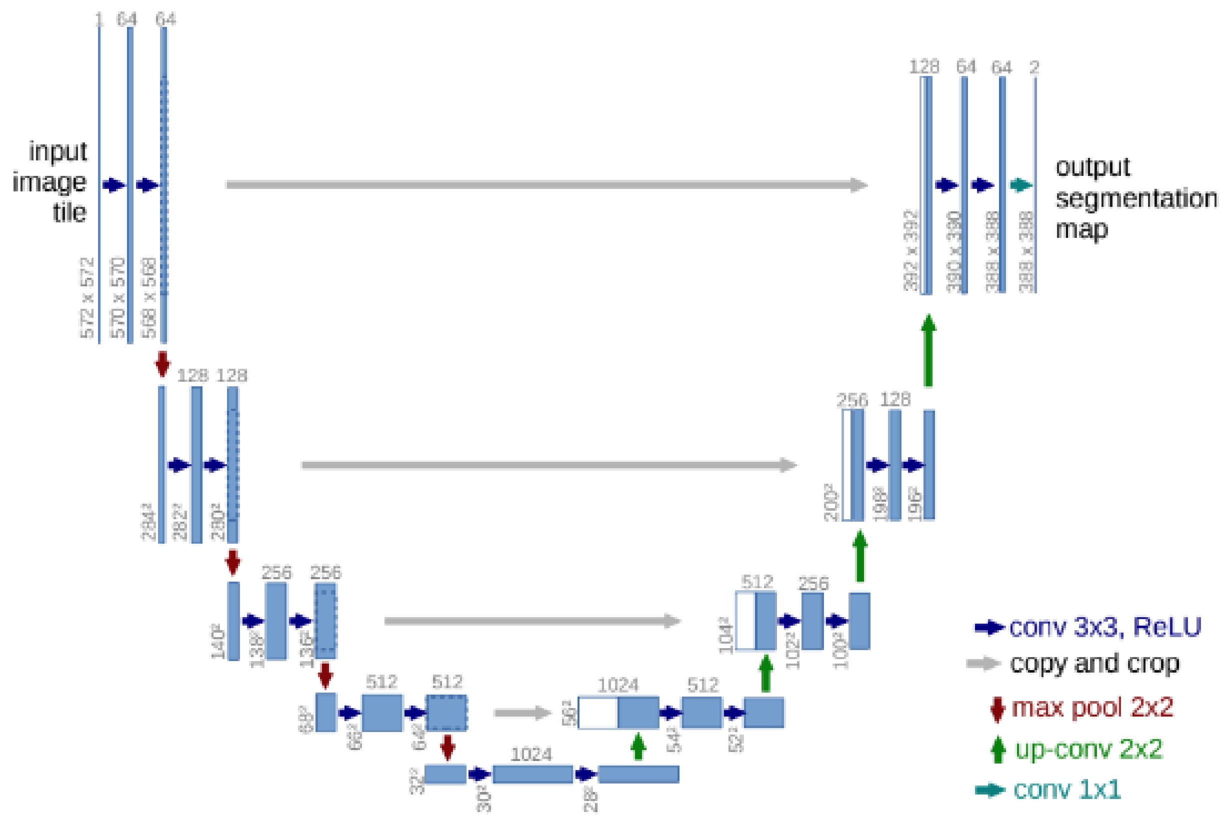



The authors of this paper have provided pre-trained models, and we are using that for semantic segmentation.

2. Background Segmentation
   For background segmentation we take the output of the U2-Net model which is a masked image and subtract that with the original image such that only the masked pixels are left in the image instance.
   To do so we use OpenCV's subtract function "cv2.subtract(image1,image2)".

3. Object Detection
   U-Net[6] is an architecture consisting of a contracting path and an expansive path. The contracting path follows the typical architecture of a convolutional network. It consists of the repeated application of two 3x3 convolutions (unpadded convolutions), each followed by a rectified linear unit (ReLU) and a 2x2 max pooling operation with stride 2 for downsampling. At each downsampling step we double the number of feature channels. Every step in the expansive path consists of an upsampling of the feature map followed by a 2x2 convolution ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the

contracting path, and two 3x3 convolutions, each followed by a ReLU. The cropping is necessary due to the loss of border pixels in every convolution. At the final layer a 1x1 convolution is used to map each 64-component feature vector to the desired number of classes. In total the network has 23 convolutional layers.




We trained the U-Net Model on DeepFashion2 dataset for object detection task for 6 classes for clothes: top, shorts, trousers, skirt, dress and outerwear.

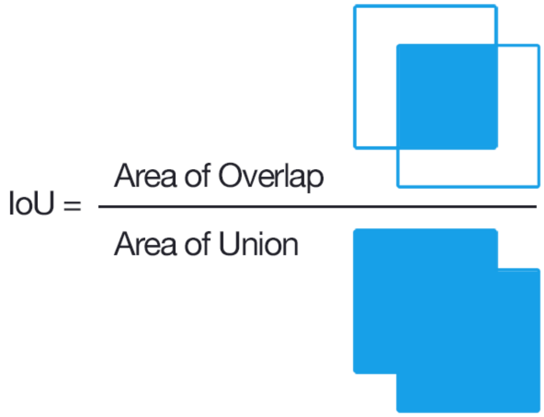
## Assessment

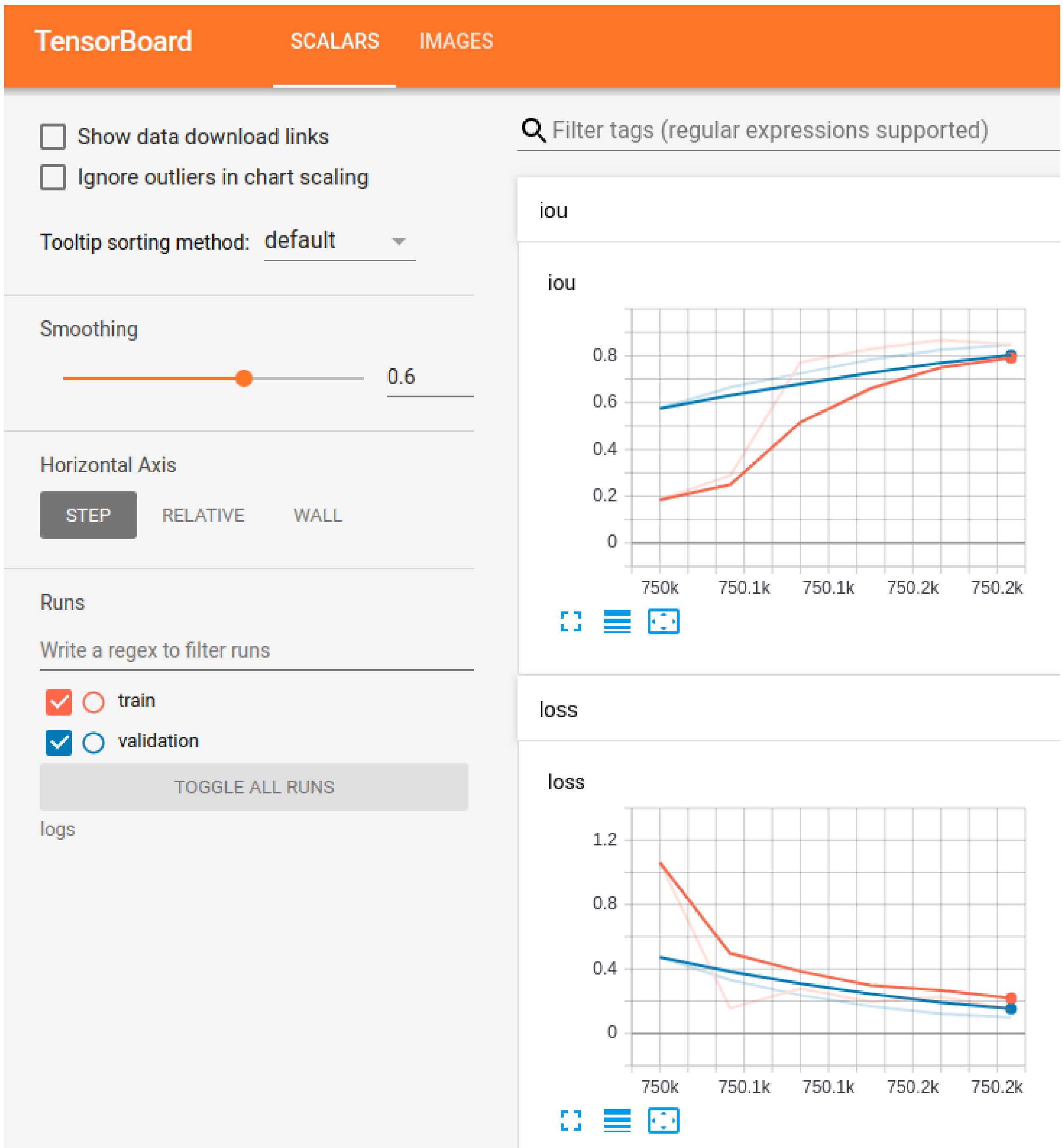
UNET Model:

To assess the unet model we used the IoU method.

IoU - Intersection over Union (IoU) is used when calculating mAP. It is a number from 0 to 1 that specifies the amount of overlap between the predicted and ground truth bounding box.

- an IoU of 0 means that there is no overlap between the boxes
- an IoU of 1 means that the union of the boxes is the same as their overlap indicating that they are completely overlapping


IoU =
Area of Overlap
Area of Union


TensorBoard
SCALARS
IMAGES
Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
train
validation
TOGGLE ALL RUNS
logs
Filter tags (regular expressions supported)
iou
iou
0.8
0.6
0.4
0.2
0
750k
750.1k
750.1k
750.2k
750.2k
loss
loss
1.2
0.8
0.4
0
750k
750.1k
750.1k
750.2k
750.2k

U2-NET Model:

Since we used a pre-trained model provided by the authors of U2-NET, we did not run any kind of assessment method.

Table 3: Comparison of our method and 20 SOTA methods on DUT-OMRON, DUTS-TE, HKU-IS in terms of model size, $maxF_\beta$ (↑), $MAE$ (↓), weighted $F^w_\beta$ (↑), structure measure $S_m$ (↑) and relax boundary F-measure $relaxF^b_\beta$ (↑). Red, Green, and Blue indicate the best, second best and third best performance.

