Introduction

The following presentation has the aim to describe a project which goal is to detect and segment human hands in a given image providing two different outputs. Detection consisting in a box for each detected hand in the image and segmentation introducing a coloured mask involving pixels related to a hand. To achieve this two goals the project includes a part in python to define the deep neural network and another in C++ to upload the deep neural network and to work with pixels in the image in order to get an appropriate detection and segmentation. The presentation is divided in chapters describing each file and the related functions.

Chapter 1

DNN

We decided to use a deep neural network approach to develop the detection part to achieve better performance. We have chosen the YOLOV5 model based on darknet. We developed all the network in python in Google Colab. The model was downloaded from the original GitHub of YOLO.

1.1 YOLO

YOLO an acronym for 'You only look once', is an object detection algorithm that divides images into a grid system. Each cell in the grid is responsible for detecting objects within itself. YOLO is one of the most famous object detection algorithms due to its speed and accuracy. We went for the version 5 cause it's new and it achieves higher results than other versions.

1.2 Dataset

The datased used to train the network was composed by "EgoHands" and "HandOverFace" mixed together. Before that we removed images included in the test dataset to follow the rule of not using images of test set in the training/valid set. As validation we uploaded a dataset containing 150 images selected randomly from internet and from my computer. The images were manually labelled with the help of Roboflow.

1.3 Training

We trained the network using the datasets talked below. The configuration was 150 epochs with a batch size of 16 and the pretrained model was yolov5m, so the medium version with 369 layers, 20871318 parameters. During the training we were checking the map every 3 epochs to avoid overfitting. Fortunately it didn't overfit and it reached a good and stable point. In fact in the last 15 epochs the mAP was pretty stable at mAP.5 at 99.4 and mAP.5-.95 was near 83.

1.4 Hyperparameters

learning-rate = 0.01 , final-learning-rate = 0.01, momentum = 0.937, weight-decay = 0.0005, warmup-epochs = 3.0, warmup-momentum = 0.8, warmup-bias-learning rate = 0.1, IoU-threshold = 0.2

1.5 Test and Export

We tested our model with the test set that was given and we achieved about 95 percent overall. To export the network we used the Onnx extension.

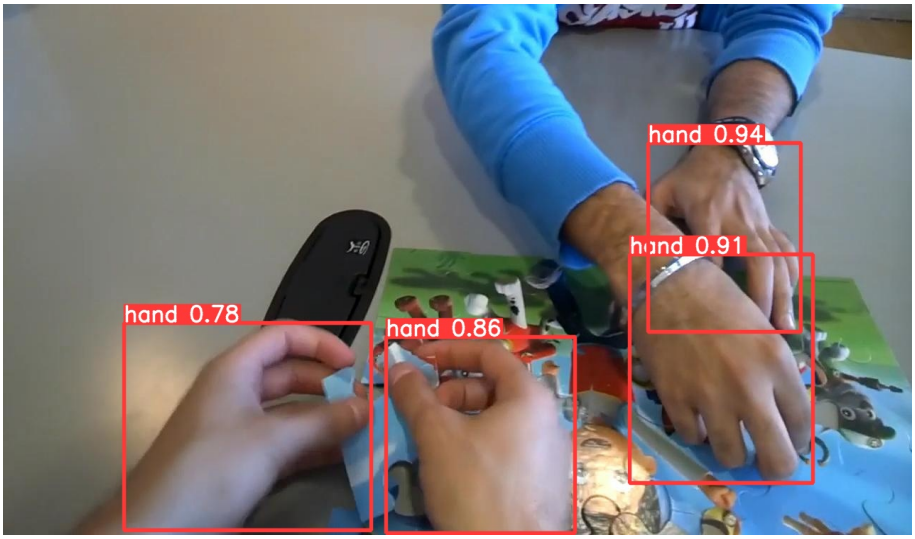

Figure 1.1: Result of Python Yolov5 model(numbers rapresents the confidence of the hand)

Figure 1.2: Python Yolov5

Chapter 2

Detection

Detection is the first module that we need to develop. In this module we will load our network, the input images and we detect the object drawing a box. Then we read the GroundTruth boxes and calculate the IoU.

2.1 Load Net

The first step involves the load of the neural network described in the previous chapter by the so called function "loadnet". This function load the net and set the global parameters needed to make the program run.

2.2 Format Yolov5

Since the network accepts only square images with 3 channels we created a function to set the right format.So this function aims to transform the image in the right type and we filled the image with black where there was no points.

2.3 Detect

The "Detect" function uses as inputs image, the net and an output vector where it saves the results. The first step involves the implementation of "format yolov5" to make the images fit with the model. Then we computed the scale factors to recenter the boxes cause the model resize the images in 640x640. After that the model is ready to go, so we make the model work.The output was a matrix of 25200 rows and 6 columns. Each row is a detection of our model. In each row the "data" contains its outputs. Data is 6 elements: x, y, width, height, confidence, labelScore. So in practice we have in the firsts 4 positions the box, in 5th position the confidence of the detection and then a list of scores for each class, in our case there was only hand class so there was only 1 score. So we cycled all the detection to save a box only if it

satisfies a chosen confidence. Since some boxes overlap each other, we needed to use a function called non-maxima suppression to apply only if two or more boxes contain pixels of the same hand. This pass was necessary to remove small boxes detected inside a bigger box.

2.4 Main

In the main we just read all images and feed them into the model to get the boxes. Then we draw a rectangle for each box with a different color.After that we read the txt files containing the real box. We performed the IoU to get a value.

Figure 2.1: Hands detection for the first image of the dataset

Figure 2.2: Hands detection for the first image of the "HandsOverFace" dataset

Chapter 3

Metrics

In the following chapter are described functions used to evaluate the performances of our deep neural network and of the implemented segmentation.

3.1 IoU

To evaluate the detection we had to define an IoU function (Intersection over Union) which consists in the ratio between the intersection of the area defined by the achieved and the target boxes and their union. To do this we implemented the IoU such that it needs as input vectors containing the achieved and the target boxes as "Rect" variables. The first part consists in finding the minimum subpart of the image containing all boxes related to that particular image and then defining two different masks, one for the generated boxes and one for the target ones. The final operation involves the computation of the ratio.

3.2 Accuracy

For what concern segmentation process' performance is defined by an accuracy function which inputs are the ground truth segmentation mask, the computed one and a vector to save results. The accuracy has to be written in a way such that it has to consider two classes of hand and no-hand pixels.
Firstly, the function performs some checks about inputs' channels and their values related to rows and cols. Secondly, we computed the intersection and the union between the two binary masks in order to get the accuracy of the hand pixels. Their union value was used even in the computation of the non-hand pixels. Both accuracy values were saved into a vector of double values.

Chapter 4

HandSegmentation

In the following file we performed the segmentation of the images after drawing boxes containing hands. Some problems were encountered in the attempt to segment image from the dataset "HandOverFace" since the first code made where the segmentation was not restricted to the subimage contained in the rectangles was unable to implement the segmentation to the hands avoiding pixels belonging to faces or some parts of the background. That's why we decided to use a kind of semantic segmentation able to distinguish between the background and the foreground. Things in the foreground were considered or not considering based on the skin colour and the most defined contour. Indeed, inside the detected rectangle the code looks for the biggest contour and it supposes that it defines a hand. The other constraint it is to look for pixels having a colour in a predefined range of values compared to what there is inside the biggest contour.