**MAULANA AZAD**

**NATIONAL INSTITUTE OF TECHNOLOGY**

**BHOPAL, INDIA, 462003**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**Human Clothes Detection Using Pspnet**

**Minor Project Report**

**Semester-6**

**Submitted by**

Bhavesh Agrawal 191112215

Ansh Kaushik 191112219

Rahul Singh 191112235

Mohit Tomar 191112277

**Under the Guidance of**

**Dr. B N ROY**

## DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

**Session: 2021-22**

**MAULANA AZAD**

**NATIONAL INSTITUTE OF TECHNOLOGY**

**BHOPAL, INDIA, 462003**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

# CERTIFICATE

This is to certify that the project report carried out **“HUMAN CLOTHES DETECTION USING PSPnet**” by the 3rd year students:

Bhavesh Agrawal 191112215

Ansh Kaushik 191112219

Rahul Singh 191112235

Mohit Tomar 191112277

## Have successfully completed their project in partial fulfillment of their Degree in Bachelor of Technology in Computer Science and Engineering.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Dr. B.N. Roy**

**(Minor Project Supervisor)**

# DECLARATION

We, hereby declare that the following report which is being presented in the Minor Project Documentation Entitled as **“HUMAN CLOTHES DETECTION USING PSPnet”** is an authentic documentation of our own original work and to best of our knowledge. The following project and its report, in part or whole, has not been presented or submitted by us for any purpose in any other institute or organization. Any contribution made to the research by others, with whom we have worked at **Maulana Azad National Institute of Technology**, Bhopal or elsewhere, is explicitly acknowledged in thereport.

Bhavesh Agrawal 191112215

Ansh Kaushik 191112219

Rahul Singh 191112235

Mohit Tomar 191112277

# ACKNOWLEDGEMENT

## With due respect, we express our deep sense of gratitude to our respected guide and coordinator **Dr. Bholanath Roy**, for his valuable help and guidance. We are thankful for the encouragement that he has given us in completing this project successfully.

## It is imperative for us to mention the fact that the report of minor project could not have been accomplished without the periodic suggestions and advice of our project guide **Dr. Bholanath Roy** and project coordinators **Dr. Sanyam Shukla** and **Dr. Namita Tiwari**.

## We are also grateful to our respected HoD, **Dr. Nilay Khare** and Head AI and central computing facility, **Dr Meenu Chawla** for permitting us to utilize all the necessary facilities of the college.

We are also thankful to all the other faculty, staff members and laboratory attendants of our department for their kind cooperation and help. Last but certainly not the least; we would like to express our deep appreciation towards our family members and batch mates for providing the much needed support and encouragement.

# ABSTRACT

Among all captured images in the world, humans are present in most of them and correctly analyzing humans in the data is an important issue in many applications. Face, hand and body clothes are the most studied components. Early works in facial domain tried to solve face detection and localization in the images while recognizing other features like body clothes detection remained a challenging problem till recent years. Automatic analyzing humans in photographs has great potential applications in computer vision containing medical diagnosis, sports, entertainment, Virtual trial room, just to name a few. Detecting Body clothes has much variability in shape and clothing. Adding human characteristics to all aforementioned variabilities makes human analysis quite a challenging task.

In this work we address the problem of human clothes detection in different modalities in both RGB, Grey-scale and depth images, captured some of the body parts and clothespresent as Hair, Upper Clothes, Pants, Left Shoe, Right Shoe hair, face etc.

We dectect the human body cloth via images using PSPNet. In a **PSPNet** architecture, we used DenseNet as beckend for extracting features, Pyramid Pooling Module, Decoder are some other Components of this architecture.

