Paper review: You Only Look Once

Basic about YOLO

-YOLO, a unified pipeline for object detection\

-Prior work on object detection repurposes classifiers to perform detection

-Instead, YOLO frames object detection as a regression problem to spatially separated bounding boxes and associated probabilities.

-A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation.

-Since the whole detection pipeline is a single network, it can be optimized end to end directly on detection performance.

Limitations of the previous models

-Previous detection systems repurpose classifiers to perform detection.

-Region proposal techniques typically generate a few thousand potential boxes per image. Selective Search, the most comon region proposal method on that time, takes 1-2 sec per image to generate these boxes

-Even a highly accurate classifier will produce false positives when faced with so many proposals. When viewed out of context, small sections of background can resemble actual objects, causing detection errors.

-These detection pipelines rely on independent techniques at every stage that cannot be optimized jointly.

Yolo model operations

-A single convolutional network simultaneously predicts multiple bounding boxes and class probabilites for these boxes.

-They train their network on full images and directly optimize detection performance.

-Their network uses global image features to predict detections which drastically reduces its errors from background detections.

-At test time, a single network evaluation of the full image produces detections of multiple objects in multiple categories without any pre or post-processing.

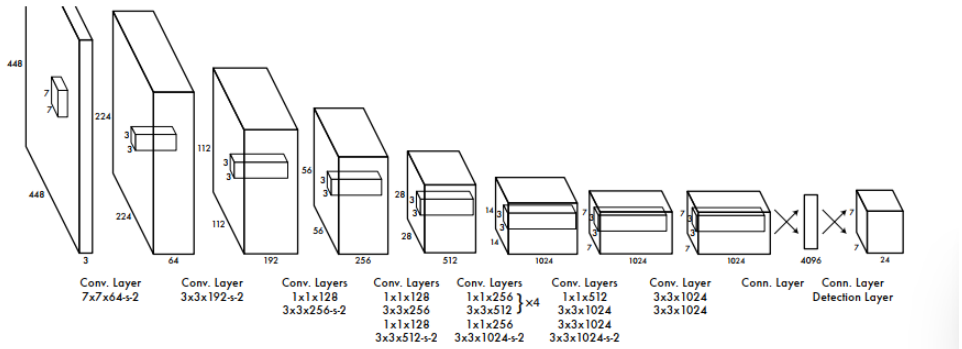

Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. The network uses strided convolutional layers to downsample the feature space instead of maxpooling layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Final Layer

-Their final layer predicts both class probabilities and bounding box coordinates.

-They normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parameterize the bounding box x and y coordinates to be offsets of a particular gird cell location so they are also bounded between 0 and 1.

-They used a logistic activation function to reflect these constraints on the final layer. All other layers use the following LeakyRelu activation:

$$\phi(x) = \begin{cases} 1.1x, & \text{if } x > 0 \\ .1x, & \text{otherwise} \end{cases}$$

Output of model

-Optimized for sum-squared error in the ouput

→ it is easy to optimize, however, it does not perfectly align with the goal of maximizing AP. It weights localization error equally with classification error which may not be ideal.

-To remedy this, they used a scaling factor lambda to adjust the weight given to error from coordinate predictions versus error from class probabilities. In the final model they used the scaling factor lambda = 4.

-SSE also equally weights errors in large boxes and small boxes. This error metric should reflect that small deviations in large boxes matter less than in small boxes.

-To partially address thism we predict the square root of the bounding box width and height instead of the width and height directly.

Loss function

If cell i predicts class probabilities $\hat{p}_i(\text{aeroplane}), \hat{p}_i(\text{bicycle})\ldots$ and the bounding box $\hat{x}_i, \hat{y}_i, \hat{w}_i, \hat{h}_i$ then our full loss function for an example is:

$$\sum_{i=0}^{48}\left(\lambda \mathbb{1}_i^{\text{obj}}((x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2+(\sqrt{w_i}-\sqrt{\hat{w}_i})^2+(\sqrt{h_i}-\sqrt{\hat{h}_i})^2)+\sum_{c\in\text{classes}}(p_i(c)-\hat{p}_i(c))^2\right) \quad (2)$$

Where $\mathbb{1}_i^{\text{obj}}$ encodes whether any object appears in cell i. Note that if there is no object in a cell we do not consider any loss from the bounding box coordinates predicted by that cell. In this case, there is no ground truth bounding box so we only penalize the associated probabilities with that region.

Hyperparameter tuning

-Based on the training and validation datasets from PasCal VOC2007 and 2012, as well as the test set from 2007, a total of 21k images, they used:

- A batch size of 64

- A momentum of 0.9 and a decay of 0.0005
- 2 learning rates during traing: 10^-2 and 10^-3
- Training diverges if we use the higher learning rate, 10^-2 from the start. We use lower rate, 10^-3 for one epoch so that the randomly initialized weights in the final layers can settle to reasonable values. Then they train with the following learning rate schedule: 10^-2 for 80 epochs, and 10^-3 for 40 epochs.

-To avoid overfitting, they used dropout and extensive data augmentation:

- A dropout layer with rate=.5 after the first connected layer prevents co-adaptation between layers.
- For data augmentation, we introduce random scaling and translations of up to 10% of the original image size.
- Randomly adjust the exposure and saturation of the image by up to a factor of 2.

Parameterizing Class Probalities

Problem with imbalance

-Each grid cell predicts class probabiliies for that area of the image. There are 49 cells with a possible 20 classes each yeilding 980 predicted probabilities per image.

-Most of these probabilies will be zero since only a few object appear in any given image. Left unchecked, this imbalance pushes all of the probabilities to zero, leading to divergence during training.

Solution

-They added an extra variable to each grid location. the probability that any object exists in that location regardless of class.

-Thus instead of 20 class, we have bonus 1 ‘objectness’:

To get the unconditional probability for an object class at a given location we simply multiply the "objectness" probability by the conditional class probability:

$$\Pr(\text{Dog}) = \Pr(\text{Object}) * \Pr(\text{Dog}|\text{Object}) \quad (3)$$

Limitations of YOLO

-Its impose strong spatial constraints on bounding box predictions since each grid cell only predicts one box.

-This spatial constraint limits the number of nearby object that our model can predict.

-If two objects fall into the same cell, the model can only predict one of them

→ the model struggles with small object that appear in groups, such as flocks of bird.

-Since the model learns to predict bounding boxes from data, it struggles to generalize to objects in new or unusual aspect ratios or configurations.

-The loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU