CS464 Spring 2020

Group 5 Term Project Report

HUMAN BODY DETECTION

Group Members:

Ömer Aysal Aytaç - 21502996

Muhammed Cemil Şişman - 21502560

Ahmet Çelebi Kocadağ - 21201739

Yusuf Furkan Salcan - 21601748

Semih Erdoğan - 21501368

10.5.2020

1. Introduction

Our project is aiming to detect human bodies from images. We will be using the LSP/MPII-MPHB Database [1]. This dataset provides images with frames, which are given as coordinates. First, we did a preprocessing on our dataset. We eliminated corrupted images and images that had more than one body. We will be using and comparing 3 different networks which are; pretrained Resnet18 with transfer learning, not pretrained Resnet18 and an 18 layered convolutional neural network. We are also using Intersection over Union metric to analyze the correctness of our detection.

2. Problem Description

We are trying to detect the position of human bodies in a given image. After the detection, we will frame the body of the human. We will evaluate the correctness of our detection by using the IoU (Intersection over Union metric).

3. Methods

3.1 Dataset

For this project, we will be using the LSP/MPII-MPHB Database [1]. In this dataset, there are 26,675 natural images of people in different places and different poses. In these images, there are 29,732 different human bodies. Positions of these human bodies are given as coordinates of a rectangle that contains that human body.

3.1 Preprocessing

In the preprocessing we have discarded images which contain more than one body. And some images were corrupted and not readable. After the elimination 13000 image left. We used 11000 in training 1000 in validation and 1000 in test. After forming datasets, each image is resized to (3,244,244) format. Also, corresponding body labels are resized according to the new image size. Labels are size of 4 for each image and contain the bounding box coordinates.

3.2 Models

In this project we have used 3 different model and compared their differences. We have used Resnet18 model with pretrained weights and with random weights. Also, we have constructed a 18 layer CNN model that has similar structıre with Resnet18 but without residual connections. Models are constructed by using PyTorch libraries.

In our models we have used smooth L1 loss as the loss function. Which is defined as follows:

$$\text{loss}(x, y) = \frac{1}{n} \sum_i z_i$$

$$z_i = \begin{cases} 0.5(x_i - y_i)^2, & \text{if } |x_i - y_i| < 1 \\ |x_i - y_i| - 0.5, & \text{otherwise} \end{cases}$$

Equation 1: Smooth L1 Loss Function [2]

where x is the ground truth and y is the prediction. This loss function behaves like L2 loss for small losses which increases the significance of small losses and yields better results compared to regular L1 loss.

3.2.1 Resnet18 (Pretrained)

Resnet is a CNN structure with residual connections. In residual network type, there are some shortcut connections that make identity mapping to reduce the training and test error. In plain network types, when the network goes deeper, accuracy converges and then the training error increases. This is caused by diminishing gradients problem. By introducing identity mapping, the material learned from previous layers is not forgotten by the deeper layers. As it is shown in Figure 1, identity mapping adds the outputs of the $□^{□h}$layer to the inputs of the $(□ + 2)^{□h}$ or $(□ + 3)^{□h}$layer [3].

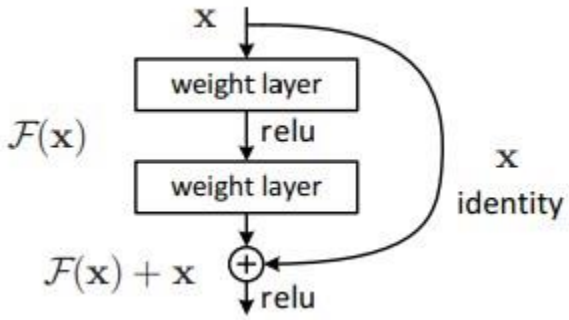


Figure 1: Identity Mapping [3]

In this model we have used the weights that is pretrained on the ImageNet dataset as the transfer learning approach. We have changed the fully connected layer of the ResNet18 with an output layer of size 4.

3.2.2 Resnet18 (Not Pretrained)

In this model everything kept same with previous model but initially random weights are used instead of pretrained weights.

3.2.3 CNN - 18

In this model, we used the same layer types with Resnet18 [3]. There are 17 convolutional layers, 1 max pooling layer, 1 average pooling layer and 1 fully connected layer which is placed to the end of the network. We can see the layer structures in the figure 1. However, in this model, we used plain network type instead of using the residual network type that is used in the Resnet18.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1: Architectures for Resnet Networks [3]

As we can see, the plain network that we used is 18-layer network in the table 1. In this model, we performed the downsampling by setting the stride of the first layers of conv3_x, conv4_x and conv5_x equal to 2. By this way, we reduced the output sizes for every 4 layers [3].

