Dog detection

Neural network course project

Jakub Zadrożny, Mateusz Hazy, Kamil Szubiński

February 14, 2019

Abstract

Object detection is a very rapidly evolving deep learning area and a wide variety of methods has been developed in recent years. Many of them suffer from a computation complexity (RCNN). In this paper we revisit YOLO [1] and YOLO v2 (introduced in YOLO9000 paper [2]) - fast real time object detection methods. Both of them are able to detect many classes of objects, however, we limit ourselves to detecting only one class - a dog. We then compare both versions of YOLO to seek for any improvement.

YOLO

The YOLO method states detection as a linear regression problem straight from pixels to bounding boxes coordinates and class probabilities. The system divides image into $S \times S$ grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell can predict B bounding boxes. Each bounding box consists of five predictions: x, y, w, h and confidence, where the first four describe the bounding box location and size. The last parameter describes how confident the model is that the bounding box contains the object. Overall the YOLO architecture takes the image, passes it through convolutional and fully connected layers and produces $S \times S \times (B \cdot 5 + C)$ predictions. Each $B \cdot 5 + C$ segment represents the predictions for the grid cell. Five parameters described above for each bounding box and C class probabilities. The original paper [1] provides more detailed description.

YOLO v2

The YOLO v2 is in fact the basic YOLO with several improvements. For example, the use of batch normalization on all convolutional layers in YOLO not

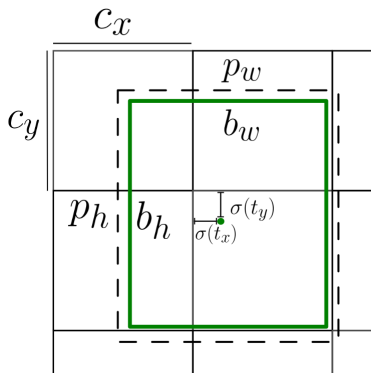


Figure 1: Direct location prediction.

only eliminates the need for other forms of regularisation (in theory), but also results in 2% improvement in mAP. Further changes include higher resolution classifier (denser grid) and direct location prediction.
The last improvement is very useful for dealing with initial instability. In region proposal networks the network predicts values t_x and t_y and the (x, y) center coordinates are calculated as

$$\begin{aligned} x &:= (t_x \cdot w_a) - x_a \\ y &:= (t_y \cdot h_a) - y_a \end{aligned} \tag{1}$$

Since this formulation is unconstrained, any anchor box can end up at any point in the image. To overcome this drawback, we bound the location using logistic activation σ, which makes the value fall between 0 to 1:

$$\begin{aligned} b_x &:= \sigma(t_x) + c_x \\ b_y &:= \sigma(t_y) + c_y \\ b_w &:= p_w \cdot e^{t_w} \\ b_h &:= p_h \cdot e^{t_h} \end{aligned} \tag{2}$$

according to **Figure 1**.
This technique results in 5% increase in mAP over the version of anchor boxes. More detailed description is available in the paper.

Dataset

We decided to use a subset of Youtube-BoundingBoxes Dataset [3]. One can find there more than 200000 frames with dogs. The biggest disadvantage of this dataset is that only one object is detected in every frame. This can lead to some troubles with learning the model. The other problem is that some bounding boxes are set incorrectly. However, this can reduce overfitting during learning.

Furthermore, we employed data augumentation, namely random translations (up to 25 px) and random rotations (up to 25 degrees).

Figure 2: Image with many dogs, where only one is detected.

Figure 3: Image with wrong bounding box.

Training

We took 8000 of the images of dogs from Youtube-BoundingBoxes Dataset and did traditional train/test split. Then we used Adam as an optimizer.

We optimized the weighted mean least squares error, as suggested in the paper [1]

The error function was defined as:

$$1 - IOU(predicted\_box, true\_box)$$

where

$$IOU(A, B) = \frac{A \cap B}{A \cup B}$$

YOLO training

As a feature extractor we tried to use pretrained vgg and resnet50 networks (documented in [4]). On top of that we added 2 fully connected layers and ReLU for nonlinearity. For regularization we used dropout with $p = 0.5$.

