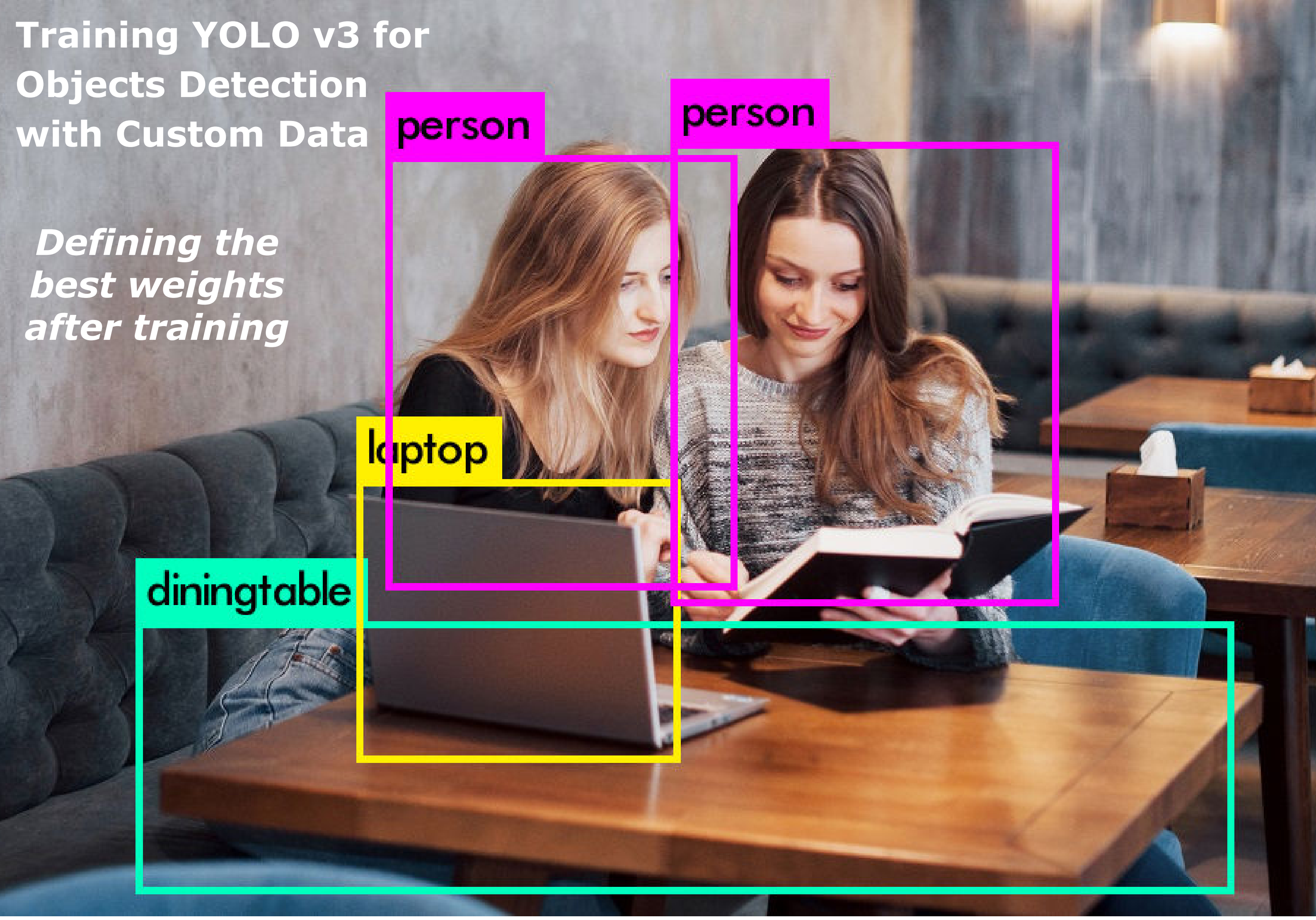
Training YOLO v3 for Objects Detection with Custom Data

Defining the best weights after training



When do we stop training?

Recommended number of iterations for training is defined by using following equation.

*max_batches = classes * 2000*
(but not less than 4000 in total)

It means that every class should have around *2000 iterations*. For example, if in our custom dataset of *Traffic Signs*, we have four classes, then recommended total number of iterations for training on this particular dataset will be *8000*.