3.3 Validation

For the validation purpose, we divided our training set into two parts. First one was the 90% percent of the training set to train the models and the rest for the validation purpose. After finding the best model for each case, we have performed the training again over the entire training set to conclude the validation. For this purpose, first we needed to determine the intervals of epoch and batch sizes to run training with respect to them. For the non-pretrained ResNet18, we chose the batch sizes of 10, 20, 50, 100, 150 and 200. 200 was selected as the maximum because the values greater than 200 were causing memory problems with our GPU. For the CNN, we chose batch sizes of 10,20, 50 and 100. As for the epoch for these two cases, we selected 8. For the pretrained ResNet18, we confined ourselves with the batch sizes of 10,20,50 and 100 and epoch of 3, simply because the model was pretrained.

We also needed to determine the performance metric to evaluate the validated models and select the best model amongst them according to that metric. Considering the boxes drawn around the bodies in our dataset as ground truth, we needed to compare that drawn box and the box we draw around the body after detection. In order to do this, we used a performance metric called Intersection over Union (IoU), also known as the Jaccard Index [4]. IoU is simply the ratio of the intersection of the predicted and ground truth boxes over the union of them. Let the box P represent out prediction and the box G represent the ground truth box. Then, IoU is:

$$\square\square\square = \frac{\cap(\square, \square)}{\cup(\square, \square)}$$

We have generated a model for each batch size and epoch to evaluate those models with respect to their IoU performance over validation data set. For that purpose, we have generated and saved a model for each and every combination of epoch and batch sizes. Which means that we have generated 32 models for CNN, 48 models for non-pretrained ResNet18 and 12 models for pretrained ResNet18. The process of generating those models was very challenging and tedious. We have spent more than 35 hours to create those models. After creation of those models, we have tested each and every model over the validation dataset and chose a batch size- epoch combination which performed the best and then we trained that combination again over the entire training set.

In the following figures, the IoU vs epoch plots are given, in order to visualize our justification, for each batch size for all three cases:

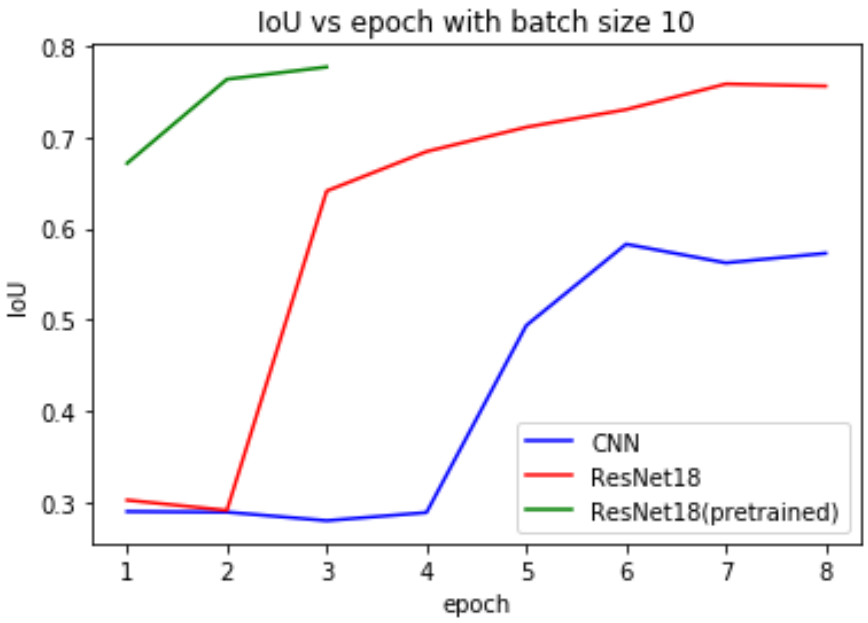


Figure 2: IoU vs Epoch Plot for Batch Size 10

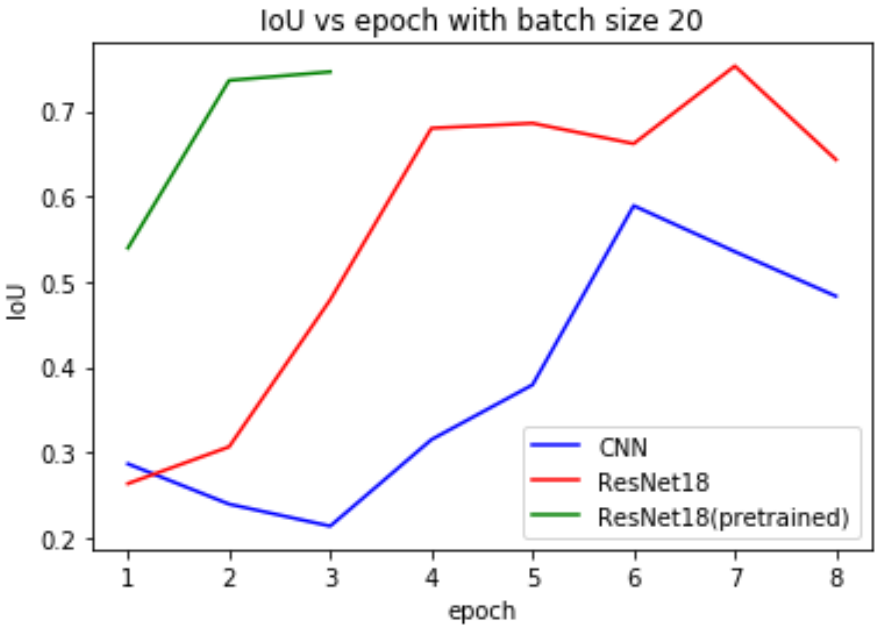


Figure 3: IoU vs Epoch Plot for Batch Size 20

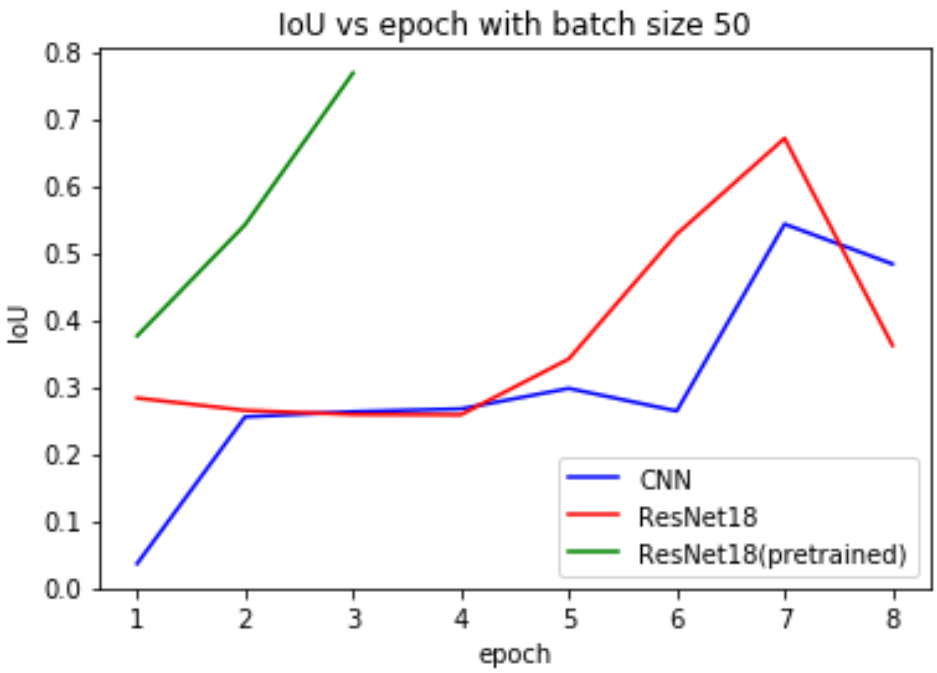


Figure 4: IoU vs Epoch Plot for Batch Size 50

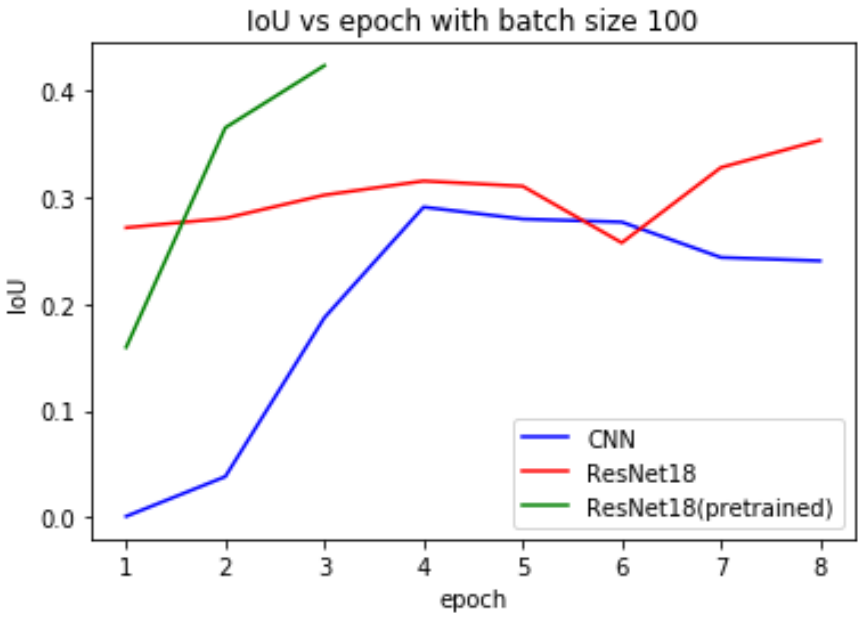