CONTENTS

[1. INTRODUCTION 1](#_Toc99663682)

[1.1. HUMAN PARSING AND SEMENTIC SEGMENTATION 1](#_Toc99663683)

[1.2. DATABASE 1](#_Toc99663685)

[2. LITERATURE REVIEW AND SURVEY 2](#_Toc99663687)

[3. GAPS IDENTIFIED 3](#_Toc99663688)

[4. PROPOSED WORK AND METHODOLOGY 3](#_Toc99663689)

[4.1. PREPROCESSING DATA 3](#_Toc99663692)

[4.2. BUILDING MODEL 4](#_Toc99663695)

[*4.2.1.* *ENCODER* 4](#_Toc99663696)

[*4.2.1.1.* *DENSENET* 4](#_Toc99663696)

[*4.2.1.1.1. TRANSITION LAYER* 5](#_Toc99663697)

[*4.2.1.1.2. DENSE BLOCK* 5](#_Toc99663697)

[*4.2.1.1.2.1. DENSE LAYER* 6](#_Toc99663697)

[*4.2.1.1.3. PYRAMID POOLING MODULE* 7](#_Toc99663697)

[*4.2.2.* *DECODER* 8](#_Toc99663696)

[*4.2.2.1.* *UPSAMPLING OF FEATURE MAP* 8](#_Toc99663696)

[*4.2.2.2.* *CLASSIFICATION LAYER* 8](#_Toc99663696)

[4.3. TRAINING AND TESTING MODEL 10](#_Toc99663698)

[5. RESULT AND DISCUSSION 11](#_Toc99663699)

[5.1. OVERALL ACCURACY 11](#_Toc99663700)

[5.2. MEAN ACCURACY 11](#_Toc99663701)

[5.3. INTERSECTION OVER UNION (IOU) 11](#_Toc99663702)

[6. FUTURE WORK 13](#_Toc99663706)

[7. CONCLUSION 14](#_Toc99663706)

[8. REFERENCES 15](#_Toc99663708)

**LIST OF FIGURES**

[Figure 1 - DenseNet Architecture 5](#_Toc99873734)

[Figure 2 – Dense Layer Architecture 6](#_Toc99873735)

[Figure 3- Pyramid Pooling Module 7](#_Toc99873736)

[Figure 4 – Prediction 1 12](#_Toc99873737)

[Figure 5 – Prediction 2 12](#_Toc99873737)

**LIST OF TABLES**

[Table 1 – LIP DATASET 2](#_Toc99663497)

[Table 2 – PSPNET Architecture](#_Toc99663498) 9

[Table 3 – Model Evaluation Matrix 11](#_Toc99663499)

# INTRODUCTION

* 1. HUMAN PARSING & SEMENTIC SEGMENTATION

Human parsing or human body part semantic segmentation, has been an active research topic due to its wide potential application.It is widely used in pedestrian detection, unmanned driving and medical image applications.It is thought as a specific image semantic segmentation task.Human parsing namely decomposing the human part in image into body regions,is a pixel-level prediction task that requires understanding features in both higher and lower levels.Human parsing is widely used in human behavior analysis, posture estimation and tidal current synthesis.At present, the methods for human parsing based on deep learning can achieve good segmentation results. Because of the emergence of Fully Convolutional Network(FCN), the framework of encoder-decoder is introduced into the neural network. That makes neural network can be applied to image semantic segmentation tasks including human parsing, greatly improving the segmentation effect.A series of great performance neural network models are applied to human parsing, such as FCN8s, U-net, SegNet,and PSPNet.In this work we are using PSPNEt as it performs better than other semantic segmentation nets like FCN,U-Net,Deeplab.

* 1. DATABASE

It is highly desired to have a large-scale dataset composed of representative instances with varied clothing appearances, strong articulation, partial (self-) occlusions, and truncation at image borders, diverse viewpoints and background clutters. Here have used LIP dataset which contains in total 50,462 images in the LIP dataset including 19,081 full-body images, 13,672 upper-body images, 403 lower-body images, 3,386 head-missed images, 2,778 back-view images and 21,028 images with occlusions. We split the images into separate training, validation and test sets. Following random selection, we arrive at a unique split consisting of 30,462 training and 10,000 validation images with publicly available annotations, as well as 10,000 test images with annotations withheld for benchmarking purpose. LIP validation set which contains 4,277 images with occlusions, 5,452 full-body images, 793 upper-body images, 112 head-missed images and 661 back-view images.