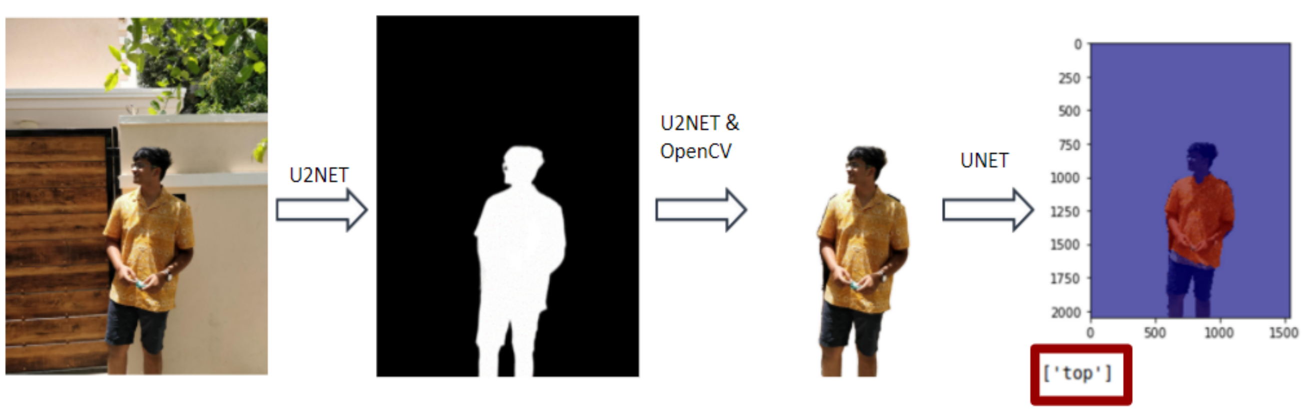
| Method | Backbone | Size(MB) | DUT-OMRON (5168) | | | | | DUTS-TE (5019) | | | | | HKU-IS (4447) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $maxF_\beta$ | $MAE$ | $F^w_\beta$ | $S_m$ | $relaxF^b_\beta$ | $maxF_\beta$ | $MAE$ | $F^w_\beta$ | $S_m$ | $relaxF^b_\beta$ | $maxF_\beta$ | $MAE$ | $F^w_\beta$ | $S_m$ | $relaxF^b_\beta$ |
| **MDF**$_{TIP16}$ | AlexNet | 112.1 | 0.694 | 0.142 | 0.565 | 0.721 | 0.406 | 0.729 | 0.099 | 0.543 | 0.723 | 0.447 | 0.860 | 0.129 | 0.564 | 0.810 | 0.594 |
| **UCF**$_{ICCV17}$ | VGG-16 | 117.9 | 0.730 | 0.120 | 0.573 | 0.760 | 0.480 | 0.773 | 0.112 | 0.596 | 0.777 | 0.518 | 0.888 | 0.062 | 0.779 | 0.875 | 0.679 |
| **Amulet**$_{ICCV17}$ | VGG-16 | 132.6 | 0.743 | 0.098 | 0.626 | 0.781 | 0.528 | 0.778 | 0.084 | 0.658 | 0.796 | 0.568 | 0.897 | 0.051 | 0.817 | 0.886 | 0.716 |
| **NLDF+**$_{CVPR17}$ | VGG-16 | 428.0 | 0.753 | 0.080 | 0.634 | 0.770 | 0.514 | 0.813 | 0.065 | 0.710 | 0.805 | 0.591 | 0.902 | 0.048 | 0.838 | 0.879 | 0.694 |
| **DSS+**$_{CVPR17}$ | VGG-16 | 237.0 | 0.781 | 0.063 | 0.697 | 0.790 | 0.559 | 0.825 | 0.056 | 0.755 | 0.812 | 0.606 | 0.916 | 0.040 | 0.867 | 0.878 | 0.706 |
| **RAS**$_{ECCV18}$ | VGG-16 | 81.0 | 0.786 | 0.062 | 0.695 | 0.814 | 0.615 | 0.831 | 0.059 | 0.740 | 0.828 | 0.656 | 0.913 | 0.045 | 0.843 | 0.887 | 0.748 |
| **PAGRN**$_{CVPR18}$ | VGG-19 | - | 0.771 | 0.071 | 0.622 | 0.775 | 0.582 | 0.854 | 0.055 | 0.724 | 0.825 | 0.692 | 0.918 | 0.048 | 0.820 | 0.887 | 0.762 |
| **BMPM**$_{CVPR18}$ | VGG-16 | - | 0.774 | 0.064 | 0.681 | 0.809 | 0.612 | 0.852 | 0.048 | 0.761 | 0.851 | 0.699 | 0.921 | 0.039 | 0.859 | 0.907 | 0.773 |
| **PiCANet**$_{CVPR18}$ | VGG-16 | 153.3 | 0.794 | 0.068 | 0.691 | 0.826 | 0.643 | 0.851 | 0.054 | 0.747 | 0.851 | 0.704 | 0.921 | 0.042 | 0.847 | 0.906 | 0.784 |
| **MLMS**$_{CVPR19}$ | VGG-16 | 263.0 | 0.774 | 0.064 | 0.681 | 0.809 | 0.612 | 0.852 | 0.048 | 0.761 | 0.851 | 0.699 | 0.921 | 0.039 | 0.859 | 0.907 | 0.773 |