Figure 5: IoU vs Epoch Plot for Batch Size 100

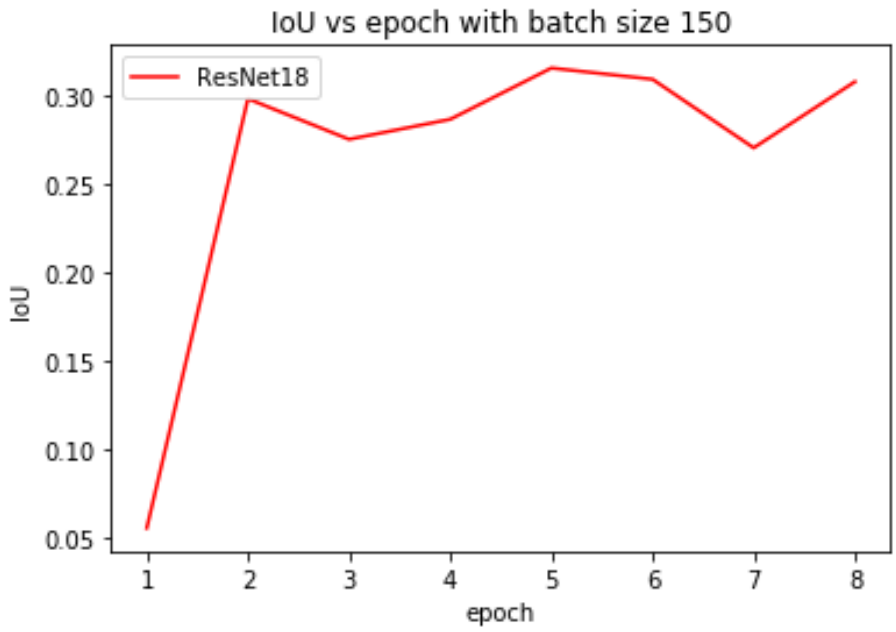


Figure 6: IoU vs Epoch Plot for Batch Size 150

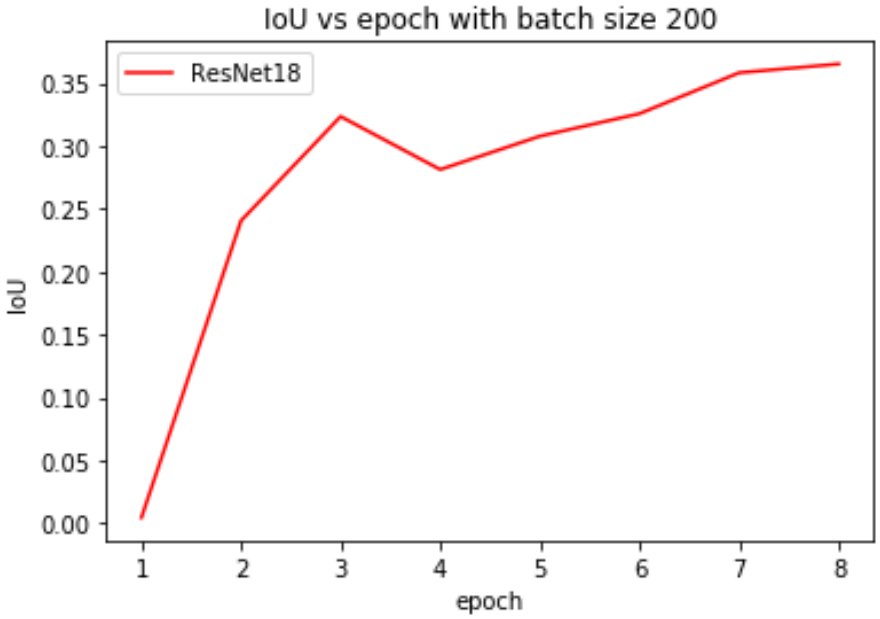
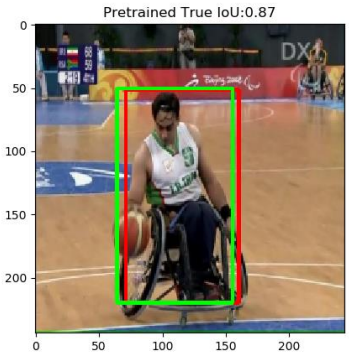
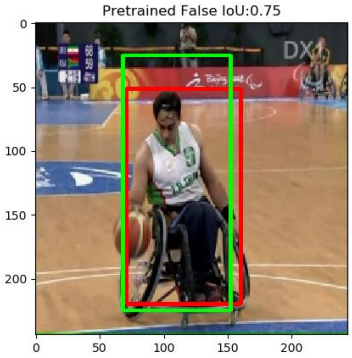
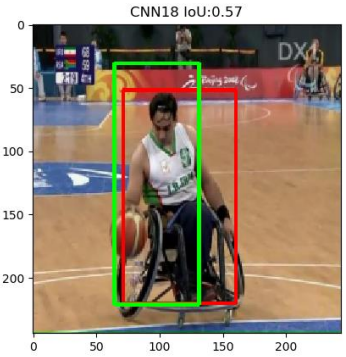