Table 1 – LIP Dataset

|  |  |  |  |
| --- | --- | --- | --- |
| DATASET | TRAINING | TEST | CATEGORIES |
| Fashionista [31] | 456 | 229 | - |
| PASCAL – Person – Part [6] | 1,716 | 1,817 | - |
| ATR [17] | 16,000 | 1,000 | - |
| LIP | 30,462 | 10,000 | 20 |

# LITERATURE REVIEW AND SURVEY

There has been a great deal of work in the last few years on the subject of clothing recognition. However, many works focus on special clothing classes and applications and it is only recently that generic clothing recognition has been directly tackled. The vast majority of the related work, as well as the approach presented here, is based on person detection cloth detection. Clothing parsing, also known as semantic segmentation of clothing items, is an important tool in Clothing image analysis. One of the first approaches toclothing parsing and creation of datasetfor benchmarking purposes. This approach washighly reliant on the performance of the pose estimation model. More recent work has used powerful deep convolutional neural networks achieving state-of-the-artresults without the need for additional meta tags.Augment fullyconvolutional networks (FCNs) by a branch that predicts combinatorialpreference of garments.

Recently, in order to facilitate thetraining of better performing models, some big-scaledatasets were released like ModaNet, LIPdataset etc. To our knowledge, not much research was conducted using the LIP dataset. The LIP (Look into Person) dataset is a large-scale dataset focusing on semantic understanding of a person.It contains 50,000 images with elaborated pixel-wise annotations of 19 semantic human part labels. The images are collected from real-world scenarios and the subjects appear with challenging poses and view, heavy occlusions, various appearances and low resolution.

# GAPS IDENTIFIED

The Dataset used in our research and the Network model used for scrutinising the dataset and classifying the image make it a fresh view on human body colthes detection. Many papers have been published based on Human parsing and semantic segmentation, but the dataset used differs in model and in efficiency on a research basis. One of the challenging part of human parsing is to detect the hairs because most of the time colour of hair matches with that of background, analyisng and training datasets over the networks also take huge amount out of time and require high-end machinery. Our project is interested in the increase in efficiency of the classification and making the model more capable of using the LIP dataset.

# PROPOSED WORK AND METHODOLOGY



## PREPROCESSING DATA

## We used torch package for data preprocessing. The torch package contains data structures for multi-dimensional tensors.

## We transformed Images by resizing them, converting to tensors and normalizing them. We Converted training images to RGB images so that no images are left grayscaled. Used ground truth images (Images with annotate objects as rectangles, lines, or pixel labels. Pixel labeling is a process in which each pixel in an image is assigned a class or category, which can then be used to train a pixel-level segmentation algorithm.) and associated them with annotations.

## Our ground-truth dataset is a regular dataset, but with annotations added to it. Annotations here are a new column of a spreadsheet so that, the machine learning algorithm should learn to output.

|  |
| --- |
| def get\_transform():      transform\_image\_list = [          transforms.Resize((256, 256), 3),          transforms.ToTensor(),          transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]),      ]      transform\_gt\_list = [          transforms.Resize((256, 256), 0),          transforms.Lambda(lambda img: np.asarray(img, dtype=np.uint8)),      ]      data\_transforms = {          'img': transforms.Compose(transform\_image\_list),          'gt': transforms.Compose(transform\_gt\_list),      }      return data\_transforms  def get\_sample\_at\_index(self, index):        img = Image.open(self.img\_path[index])        if img.mode == 'L': # If image is greyscale convert to RGB            img = img.convert('RGB')        gt = Image.open(self.gt\_path[index])        if gt.mode == 'RGB': # If ground truth is RGB convert to greyscale (L)            gt = gt.convert('L')        # Run transforms on training example        img = self.transform['img'](img)        gt = self.transform['gt'](gt)          return img, gt |

* 1. BUILDING MODEL

A PSPNet has 2 segments i.e. **Encoder** and **Decoder**. Encoder is responsible for extracting out different features from the image and mostly based on CNN, DCNN, FCN. Whereas **Decoder** is the one the use these extracted features and upscale them to predict the classes[7].