| **AFNet**$_{CVPR19}$ | VGG-16 | 143.0 | 0.797 | 0.057 | 0.717 | 0.826 | 0.635 | 0.862 | 0.046 | 0.785 | 0.855 | 0.714 | 0.923 | 0.036 | 0.869 | 0.905 | 0.772 |
| **MSWS**$_{CVPR19}$ | Dense-169 | 48.6 | 0.718 | 0.109 | 0.527 | 0.756 | 0.362 | 0.767 | 0.908 | 0.586 | 0.749 | 0.376 | 0.856 | 0.084 | 0.685 | 0.818 | 0.438 |
| **R$^3$Net+**$_{IJCAI18}$ | ResNeXt | 215.0 | 0.795 | 0.063 | 0.728 | 0.817 | 0.599 | 0.828 | 0.058 | 0.763 | 0.817 | 0.601 | 0.915 | 0.036 | 0.877 | 0.895 | 0.740 |
| **CapSal**$_{CVPR19}$ | ResNet-101 | - | 0.699 | 0.101 | 0.482 | 0.674 | 0.396 | 0.823 | 0.072 | 0.691 | 0.808 | 0.605 | 0.882 | 0.062 | 0.782 | 0.850 | 0.654 |
| **SRM**$_{ICCV17}$ | ResNet-50 | 189.0 | 0.769 | 0.069 | 0.658 | 0.798 | 0.523 | 0.826 | 0.058 | 0.722 | 0.824 | 0.592 | 0.906 | 0.046 | 0.835 | 0.887 | 0.680 |
| **DGRL**$_{CVPR18}$ | ResNet-50 | 646.1 | 0.779 | 0.063 | 0.697 | 0.810 | 0.584 | 0.834 | 0.051 | 0.760 | 0.836 | 0.656 | 0.913 | 0.037 | 0.865 | 0.897 | 0.744 |
| **PiCANetR**$_{CVPR18}$ | ResNet-50 | 197.2 | 0.803 | 0.065 | 0.695 | 0.832 | 0.632 | 0.860 | 0.050 | 0.755 | 0.859 | 0.696 | 0.918 | 0.043 | 0.840 | 0.904 | 0.765 |
| **CPD**$_{CVPR19}$ | ResNet-50 | 183.0 | 0.797 | 0.056 | 0.719 | 0.825 | 0.655 | 0.865 | 0.043 | 0.795 | 0.858 | 0.741 | 0.925 | 0.034 | 0.875 | 0.905 | 0.795 |
| **PoolNet**$_{CVPR19}$ | ResNet-50 | 273.3 | 0.808 | 0.056 | 0.729 | 0.836 | 0.675 | 0.880 | 0.040 | 0.807 | 0.871 | 0.765 | 0.932 | 0.033 | 0.881 | 0.917 | 0.811 |
| **BASNet**$_{CVPR19}$ | ResNet-34 | 348.5 | 0.805 | 0.056 | 0.751 | 0.836 | 0.694 | 0.860 | 0.047 | 0.803 | 0.853 | 0.758 | 0.928 | 0.032 | 0.889 | 0.909 | 0.807 |
| **U$^2$-Net** (Ours) | RSU | 176.3 | 0.823 | 0.054 | 0.757 | 0.847 | 0.702 | 0.873 | 0.044 | 0.804 | 0.861 | 0.765 | 0.935 | 0.031 | 0.890 | 0.916 | 0.812 |
| **U$^2$-Net$^\dagger$** (Ours) | RSU | 4.7 | 0.813 | 0.060 | 0.731 | 0.837 | 0.676 | 0.852 | 0.054 | 0.763 | 0.847 | 0.723 | 0.928 | 0.037 | 0.867 | 0.908 | 0.794 |