Figure 7: IoU vs Epoch Plot for Batch Size 200

From the figures given above, one can make the same selection of batch size – epoch combination as we did. For the CNN case, we have chosen the batch size 10 and epoch 6, for non-pretrained ResNet18 we chose batch size 10 and epoch 7 and finally, for the pretrained ResNet18, we chose batch size 10 and epoch 3. In the following table, batch size, epoch and IoU values for the models that we chose according to our validation is given for each case:

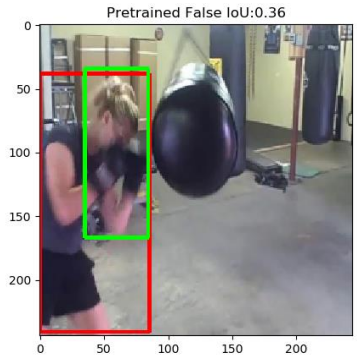






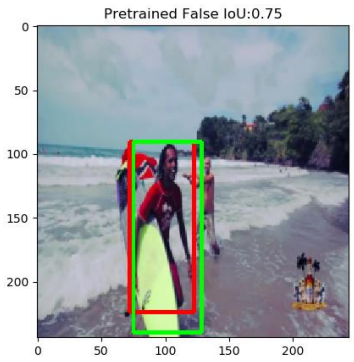
Neural Network	Batch Size	Epoch	IoU
ResNet18	10	7	0.7581
ResNet18(pretrained)	10	3	0.7767
CNN - 18	10	6	0.5811

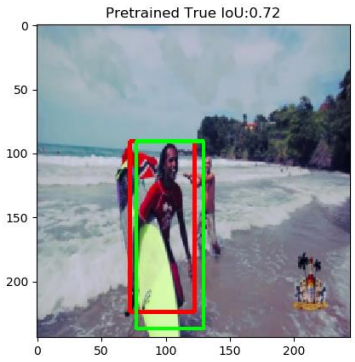
Table 2: Batch Size, Epoch and IoU Values for the Best Models Chosen for Each Case

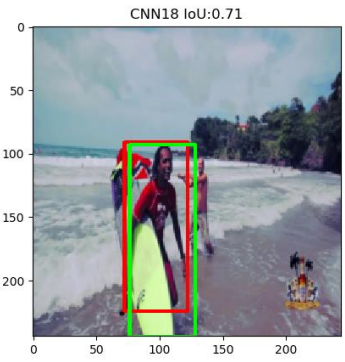


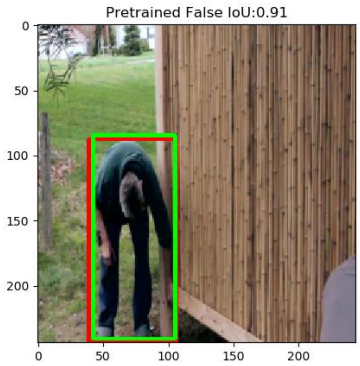
4. Results

Models test results on some sample images are as follows. Green line shows the predictions and read line shows the ground truth:

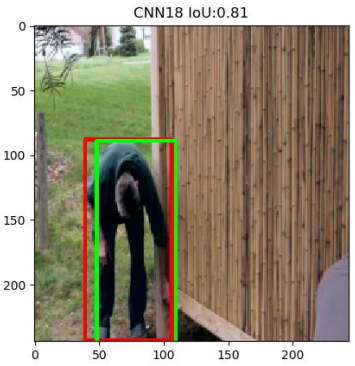

Pretrained False IoU:0.75

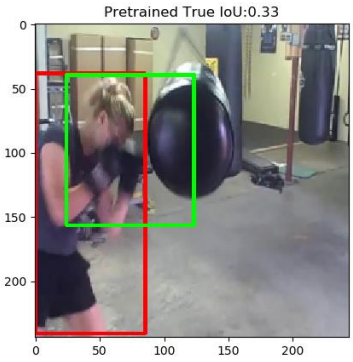

Pretrained True IoU:0.72

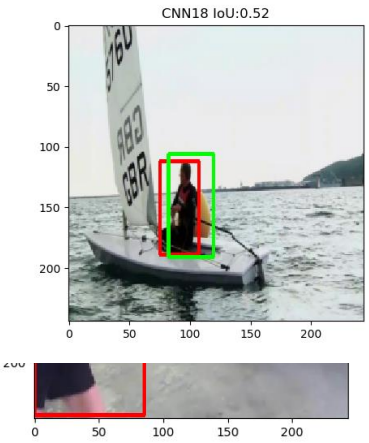

CNN18 IoU:0.71


Pretrained False IoU:0.91

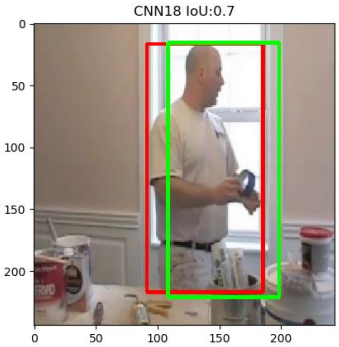

Pretrained True IoU:0.83

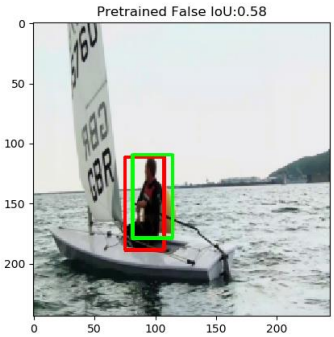

CNN18 IoU:0.81

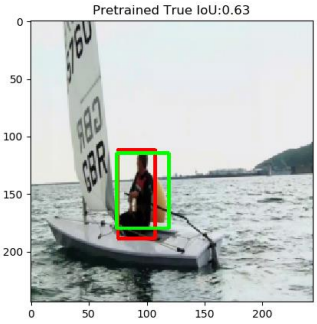

Pretrained True IoU:0.33

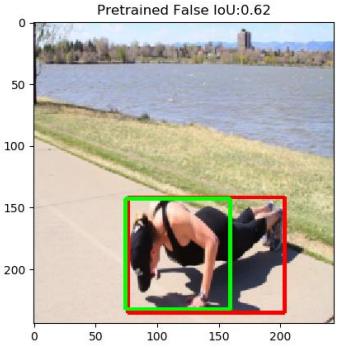

CNN18 IoU:0.52

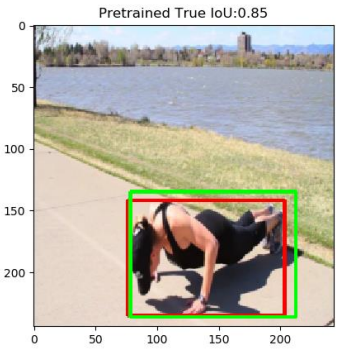

Pretrained False IoU:0.33

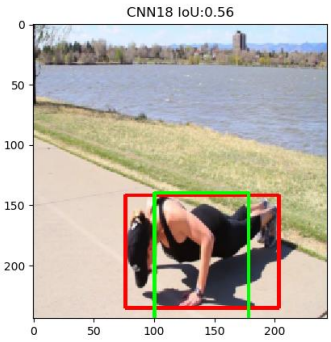

CNN18 IoU:0.7

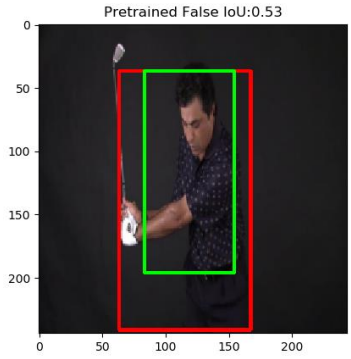

Pretrained False IoU:0.58

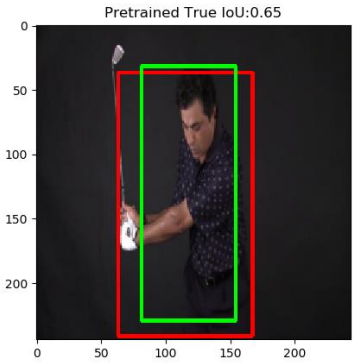

Pretrained True IoU:0.63

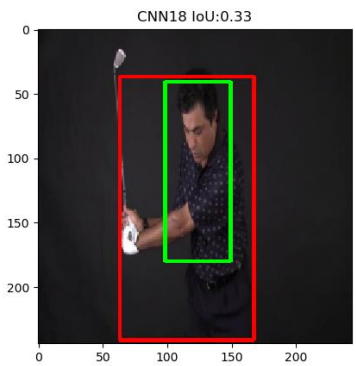

Pretrained False IoU:0.62

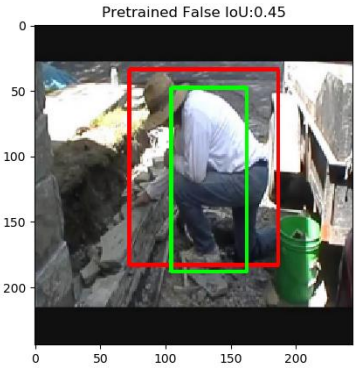

Pretrained True IoU:0.85

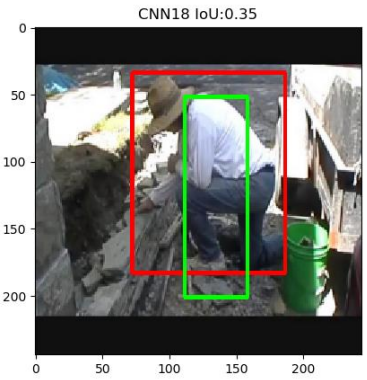

CNN18 IoU:0.56











Test results of 3 models are as follows:

	Random Weights ResNet18	Pretrained Weights ResNet18	18 Layer CNN
Test IoU Rate	60%	70%	56%

5. Discussion

According to the IoU results on the test images, we can see that the best model for this body detection problem is the pretrained ResNet model. It can fully cover all part of the body without omitting arms and legs. The main reason of that is the pretrained weights in the ResNet model that is optimized for ImageNet dataset. We trained weights which are already trained for a similar image processing problem, therefore, we obtained the best results in pretrained ResNet model. The worst results among these three models is the results of 18-layered CNN model. The performance of 18-layered CNN is not poor but it is not as good as the others. The main reason of that is the identity mapping in the ResNet models. It improves the performance of the network and gives better results than the plain network. Identity mapping allows residual networks to identify details in the picture like arms and legs. Therefore residual networks can better approximate the bounding boxes that fully cover all the parts of the body.

6. Conclusions

In this project we have implemented 3 CNN models to locate the position of a human body in a given image. Best result was achieved by the pretrained ResNet18. This was expected because ResNet was trained on ImageNet dataset which is much larger. The model can easily optimize the weights for a similar job. We have learned that using identity mapping increases the performance significantly and it is a good solution to diminishing gradients problem. Although, our 18 layer CNN model performs well in finding the general location of the body, it is not as successful in fully framing the body. Since we are only detecting the location of the body and not classifying, the complexity of 18 layer was enough. As further improvements, more layers can be used if classification feature will be added. Also, more than one body can be detected by adding one more output node that shows the probability of detection of candidate detection.

References

[1] Y. Cai and X. Tan, “Weakly supervised human body detection under arbitrary poses,” in International Conference on Image Processing. IEEE, 2016.

[2] Loss Functions, Accessed on 10.5.2020, 04:50 AM. [Online] Available: https://pytorch.org/docs/stable/nn.html#loss-functions

[3] K. He, X. Zhang, S. Ren, and J. Sun, “[1512.03385] Deep Residual Learning for Image Recognition.” [Online]. Available: https://arxiv.org/abs/1512.03385. [Accessed: 10-May-2020].

[4] wikipedia.org . *“Jaccard Index”.* Accessed on 10.5.2020, 04:50 AM. [Online] Available: https://en.wikipedia.org/wiki/Jaccard_index

APPENDIX

Personal Contribution

Muhammed Cemil Şişman: CNN-18 model design

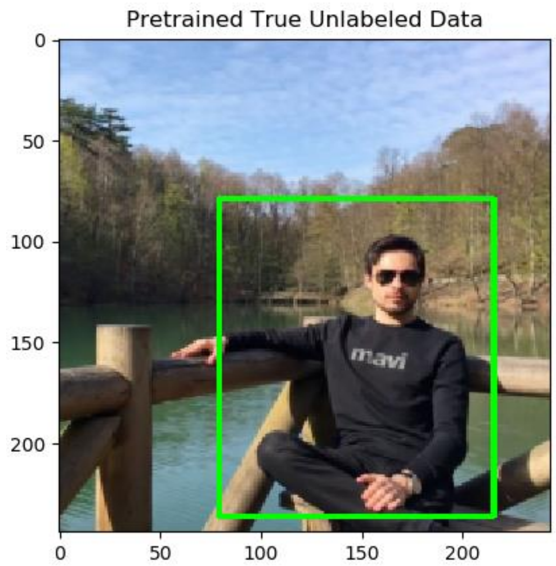
Yusuf Furkan Salcan: Preprocessing, dataset and ResNet model implementation, testing

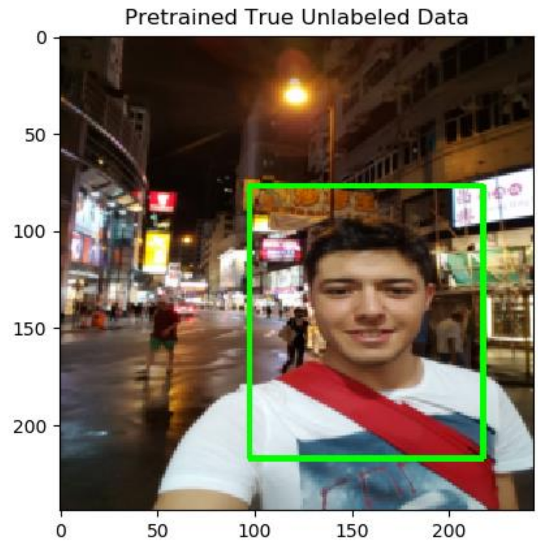
Ahmet Çelebi Kocadağ: Validation and training