* + 1. ENCODER
       1. DENSENET

We Use DenseNet as a backend(Encoder) to extract the features of Images. We use pre-trained weights in DenseNet for better efficiency in extracting features.DenseNet(Dense Convolution Network) has 4 Dense block and between each Dense block there is a transition layer

* + - * 1. TRANSITION LAYER

In between every Dense Block there is a transition layer which does the convolution and pooling. We here used 1\*1 convolution layer followed by 2\*2 average pooling layer at the first transition layer and for remaining transition layer we used 1\*1 average pooling layer to make the image size compatible for up scaling in Decoder part.

class \_Transition(nn.Sequential):

    def \_\_init\_\_(self, num\_input\_features, num\_output\_features, downsample=True):

        super(\_Transition, self).\_\_init\_\_()

        self.add\_module('norm', nn.BatchNorm2d(num\_input\_features))

        self.add\_module('relu', nn.ReLU(inplace=True))

        self.add\_module('conv', nn.Conv2d(num\_input\_features, num\_output\_features,kernel\_size=1, stride=1, bias=False))

        if downsample:

            self.add\_module('pool', nn.AvgPool2d(kernel\_size=2, stride=2))

        else:

            self.add\_module('pool', nn.AvgPool2d(kernel\_size=1, stride=1))

* + - * 1. DENSE BLOCK

In the Model, we build 4 dense block with varying number of layers such as 4 blocks contains 6,12,24,16 dense layer respectively[8].

class \_DenseBlock(nn.Sequential):

    def \_\_init\_\_(self, num\_layers, num\_input\_features, bn\_size, growth\_rate, drop\_rate):

        super(\_DenseBlock, self).\_\_init\_\_()

        for i in range(num\_layers):

            layer = \_DenseLayer(num\_input\_features + i \* growth\_rate, growth\_rate, bn\_size, drop\_rate)

            self.add\_module('denselayer%d' % (i + 1), layer)

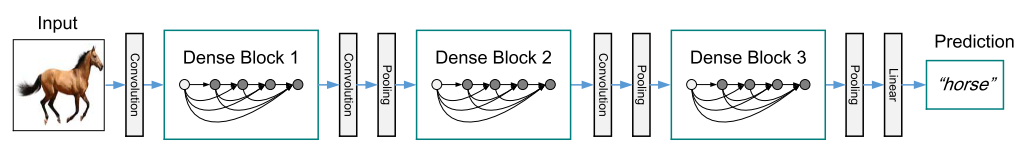


Fig 1: DenseNet Architecture

DENSE LAYER

A denseLayer accepts an input, concatenates the input together and passes it to 1\*1 convolution layer this reduces the feature to 4\*k where k is growth rate then this act as bottleneck in the model as it reduces the input feature of feature map thus increasing the computational efficiency then this input map is featured map is passed to 3\*3 conv layer for convolution making the final feature map to of size k. So after every dense block the output features of feature map will be init\_features+(numbers of denseLayer)\*growth\_rate

class \_DenseLayer(nn.Sequential):

    def \_\_init\_\_(self, num\_input\_features, growth\_rate, bn\_size, drop\_rate):

        super(\_DenseLayer, self).\_\_init\_\_()

        self.add\_module('norm1', nn.BatchNorm2d(num\_input\_features)),

        self.add\_module('relu1', nn.ReLU(inplace=True)),

        self.add\_module('conv1', nn.Conv2d(num\_input\_features, bn\_size \* growth\_rate, kernel\_size=1, stride=1, bias=False)),

        self.add\_module('norm2', nn.BatchNorm2d(bn\_size \* growth\_rate)),

        self.add\_module('relu2', nn.ReLU(inplace=True)),

        self.add\_module('conv2', nn.Conv2d(bn\_size \* growth\_rate, growth\_rate, kernel\_size=3, stride=1, padding=1, bias=False)),

        self.drop\_rate = drop\_rate

    def forward(self, x):

        new\_features = super(\_DenseLayer, self).forward(x)

        if self.drop\_rate > 0:

            new\_features = F.dropout(new\_features, p=self.drop\_rate, training=self.training)

        return torch.cat([x, new\_features], 1)