Table 4: Comparison of our method and 20 SOTA methods on ECSSD, PASCAL-S, SOD in terms of model size, $maxF_\beta$ (↑), $MAE$ (↓), weighted $F^w_\beta$ (↑), structure measure $S_m$ (↑) and relax boundary F-measure $relaxF^b_\beta$ (↑). Red, Green, and Blue indicate the best, second best and third best performance.

| Method | Backbone | Size(MB) | ECSSD (1000) | | | | | PASCAL-S (850) | | | | | SOD (300) | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | $maxF_\beta$ | $MAE$ | $F^w_\beta$ | $S_m$ | $relaxF^b_\beta$ | $maxF_\beta$ | $MAE$ | $F^w_\beta$ | $S_m$ | $relaxF^b_\beta$ | $maxF_\beta$ | $MAE$ | $F^w_\beta$ | $S_m$ | $relaxF^b_\beta$ |
| **MDF**$_{TIP16}$ | AlexNet | 112.1 | 0.832 | 0.105 | 0.705 | 0.776 | 0.472 | 0.759 | 0.142 | 0.589 | 0.696 | 0.343 | 0.746 | 0.192 | 0.508 | 0.643 | 0.311 |
| **UCF**$_{ICCV17}$ | VGG-16 | 117.9 | 0.903 | 0.069 | 0.806 | 0.884 | 0.669 | 0.814 | 0.115 | 0.694 | 0.805 | 0.493 | 0.808 | 0.148 | 0.675 | 0.762 | 0.471 |
| **Amulet**$_{ICCV17}$ | VGG-16 | 132.6 | 0.915 | 0.059 | 0.840 | 0.894 | 0.711 | 0.828 | 0.100 | 0.734 | 0.818 | 0.541 | 0.798 | 0.144 | 0.677 | 0.753 | 0.454 |
| **NLDF+**$_{CVPR17}$ | VGG-16 | 428.0 | 0.905 | 0.063 | 0.839 | 0.897 | 0.666 | 0.822 | 0.098 | 0.737 | 0.798 | 0.495 | 0.841 | 0.125 | 0.709 | 0.755 | 0.475 |
| **DSS+**$_{CVPR17}$ | VGG-16 | 237.0 | 0.921 | 0.052 | 0.872 | 0.882 | 0.696 | 0.831 | 0.093 | 0.759 | 0.798 | 0.499 | 0.846 | 0.124 | 0.710 | 0.743 | 0.444 |
| **RAS**$_{ECCV18}$ | VGG-16 | 81.0 | 0.921 | 0.056 | 0.857 | 0.893 | 0.741 | 0.829 | 0.101 | 0.736 | 0.799 | 0.560 | 0.851 | 0.124 | 0.720 | 0.764 | 0.544 |
| **PAGRN**$_{CVPR18}$ | VGG-19 | - | 0.927 | 0.061 | 0.834 | 0.889 | 0.747 | 0.847 | 0.090 | 0.738 | 0.822 | 0.594 | - | - | - | - | - |
| **BMPM**$_{CVPR18}$ | VGG-16 | - | 0.928 | 0.045 | 0.871 | 0.911 | 0.770 | 0.850 | 0.074 | 0.779 | 0.845 | 0.617 | 0.856 | 0.108 | 0.726 | 0.786 | 0.562 |
| **PiCANet**$_{CVPR18}$ | VGG-16 | 153.3 | 0.931 | 0.046 | 0.865 | 0.914 | 0.784 | 0.856 | 0.078 | 0.772 | 0.848 | 0.612 | 0.854 | 0.103 | 0.722 | 0.789 | 0.572 |
