YOLO final-report

김태훈[1]
[1]부산대학교 전기컴퓨터공학부.

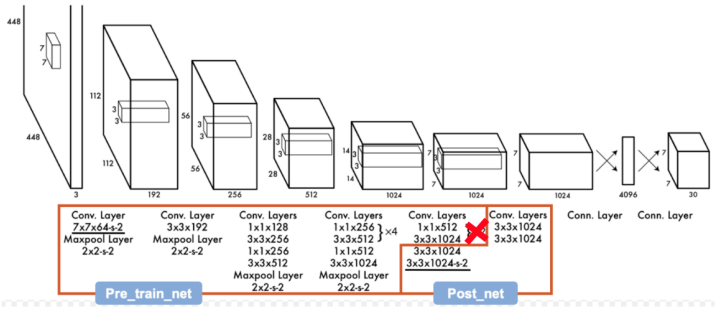

Figure 1: Architecture of the YOLO model. Our network has 22 convolutional layers followed by 2 fully connected layers. 1×1 convolutional layers reduce the features space

1 Introduction

YOLO[4] is the object detection model that detect object using a single neural-network regression of spatially separated bounding boxes and associated class probabilities. the paper[4] says YOLO is extremly fast(45 frames per second) while achieving double the mAP of other real-time object detection like Fastest DPM[6].

In this report, We use Pytorch to implement YOLO, we use PASCAL VOC 2007 and 2012[1] dataset to training the model, and we will analyze and discuss the training and inference results.

2 Dataset

As mentioned before, we use PASCAL VOC 2007 and 2012[1] dataset for training, and PASCAL VOC 2007test dataset for testing. there are 4300 images for training, and 1227 for testing. there are 6 categories(aeroplane, bicycle, bus, car, cat, dog) in the dataset. the Labeling data are consist of objects category and coordinate(xmin, ymin, xmax, ymax)

2.1 Pre-processing

We increase the image size from 224×224 to 448×448 same as the paper.[4]

For data augmentation, Each image was horizontal flipped, resized(0.8 to 1.2), shifted, cropped with a 50% chance. Also, changing brightness, saturation, hue of the image and blurring it were processed with a 50% chance. Labeling data are also changed when corresponding image was changed.

3 Network Architecture

Our network has 22 convolution layers followed by 2 Fully-connected layers, which is slightly different from the paper[4]'s architecture that has 24 convolution layers followed by 2 fully-connected layers. Our network has 22 Convolutional layers, 4 Max Pooling layers, 2 Fully connected layers. The activation function of this model is Leaky ReLU with 0.1 negative slope. The full network is shown in Figure 1.

The output shape of the model is $7 \times \ 7 \times 30$. the 7×7 means the input image is divided into an 7×7 grid of cells. and 30 means Each grid cell predict 2 bounding boxes and 20 classes probabilities. Each bounding box consists of x,y, width, height, and confidence score. (x,y) coordinates mean the center of the box relative to the top-left bounds of the grid cell. The width and height are relative to the whole image.

The target confidence score is zero if there are no objects in that cell. If the cell contains objects, the confidence score is equal to the intersection over union between predicted box and the ground truth.

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2
\end{aligned}
$$

Figure 2: Loss function of the YOLO model.

the YOLO and YOLO vanila model has about 266M parameters, and YOLO VGG-16 has about 260M parameters.

4 Loss function

The loss function of the model is shown in Figure 2, where $\mathbf{1}_i^{obj}$ denotes whether object appears in cell and $\mathbf{1}_{ij}^{obj}$ denotes that the jth bounding box predictor in cell i is responsible for that prediction. the "responsible" bounding boxes has highest IOU with the ground truth. Also, λ_{coord} is balancing parameter for increasing the loss from bounding box coordinate than classification, and λ_{noobj} is for decreasing loss from confidence score of grid that don't contain objects. The error function is sum-squared error.

The meaning of each term is as follows:

1. For the responsible bounding box predictor j in cell i that containing object, get loss of coordinate of center of the bounding box.
2. For the responsible bounding box predictor j in cell i that containing object, get loss of square root of width and height
3. For the responsible bounding box predictor j in cell i that containing object, get loss of confidence score ($C_i = 1$)
4. For the responsible bounding box predictor j in cell i that don't contain object, get loss of confidence score ($C_i = 0$)
5. For cell that contains object, get loss of class probabilities.

The above loss functions are in the paper[4]. however, Our loss function is little different from the paper[4]. First, We get loss for the bounding box predictor j in cell i that don't contain object, not "responsible" bounding box. Also, We use mean-squared error for YOLO in Table 1 because the loss exploded to nan after applying sum-squared error and pre-train weights.