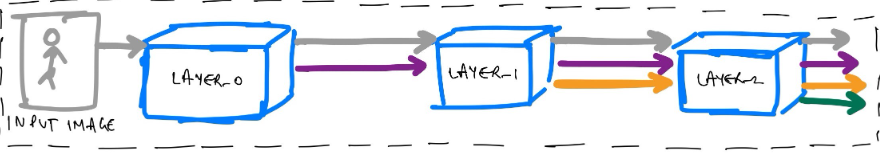


Fig 2: Dense Layer Architecture

* + - * 1. PYRAMID POOLING MODULE

Pyramid Pooling Module is the most important part of this model as it extracts the global features from images. This gets the feature map from the DenseNet model which is pooled to different sizes and passed to convolution layers then to make them of same size we upscale all the feature map and concatenated together which then passed to decoder[7]

class PSPModule(nn.Module):

    def \_\_init\_\_(self, features, out\_features=1024, sizes=(1, 2, 3, 6)):

        super().\_\_init\_\_()

        self.stages = []

        self.stages = nn.ModuleList([self.\_make\_stage(features, size) for size in sizes])

        self.bottleneck = nn.Conv2d(features \* (len(sizes) + 1), out\_features, kernel\_size=1)

        self.relu = nn.ReLU()

    def \_make\_stage(self, features, size):

        prior = nn.AdaptiveAvgPool2d(output\_size=(size, size))

        conv = nn.Conv2d(features, features, kernel\_size=1, bias=False)

        return nn.Sequential(prior, conv)

    def forward(self, feats):

        h, w = feats.size(2), feats.size(3)

        priors = [F.upsample(input=stage(feats), size=(h, w), mode='bilinear') for stage in self.stages] + [feats]

        bottle = self.bottleneck(torch.cat(priors, 1))

        return self.relu(bottle)

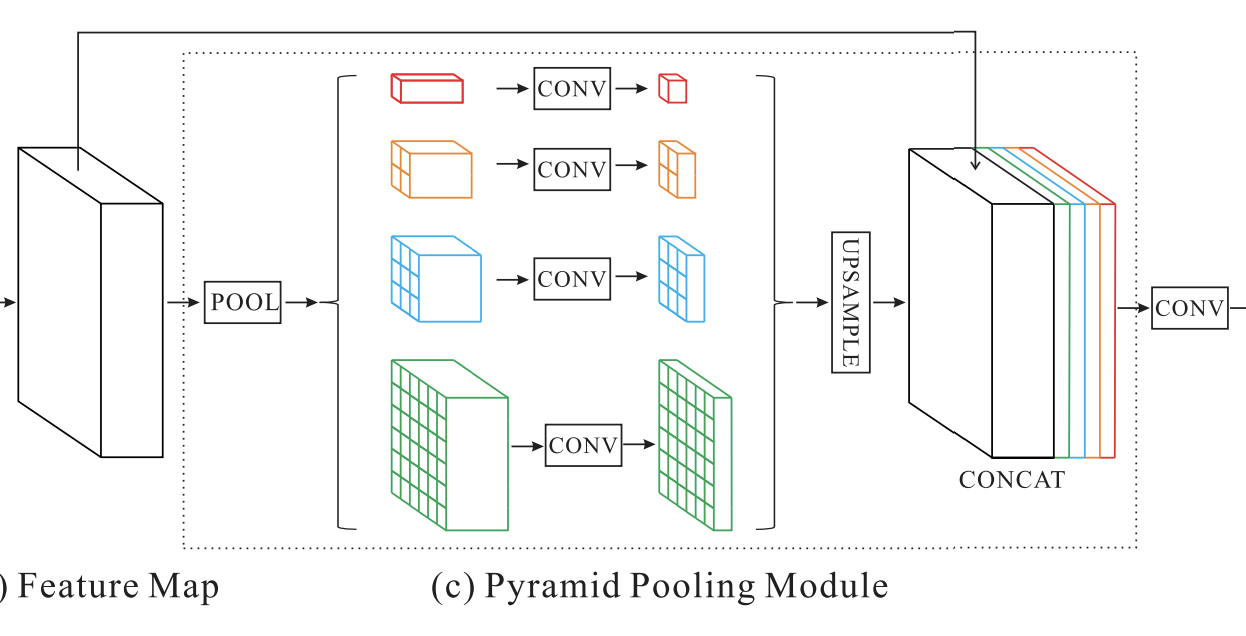
Here we first pass the feature map with average pooling layer each resulting in 1\*1, 2\*2, 3\*3, 6\*6 feature map which then is passed to a convolution layer of 1\*1 with same output feature to extract global and local objects then each of these feature map is **bilinear upsampled** to original size and all these with original feature map are concatenated to each other which then is passed to a bottleneck layer to increase the computational efficiency by limiting the features of feature map[10]