| **MLMS**$_{CVPR19}$ | VGG-16 | 263.0 | 0.928 | 0.045 | 0.871 | 0.911 | 0.770 | 0.855 | 0.074 | 0.779 | 0.844 | 0.620 | 0.856 | 0.108 | 0.726 | 0.786 | 0.562 |
| **AFNet**$_{CVPR19}$ | VGG-16 | 143.0 | 0.935 | 0.042 | 0.887 | 0.914 | 0.776 | 0.863 | 0.070 | 0.798 | 0.849 | 0.626 | 0.856 | 0.111 | 0.723 | 0.774 | - |
| **MSWS**$_{CVPR19}$ | Dense-169 | 48.6 | 0.878 | 0.096 | 0.716 | 0.828 | 0.411 | 0.786 | 0.133 | 0.614 | 0.768 | 0.289 | 0.800 | 0.167 | 0.573 | 0.700 | 0.231 |
| **R$^3$Net+**$_{IJCAI18}$ | ResNeXt | 215.0 | 0.934 | 0.040 | 0.902 | 0.910 | 0.759 | 0.834 | 0.092 | 0.761 | 0.807 | 0.538 | 0.850 | 0.125 | 0.735 | 0.759 | 0.431 |
| **CapSal**$_{CVPR19}$ | ResNet-101 | - | 0.874 | 0.077 | 0.771 | 0.826 | 0.574 | 0.861 | 0.073 | 0.786 | 0.837 | 0.527 | 0.773 | 0.148 | 0.597 | 0.695 | 0.404 |
| **SRM**$_{ICCV17}$ | ResNet-50 | 189.0 | 0.917 | 0.054 | 0.853 | 0.895 | 0.672 | 0.838 | 0.084 | 0.758 | 0.834 | 0.509 | 0.843 | 0.128 | 0.670 | 0.741 | 0.392 |
| **DGRL**$_{CVPR18}$ | ResNet-50 | 646.1 | 0.925 | 0.042 | 0.883 | 0.906 | 0.753 | 0.848 | 0.074 | 0.787 | 0.839 | 0.569 | 0.848 | 0.106 | 0.731 | 0.773 | 0.502 |
| **PiCANetR**$_{CVPR18}$ | ResNet-50 | 197.2 | 0.935 | 0.046 | 0.867 | 0.917 | 0.775 | 0.857 | 0.076 | 0.777 | 0.854 | 0.598 | 0.856 | 0.104 | 0.724 | 0.790 | 0.528 |
| **CPD**$_{CVPR19}$ | ResNet-50 | 183.0 | 0.939 | 0.037 | 0.898 | 0.918 | 0.811 | 0.861 | 0.071 | 0.800 | 0.848 | 0.639 | 0.860 | 0.112 | 0.714 | 0.767 | 0.556 |
| **PoolNet**$_{CVPR19}$ | ResNet-50 | 273.3 | 0.944 | 0.039 | 0.896 | 0.921 | 0.813 | 0.865 | 0.075 | 0.798 | 0.832 | 0.644 | 0.871 | 0.102 | 0.759 | 0.797 | 0.606 |
| **BASNet**$_{CVPR19}$ | ResNet-34 | 348.5 | 0.942 | 0.037 | 0.904 | 0.916 | 0.826 | 0.856 | 0.076 | 0.798 | 0.838 | 0.660 | 0.851 | 0.113 | 0.730 | 0.769 | 0.603 |
| **U$^2$-Net** (Ours) | RSU | 176.3 | 0.951 | 0.033 | 0.910 | 0.928 | 0.836 | 0.859 | 0.074 | 0.797 | 0.844 | 0.657 | 0.861 | 0.108 | 0.748 | 0.786 | 0.613 |
| **U$^2$-Net$^\dagger$** (Ours) | RSU | 4.7 | 0.943 | 0.041 | 0.885 | 0.918 | 0.808 | 0.849 | 0.086 | 0.768 | 0.831 | 0.627 | 0.841 | 0.124 | 0.697 | 0.759 | 0.559 |