How to define the best weights after training

Weights will be saved every *100 iterations*. Our task now is to define the best one to use for detection making sure that there is *no overfitting*.

```
darket/
    backup/
        yolo-obj_last.weights
         ...
        yolo-obj_1000.weights
         ...
        yolo-obj_2000.weights
         ...
        yolo-obj_final.weights
```

Algorithm is as following: start checking saved and trained weights from the end one by one calculating *mean average precision (mAP)*. The goal is to find *weights* that have the biggest *mAP.*

Find the biggest mAP

yolo-obj_8000.weights

yolo-obj_7000.weights

yolo-obj_6000.weights

For example, if we consider *Traffic Signs* dataset, and if *mAP* for *7000 iterations* is bigger than for *8000*, then it is needed to check *weights* for *6000 iterations*. Next, if *mAP* for *6000 iterations* is already less than for *7000 iterations*, then you can stop checking and use weights for *7000 iterations* in detection tasks.

Also, it is possible to continue checking *weights* between *6000* and *7000 iterations*, trying to find *weights* even with bigger *mAP*.

How to calculate *mAP* for trained weights

There is special command in *Darknet framework* that calculates *mAP*. To start calculating process of *mAP* in *Darknet framework* for particular weights, navigate to the directory with executable file and type in *Terminal* or *command line* specific command as described below.

Calculating *mAP* for Traffic Signs dataset

- **For Linux and MacOS** navigate to root directory where *Darknet framework* was installed and type in following command:
```
./darknet detector map cfg/ts_data.data cfg/yolov3_ts_train.cfg backup/yolo-obj_8000.weights
```

- **For Windows** navigate to *darknet\build\darknet\x64* and type in following command:
```
darknet.exe detector map cfg\ts_data.data cfg\yolov3_ts_train.cfg backup\yolo-obj_8000.weights
```

Calculating *mAP* for custom dataset with Car, Bicycle wheel and Bus

- **For Linux and MacOS** navigate to root directory where *Darknet framework* was installed and type in following command:
```
./darknet detector map cfg/custom_data.data cfg/yolov3_custom_train.cfg backup/yolo-obj_6000.weights
```

- **For Windows** navigate to *darknet\build\darknet\x64* and type in following command:
```
darknet.exe detector map cfg\custom_data.data cfg\yolov3_custom_train.cfg backup\yolo-obj_6000.weights
```

After calculation, verbose information will be provided. Find calculated *mAP* at the last lines. *Take notes* that for this trained weights *mAP* is, for example, 55%. Continue checking in order to define the weights with biggest *mAP*.

What is overfitting?

Overfitting happens when after training model can detect almost 100% of objects on *training dataset* and almost 0% on *validation dataset* or any other real-life images.

It can be represented graphically, when from start point *loss* is decreasing both for *training dataset* and *validation dataset*. But after some point, loss for *validation dataset* stop decreasing and starts to increase. At the same time, loss for *training dataset* continues decreasing reaching almost 0, that is no mistakes and almost 100% of correct detections.

The goal is to define this particular point where the loss for *validation dataset* is minimum.

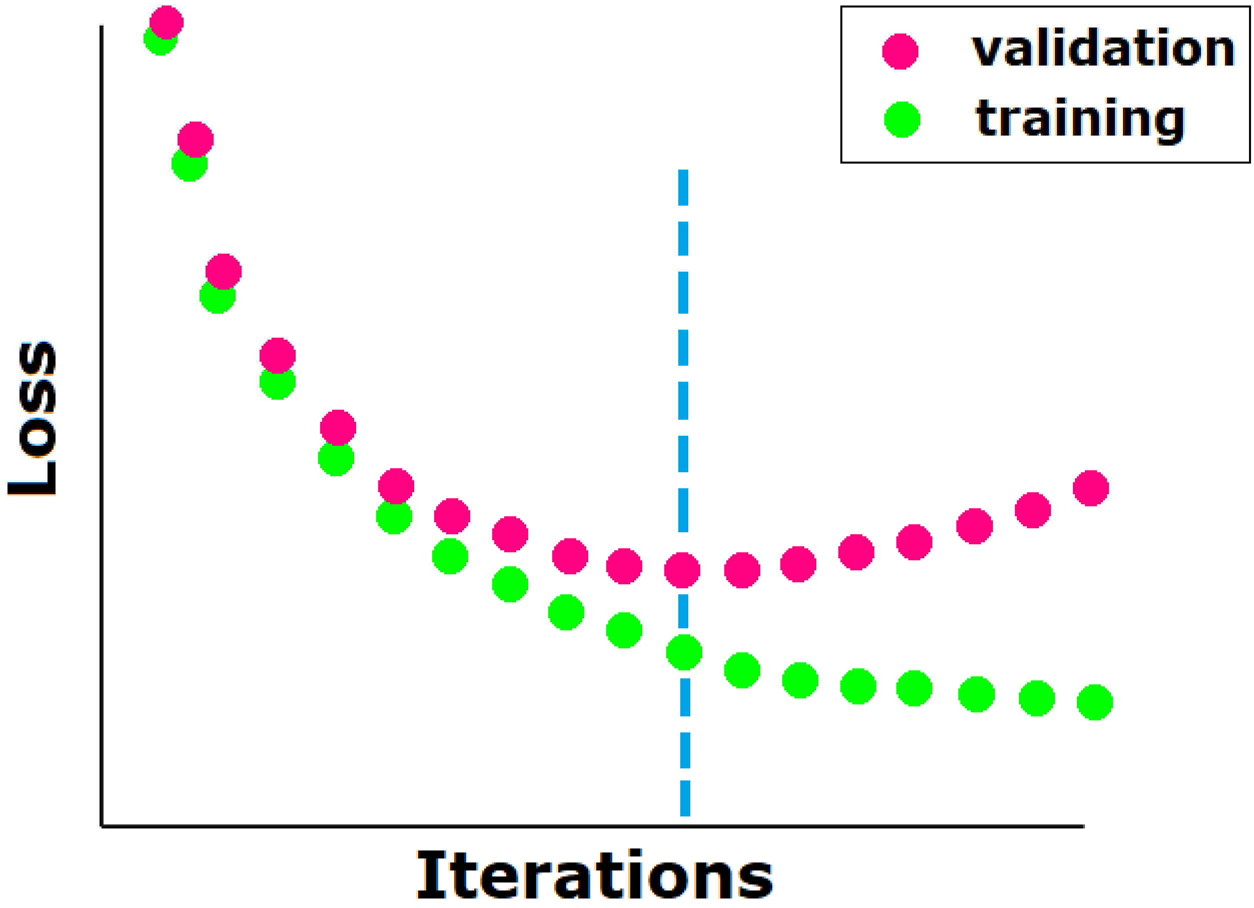


Figure 1. Loss for validation and training datasets during training process

What is *mAP* for Objects Detection tasks?

mAP (mean average precision) is a *metric* used to evaluate accuracy, in our case, for *Objects Detection* tasks.

In general, to calculate *mAP* for a custom model that is trained for *Objects Detection* tasks, firstly, *Average Precision* is calculated for every class in the custom model. Then, mean of these calculated *Average Precisions* across all classes gives *mAP.*

Pay attention! Some papers use *Average Precision* and *mAP* interchangeably.

Important terminology

Understanding the calculation process of *Average Precision* needs to update knowledge of definitions for used parameters.

Threshold

Threshold is used to identify whether prediction of *Bounding Box (BB)* can be considered as *True* or *False*. Usually threshold is set to one of the following: *50%*, *75%*, *95%*.

Intersection Over Union (IoU)

IoU is a measure that is used to evaluate *overlap* between two *Bounding Boxes (BB)*. *IoU* shows how much predicted *BB* overlaps with so called *Ground Truth BB* (the one that has real object inside). Comparing *IoU* with threshold it is possible to define whether predicted *BB* is True Positive (valid in other words) or False Positive (not valid).

IoU is calculated by overlapping area between predicted *BB* and *Ground Truth BB* divided by union area of two *BB* as shown on the Fig.2.