Fig 3: Pyramid Pooling Module

* + 1. DECODER

After encoding, we passes the feature map into decoder for up scaling and convert those feature map into prediction

* + - 1. UPSAMPLING OF FEATURE MAP

The feature map passed to decoder has dimension as h/8,w/8 where h and w are dimension of original image so we have to pass the feature map from a bilinear upscaling 3 times to make the image of original size and with each upscaling we also pass it to a 3\*3 conv[8].

class PSPUpsample(nn.Module):

    def \_\_init\_\_(self, in\_channels, out\_channels):

        super().\_\_init\_\_()

        self.conv = nn.Sequential(

            nn.Conv2d(in\_channels, out\_channels, 3, padding=1),

            nn.BatchNorm2d(out\_channels),

            nn.PReLU()

        )

    def forward(self, x):

        h, w = 2 \* x.size(2), 2 \* x.size(3)

        p = F.upsample(input=x, size=(h, w), mode='bilinear')

        return self.conv(p)

* + - 1. CLASSIFICATION LAYER

Finally this feature map is passed to a convolution layer with outfeature equal to 20 representing different classes to be predicted with activation function as Softmax[4].

Table 2 – PSPNet Architecture

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Layers** | **Output** | **Architecture** |
| Start features |  | (ORIGINAL) 256 X 256X 3 | 7x7 conv , stride 2 |
| Densenet | |  | | --- | | Dense Block (1) | | Transition Layer (1) | | Dense Block (2) | | Transition Layer (2) | | Dense Block (3) | | Transition Layer (3) | | Dense Block (4) | | |  | | --- | | 128 X 128 X 64  64 X 64 X 64 | | 64 X 64 X 256 | | 32 X 32 X 128  32 X 32 X 512 | | 32 X 32 X 256 | | 32 X 32 X 1024  32 X 32 X 512 | | 32 X 32 X 256 | | 32 X 32 X 1024 | | |  | | --- | | 1 x 1 conv  3 x 3 conv x 6 | | { 1 x 1 conv }  2 x 2 average pool | | 1 x 1 conv  3 x 3 conv x 12 | | { 1 x 1 conv }  2 x 2 average pool | | 1 x 1 conv  3 x 3 conv x 24 | | { 1 x 1 conv }  2 x 2 average pool | | 1 x 1 conv  3 x 3 conv x 16 | |
| Pyramid Polling | |  | | --- | | Stages | | Upsample | | BottleNeck | | |  | | --- | | 1 X 1 X 1024  2 X 2 X 1024  3 X 3 X 1024  6 X 6 X 1024 | | 32 X 32 X 5120 | | 32 X 32 X 1024 | | |  | | --- | | AdaptiveAvgPool2d {1 x 1}  AdaptiveAvgPool2d {2 x 2}  AdaptiveAvgPool2d {3 x 3}  AdaptiveAvgPool2d {6 x 6} | | {1 x 1 conv } x 4 | | Bilinear Upsample | | {1 x 1 conv } | |
| Decoder | |  | | --- | | PSPUpsample (1) | | PSPUpsample (2) | | PSPUpsample (3) | | Final (Classification) | | |  | | --- | | 64 X 64 X 1024 | | 64 X 64 X 256 | | 128 X 128 X 256 | | 128 X 128 X 64 | | 256 X 256 X 64 | | 256 X 256 X 64 | | 256 X 256 X 20 | | |  | | --- | | Bilinear Upsample | | { 3 x 3 conv } | | Bilinear Upsample | | { 3 x 3 conv } | | Bilinear Upsample | | { 3 x 3 conv } | | { 1 x 1 conv } | |

* 1. TRAINING AND TESTING MODEL

For training and testing purposes of our model, we should have our data broken down into two distinct dataset splits.

The Training Set is the set of data that is used to train and make the model learn the hidden features/patterns in the data.

The Test Set is a separate set of data used to test the model after completing the training.

In total, there are 40,462 images in the LIP dataset including 19,081 full-body images, 13,672 upper-body images, 403 lower-body images, 3,386 head-missed images, 2,778 back-view images and 21,028 images with occlusions. We split the images into separate training, validation and test sets. Following random selection, we arrive at a unique split consisting of 30,462 training with publicly available annotations, as well as 10,000 test images with annotations withheld for benchmarking purpose.