## Presentation and visualization

The presentation and visualization will be done as shown below:



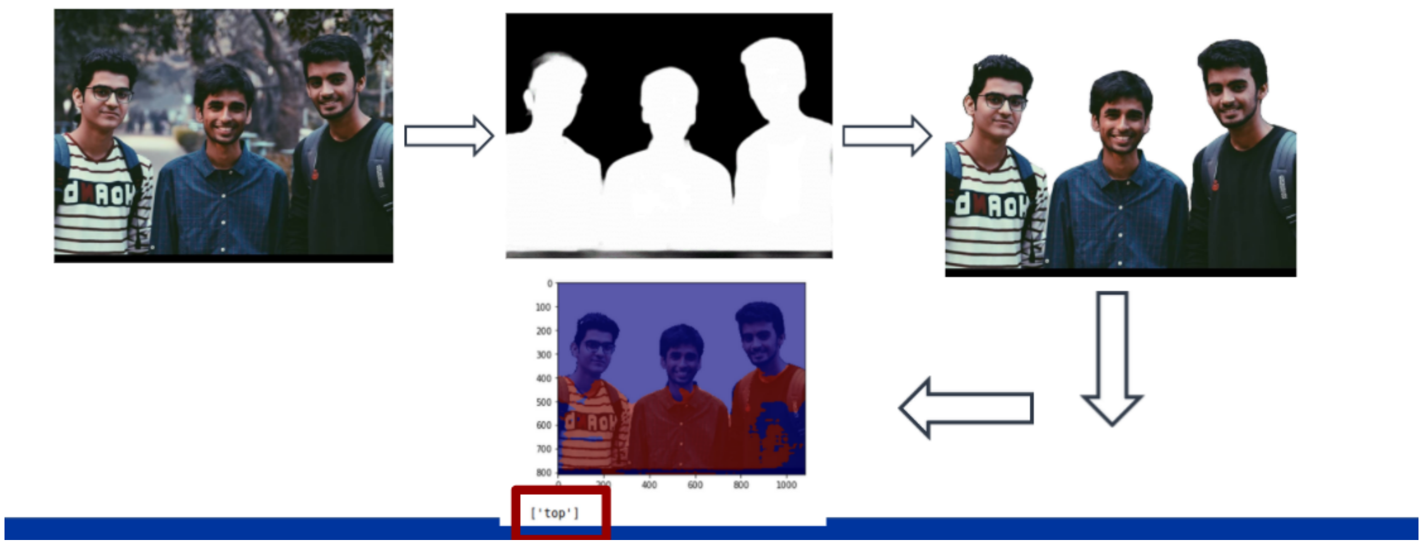

**Input Image**

**Segmented Image**

**Background Removed**

**Output – Fashion apparel type with the highlighted area.**

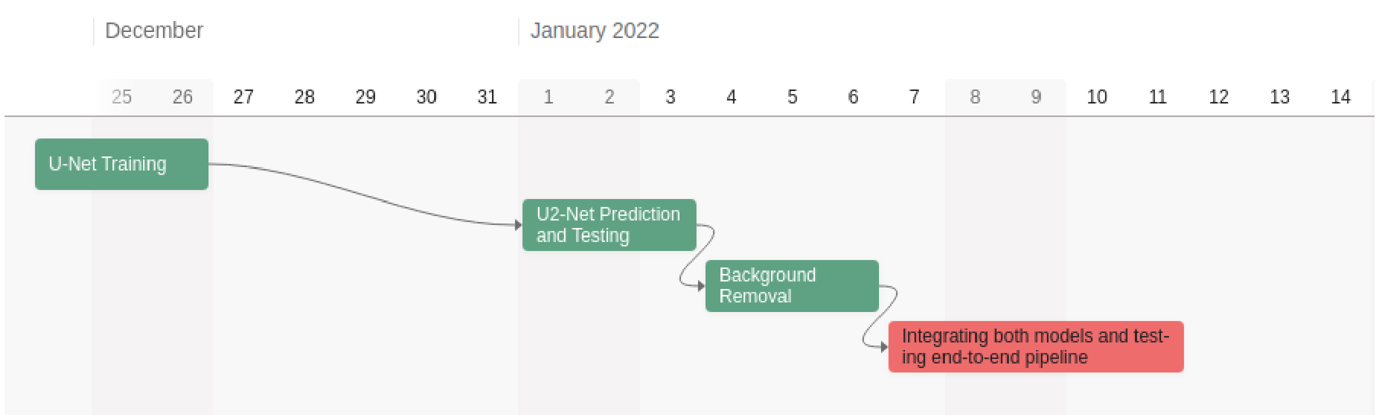
**Another Example**




## Roles

- Shadaab : U-Net Training on DeepFashion2 and end-to-end integration
- Khushi : U-Net Prediction and DeepFashion2 class mappings
- Simran : U2-Net Prediction and Background Removal

## Schedule



Bibliography

[1]
https://www.analyticsvidhya.com/blog/2019/02/tutorial-semantic-segmentation-google-deeplab/#:~:text=Semantic%20segmentation%20is%20the%20task,to%20one%20of%20the%20classes.

[2]
https://www.cogitotech.com/blog/computer-vision-in-ai-and-machine-learning#:~:text=In%20Machine%20Learning%20(ML)%20and,results%20in%20real%2Dlife%20use.

[3] DeepFashion2 Code Repo https://github.com/switchablenorms/DeepFashion2

[4] DeepFashion2 https://arxiv.org/abs/1901.07973

[5] U2-Net: Going Deeper with Nested U-Structure for Salient Object Detection Xuebin Qin, Zichen Zhang, Chenyang Huang, Masood Dehghan, Osmar R. Zaiane and Martin Jagersand https://arxiv.org/pdf/2005.09007

[6] U-Net: Convolutional Networks for Biomedical Image Segmentation, Olaf Ronneberger and Philipp Fischer and Thomas Brox, 2015

[7] https://towardsdatascience.com/iou-a-better-detection-evaluation-metric-45a511185be1