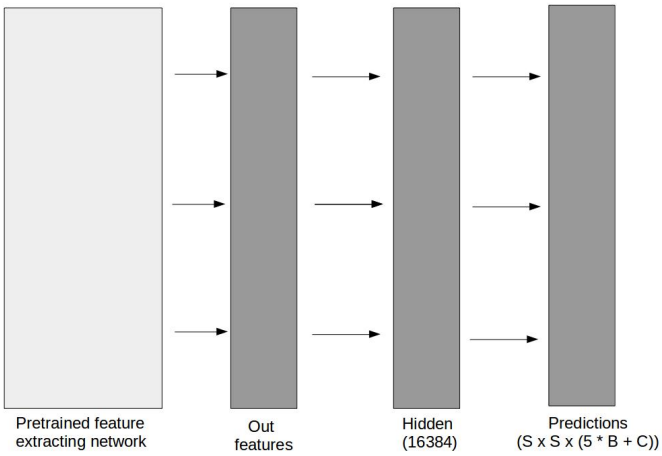


Figure 4: YOLO architecture

We managed to train the network down to 25% train error rate. However the test error rate has never been smaller than 65%. We tried different configurations of hyperparameters, adding noise to input values and data augmentation, yet it did not improve the results. We decided to abandon the YOLO approach and move to the YOLO 9000.

YOLO 2 training

To get good anchor boxes, we ran K-means on bounding boxes from the dataset. However the closest centroid was chosen not by euclidean distance but by the best IOU score.

As a feature extractor we used again resnet50, but instead of fully connected layers, we added 4 convolutional layers on top of it. Each convolutional layer was followed by batch normalisation and ReLU.

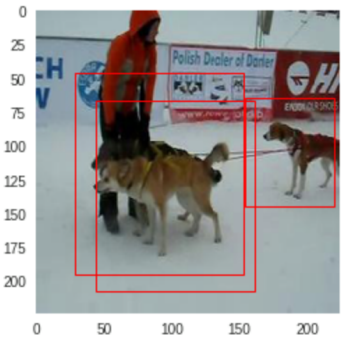
As a result we managed to reduce test error rate down to 48% which we found satisfying.

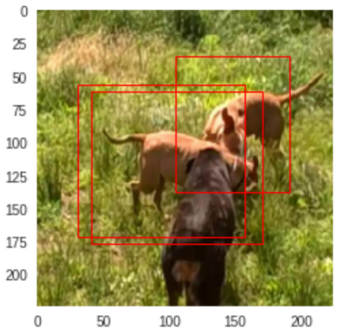
Conclusion

Although we did not achieve spectacular results, there are some interesting outcomes we noticed.

Our YOLO network had a problem with overfitting. We used dropout and weight decay, but results were not satisfactory enough. YOLOv2 uses batch norm and smaller weight decay, which resulted in much better performance.

Even though we trained the model on the dataset with only on dog per image, it is quite good at finding multiple dogs. If there are two or three dogs on the picture, then top 3 predictions find them. We present two examples below.





We consider the failure with YOLO as our mistake. Maybe tuning the model could improve the results. However, it might have been the problem with the dataset quality as well.

Using the anchor boxes int YOLO V2 significantly improves the learning. Intuitively, there are B boxes in each grid cell, because we want to encourage each box to specialize to detect specific type of shape. If the model has good priors of these types, it only has to adjust the boxes, instead of learning from scratch. Moreover, it is worth mentioning that our network can work and learn on images of any size.

References

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi: You Only Look Once: Unified, Real-Time Object Detection `https://arxiv.org/pdf/1506.02640v5.pdf`

[2] Joseph Redmon, Ali Farhadi: YOLO9000:Better, Faster, Stronger `https://arxiv.org/pdf/1612.08242v1.pdf`

[3] `https://research.google.com/youtube-bb/`

[4] `https://pytorch.org/docs/stable/torchvision/models.html`