5 Training

We set two parameters $\lambda_{coord} = 5$ and $\lambda_{noobj} = 0.5$. We train 3 slightly different model for comparison. explanation of the each model is shown in Table 1

We train the networks for 15 epochs with 10 batch size, 10^{-4} learning rate and 5×10^{-4} weight decay. Learning rate is fixed to 10^{-4} during training.

model name	pre-trained	error
YOLO	all layers are pre-trained with 68.5mAP performance weights	MSE
YOLO VGG-16	first 18 convolution layers are changed to VGG-16[5] pre-trained with Imagenet	SSE
YOLO vanila	all layers are not pre-trained	SSE

Table 1: The explanation of the models. SSE means Sum-Squared error, MSE means Mean-Squared error

	mAP	aero	bike	bus	car	cat	dog
YOLO	62.6	59.4	62.0	52.0	61.4	69.7	71.3
YOLO VGG-16	30.0	36.6	40.4	0.0	34.5	38.2	30.4
YOLO vanila	4.4	6.4	0.0	0.0	2.0	0.3	17.6

Table 2: The average precision of the models after training. Each value is rounded off to the first decimal place. The test dataset is PASCAL VOC 2007test.

6 Experiment

After training, We test the models with PASCAL VOC 2007test dataset. The results of each model are shown in Table 2. the mAP of the YOLO model is 62.6, YOLO VGG-16 is 30.0, YOLO vanila is 4.4.

6.1 Discussion

The mAP of the YOLO model is less than pre-trained model. this may be because the construction of the loss function is wrong and applying mean-squared error function. We tried to apply pre-trained weights and sum-squared error same as the paper[4], but we struggle to apply it because the loss is exploded to nan. So We apply MSE error function as stopgap. We will find a solution after the report. but the mAP is 62.6, which is better than other real-time object detection model like DPM[3].

the mAP of YOLO VGG-16 is 30.0, which significantly lower than the YOLO VGG-16 of the paper[4]. this may be because 15 epochs is too smaller than 135 epochs of the paper[4].

the mAP of YOLO vanila is the lowest among the models. this is because all layers are not pre-trained. 4300 images is insufficient to train all parameters in the model. The paper[4] pretrained first 20 convolution layers with Imagenet dataset.

7 Conclusion

We implemented YOLO, object detection model using regression with single neural network. We confirm that the YOLO has quite good performance despite it uses single neural network Unlike other object detection model like Fast R-CNN[2]. If we solve the nan loss problem, we will be able to confirm better on this model.

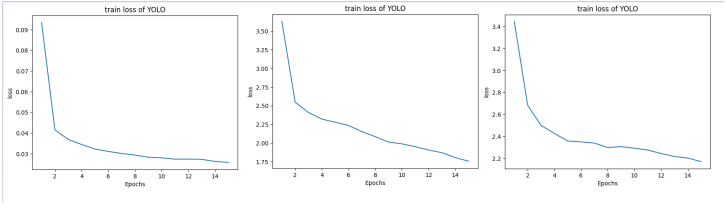

Figure 3: Training loss of the each model. The left is YOLO, the middle is YOLO VGG-16, and right is YOLO vanila.

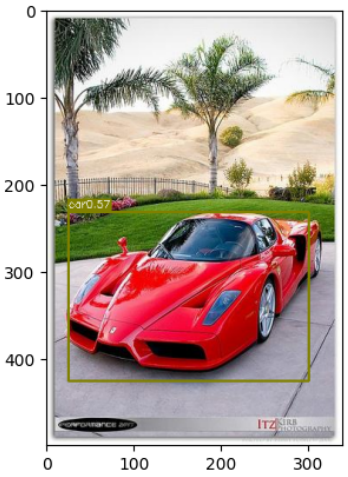

Figure 4: Visualizing result of the YOLO model using a car image.

[1] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[2] Ross Girshick. Fast r-cnn, 2015. URL https://arxiv.org/abs/1504.08083.

[3] Ross Girshick, Forrest Iandola, Trevor Darrell, and Jitendra Malik. Deformable part models are convolutional neural networks, 2014. URL https://arxiv.org/abs/1409.5403.

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016. URL https://arxiv.org/abs/1506.02640.

[5] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. URL https://arxiv.org/abs/1409.1556.

[6] Junjie Yan, Zhen Lei, Longyin Wen, and Stan Z. Li. The fastest deformable part model for object detection. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2497–2504, 2014. doi: 10.1109/CVPR.2014.320.