$$IoU = \frac{area\ of\ overlap}{area\ of\ union} =$$

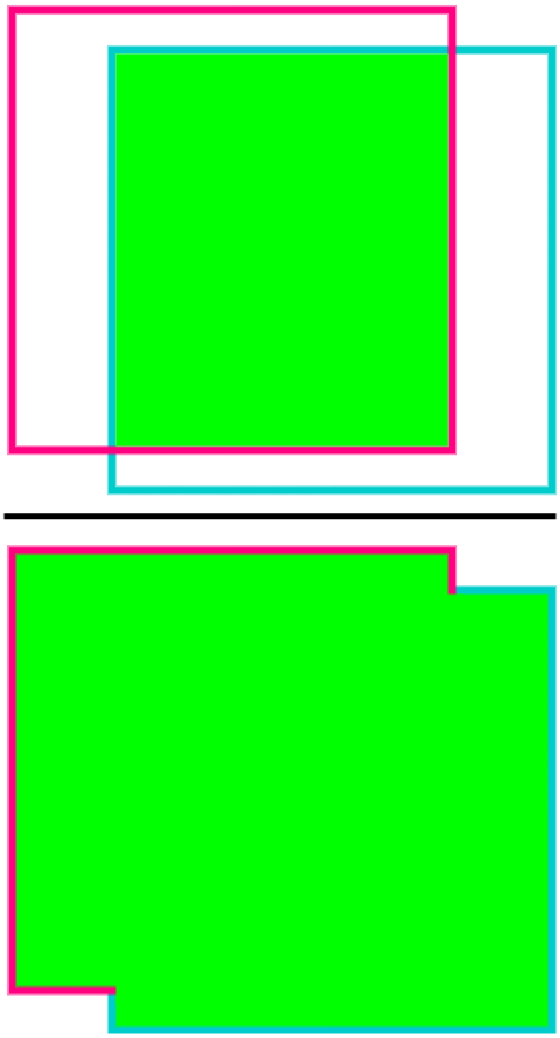


Figure 2. IoU between Ground Truth BB and predicted BB

True Positive, False Positive, False Negative, True Negative
Following concepts are used in order to calculate and understand metrics:

- *True Positive (TP)* → is a number of *BB* with correct predictions, *IoU* ≥ threshold
- *False Positive (FP)* → is a number of *BB* with wrong predictions, *IoU* < threshold
- *False Negative (FN)* → is a number of *Ground Truth BB* that are not detected
- *True Negative (TN)* → is a number of *BB* that are correctly not predicted (as many as possible within an image but not overlap any *Ground Truth BB*); this parameter is not used for calculating metrics

Precision
Precision represents percentage of correct positive predictions of *BB* (how accurate are predicted *BB*) and shows an ability of the trained model to detect relevant objects. *Precision* is calculated as following:

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{all\ detections}$$

Recall
Recall represents percentage of *True Positive* predictions of *BB* among all relevant *Ground Truth BB* and shows an ability of the trained model to detect all *Ground Truth BB*. *Recall* is calculated as following:

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{all\ Ground\ Truth}$$

Precision and Recall curve
Precision and Recall curve represents performance of the trained model by plotting a curve of *Precisions* values against *Recalls* values and form a kind of *zig-zag graph* as shown on Fig.3 below.

In order to plot *Precision and Recall curve*, it is needed to collect detected *BB* by their confidences in descending order. Then, calculate *Precision* and *Recall* for every detected *BB* as it is shown in Table 1 below. In current example, *threshold* is set to *50%* saying that predicted *BB* is correct if *IoU ≥ 0.5*. Total number of correct predictions *TP = 5* and total number of wrong predictions *FP = 5*.

Table 1. Collecting predicted BB in descending order according to their confidences and calculating Precision and Recall

BB	Confidence	TP or FP	Precision	Recall
1	96%	TP	1/1 = 1	1/5 = 0.2
2	94%	FP	1/2 = 0.5	1/5 = 0.2
3	90%	TP	2/3 = 0.67	2/5 = 0.4
4	89%	TP	3/4 = 0.75	3/5 = 0.6
5	81%	FP	3/5 = 0.6	3/5 = 0.6
6	75%	TP	4/6 = 0.67	4/5 = 0.8
7	63%	TP	5/7 = 0.71	5/5 = 1
8	59%	FP	5/8 = 0.62	5/5 = 1
9	54%	FP	5/9 = 0.56	5/5 = 1
10	51%	FP	5/10 = 0.5	5/5 = 1