The training as well as learning of a model can be done using the Adam optimizer or any other kind of optimizer. The given optimizer uses a categorical cross entropy function and does the adaptation of weights. The model is trained with a fit function and evaluated using test data.

1. **RESULT AND DISCUSSION**

The 3 parameters calculated for in out project as output are as follows:

* 1. Overall Accuracy

It tells us about total how any pixels of predicted image are correctly classified

**Overall\_acc=sum(equal\_pixel\_in\_predicted\_and\_label\_img)/sum(pixels\_in\_label)**

* 1. Mean Accuracy

It tells us about the area of predicted image that is correctly predicted

**Mean\_acc=Area\_of\_overlap/Area\_of\_label\_image**

* 1. Intersection over union(IoU)

It is a number from 0 to 1 that specifies the amount of overlap between the predicted and ground truth bounding box.

**IoU=Area\_of\_overlap/Area\_of\_Union**

* 0 value of IoU means there is no overlap between boxes
* 1 value of IoU means the predicted image completely overlap the ground truth image

Table 3 – Model Evaluation Matrix

|  |  |
| --- | --- |
| Parameters | Value |
| Overall Accuracy | 81.1932 |
| Mean Accuracy | 78.4522 |
| Mean IOU | 0.5832 |



Fig: 4 – Prediction 1

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

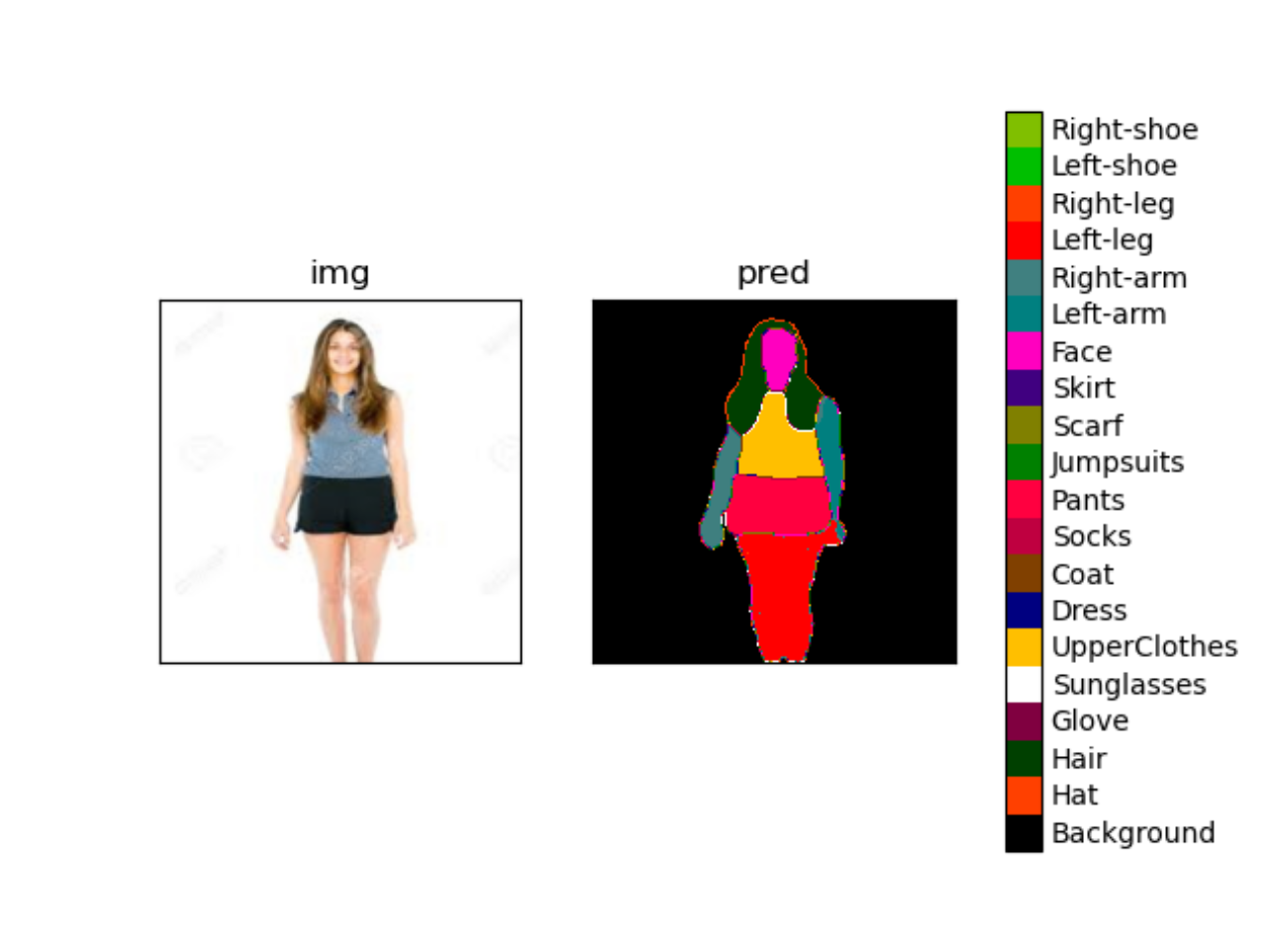


Fig: 5 – Prediction 2

# FUTURE WORK

Virtual try-on is a rapidly growing field of research with a variety of real-life applications. We find great application of our project in that field. Segmentationmask generation is a relatively new but fast developing field that has the potential to improve thecurrent state and use case scenarios of virtual try-on.As future work, a finer model that takes into accountmultiple preson detection in the training set can boostdetection performance, while also help mis-classifications.Gender classification on the detected people can also boostperformance. By generalizing our model, we can includeboth attributes and gender information for the clothing classdetection and suggestion stages of our approach.Also detecting different hand position and elbow movement can help in increasing the efficiency of the model.