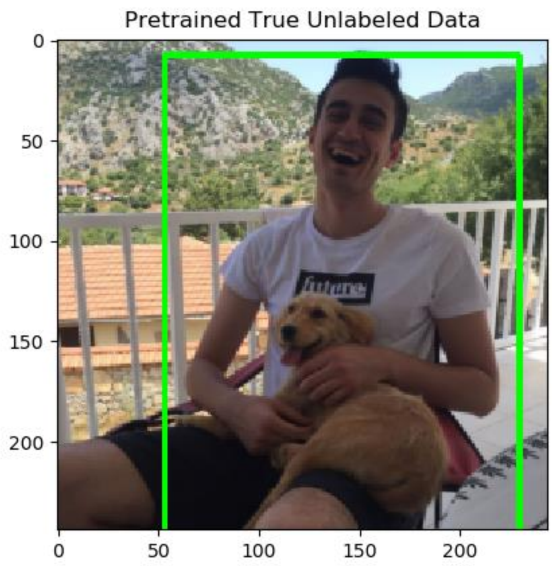
Ömer Aysal Aytaç: CNN-18 model design

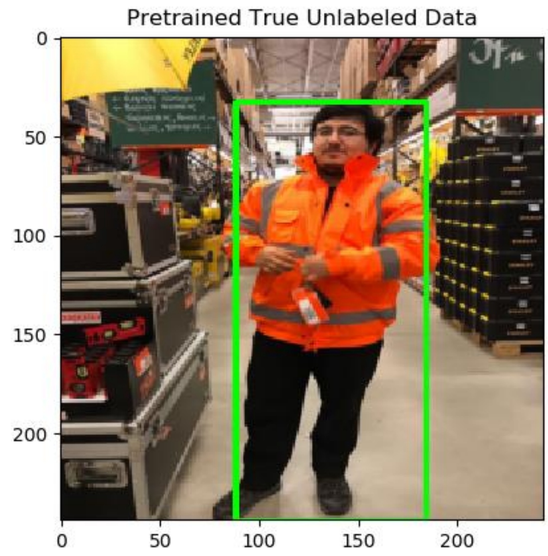
Semih Erdoğan: Validation and training

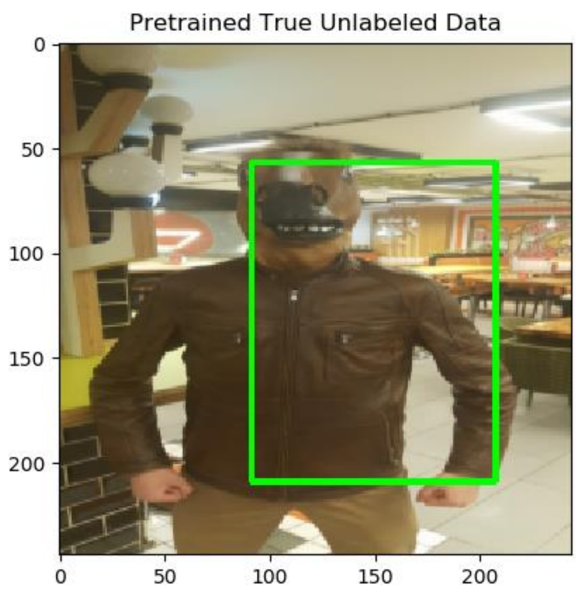
Additional results on unlabeled data by using the best performed model:

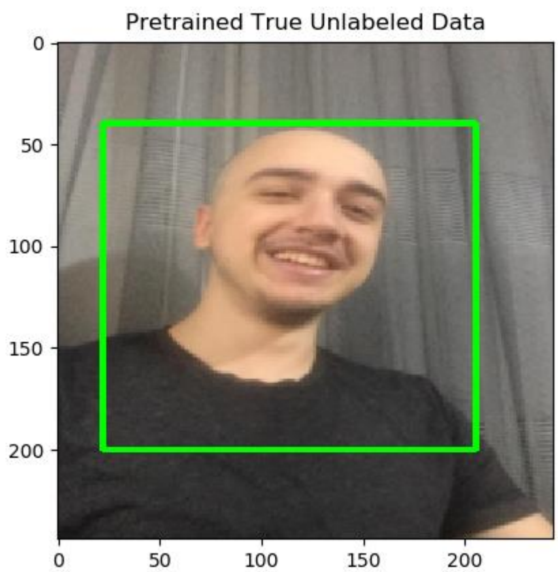

Pretrained True Unlabeled Data


Pretrained True Unlabeled Data


Pretrained True Unlabeled Data


Pretrained True Unlabeled Data


Pretrained True Unlabeled Data


Pretrained True Unlabeled Data