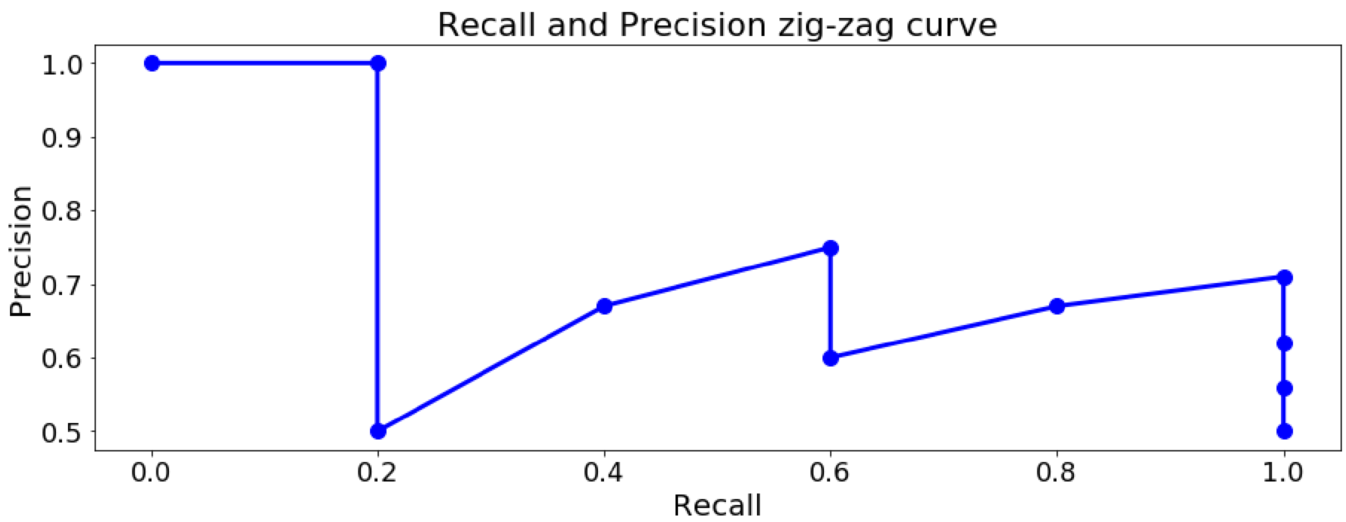


Figure 3. Calculated Precision and Recall zig-zag curve

Calculating Average Precision (AP)

AP is calculated by considering area under *Interpolated Precision and Recall curve*. Firstly, *Recall* values are divided into 11 points as following: [0, 0.1, 0,2 … 1] as shown on Fig.4 below. Then, average of maximum precision values is computed for these 11 *Recall* points.

$$\mathrm{AP} = \frac{1}{11} \sum_{r\,\epsilon\{0,0.1,0.2\ldots1\}} p_{interpolated}\,(r),$$

$p_{interpolated(r)} = \max p(\tilde{r}),$ $p(\tilde{r})$ → is evaluated *Precision* at *Recall* $\tilde{r}$.

From our example, *AP* will be calculated as following:

$$\mathrm{AP} = \frac{1}{11}(1 + 1 + 1 + 0.75 + 0.75 + 0.75 + 0.75 + 0.71 + 0.71 + 0.71 + 0.71) = 0.81$$

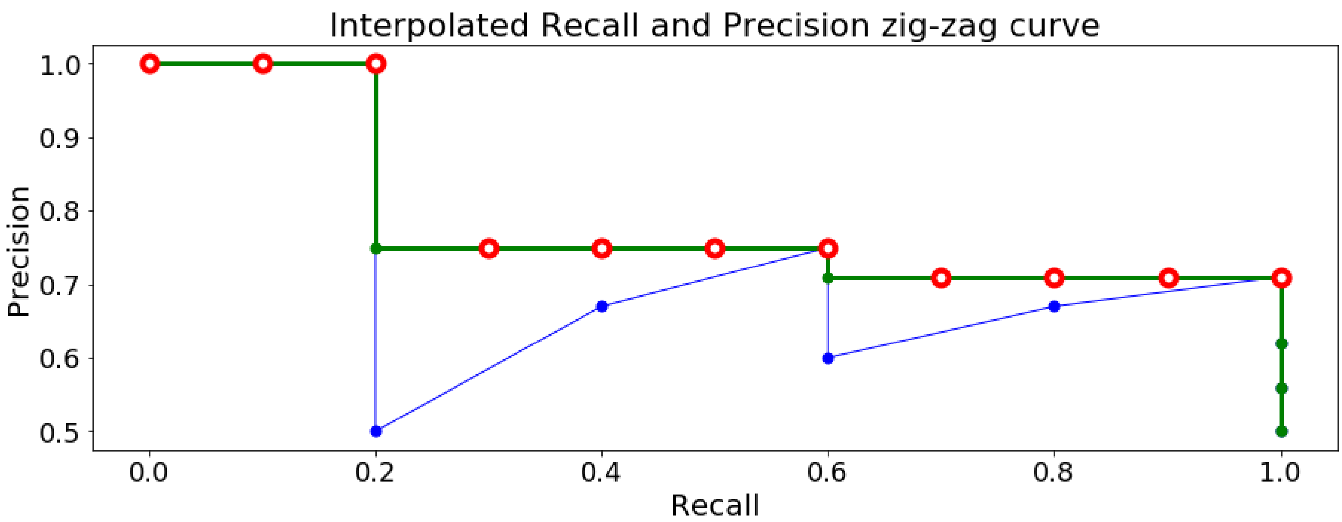


Figure 4. Interpolated Precision and Recall zig-zag curve and 11 points

Useful Links

Check out additional links with detailed explanation of *metrics* used for *Objects Detection* tasks and other useful information for further reading:

[1] Metrics for Object Detection – repository, that describes in details and in easy-to-understand manner *metrics* used for evaluation results of *Objects Detection* tasks

[2] Evaluation of multi-class detections on VOC2007 – paper with description of evaluation process used on PASCALVOC competition (page 11)