And also as a future we would like to work to expand on this to the 3D images which could potentially be useful for virtualtry-on in 3D environments such as augmented and virtual reality. Additionally, future work can alsoexpand the classes being identified such as specific types of headwear or accessories.

# CONCLUSION

This project work addresses the clothing parsing task withmotivation of its applicability in real-life scenarios.In this work we present a fully automated clothing parsing approach that can trivially scale to hundreds of product classes and millions of product images. We achieve clothing detection performance comparable to the state-of-the-art on a very recent annotateddataset, while using low end technology and using google colab as an online interpreter with an efficiency of more than 81%.

# REFERENCES

[1] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7291–7299, 2017.

[2]Xintong Han, Zuxuan Wu, Zhe Wu, Ruichi Yu, and Larry S Davis. Viton: An image-based virtual try-on network. In CVPR, 2018.

[3] Warren Jouanneau, Aurelie Bugeau, Marc Palyart, Nicolas Papadakis, and Laurent Vezard. Where are my clothes? a multi-level approach for evaluating deep instance segmentation architectures on fashion images. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3951–3955, 2021.

[4]Khurana, T., Mahajan, K., Arora, C., and Rai, A. (2018). Exploiting texture cues for clothing parsing in fashion images. In 2018 25th IEEE International Conference on Image Processing (ICIP), pages 2102–2106. IEEE.

[5] Maeda-Gutierrez, V., Galvan-Tejada, C. E., Zanella-Calzada, L. A., Celaya-Padilla, J. M., Galván-Tejada, J. I., Gamboa-Rosales, H., ... & Olvera-Olvera, C. A. (2020). Comparison of convolutional neural network architectures for classification of tomato plant diseases. *Applied Sciences*, *10*(4), 1245.

[6]Z. Song, M. Wang, X. Hua, and S. Yan. Predicting occupation via human clothing and contexts. In ICCV, 2011.

[7]Hengshuang Zhao Jianping Shi Xiaojuan Qi Xiaogang Wang Jiaya Jia The Chinese University of Hong Kong SenseTime Group Limited. Pyramid Scene Parsing Network. In arXiv:1612.01105v2 [cs.CV] 27 Apr 2017

[8]Gao Huang and Zhuang Liu and Laurens van der Maaten and Kilian Q. Weinberger, Densely Connected Convolutional Networks, arXiv 1608.06993 (2016)

[9]J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.

[10]Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In ACM MM, 2014.

[11] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In ECCV, 2018.