OBJECT DETECTION USING YOLO PROJECT REPORT

YOLOV3 pre-trained on MSCOCO Dataset

Abstract ON YOLO Net

The Network developers present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, the Network developers frame object detection as a regression problem to spatially separated bounding boxes and associated class 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 optimised end-to-end directly on detection performance.

This unified architecture is extremely fast. This base YOLO model processes images in real-time at 45 frames per second. A smaller version of the network, Fast YOLO, processes an astounding 155 frames per second while still achieving double the mAP of other real-time detectors. Compared to state-of-the-art detection systems, YOLO makes more localisation errors but is far less likely to pre- dict false detections where nothing exists. Finally, YOLO learns very general representations of objects. It outper- forms all other detection methods, including DPM and R- CNN, by a wide margin when generalising from natural images to artwork on both the Picasso Dataset and the People-Art Dataset.

Introduction

Humans glance at an image and instantly know what objects are in the image, where they are, and how they interact. The human visual system is fast and accurate, allowing us to perform complex tasks like driving with little conscious thought. Fast, accurate, algorithms for object detection would allow computers to drive cars in any weather without specialised sensors, enable assistive devices to con- vey real-time scene information to human users, and unlock the potential for general purpose, responsive robotic systems.

Current detection systems repurpose classifiers to per- form detection. To detect an object, these systems take a classifier for that object and evaluate it at various locations and scales in a test image. Systems like deformable parts models (DPM) use a sliding window approach where the classifier is run at evenly spaced locations over the entire image

More recent approaches like R-CNN use region proposal methods to first generate potential bounding boxes in an image and then run a classifier on these proposed boxes. After classification, post-processing is used to refine the bound- ing box, eliminate duplicate detections, and rescore the box based on other objects in the scene. These

complex pipelines are slow and hard to optimise because each individual component must be trained separately.

The Network developers reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. Using This system, you only look once (YOLO) at an image to predict what objects are present and where they are.

YOLO is refreshingly simple: see Figure 1. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for those boxes. YOLO trains on full images and directly optimises detection performance. This unified model has several benefits over traditional methods of object detection.

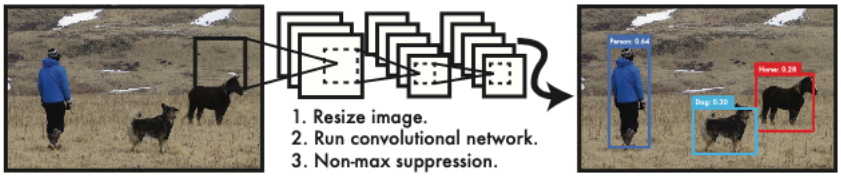

Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

First, YOLO is extremely fast. Since the Network developers frame detection as a regression problem the Network developers don't need a complex pipeline. The Network developers simply run This neural network on a new image at test time to predict detections. This base network runs at 45 frames per second with no batch processing on a Titan X GPU and a fast version runs at more than 150 fps. This means the Network developers can process streaming video in real-time with less than 25 milliseconds of latency. Furthermore, YOLO achieves more than twice the mean average precision of other real-time systems. For a demo of This system running in real-time on a webcam please see This (anonymous) YouTube channel: https://goo.gl/bEs6Cj.

Second, YOLO reasons globally about the image when making predictions. Unlike sliding window and region proposal-based techniques, YOLO sees the entire image during training and test time so it encodes contextual in- formation about classes as well as their appearance. Fast R-CNN, a top detection method, mistakes background patches in an image for objects because it can't see the larger context. YOLO makes less than half the number of background errors compared to Fast R-CNN.

Third, YOLO learns generalisable representations of objects. When trained on natural images and tested on art- work, YOLO outperforms top detection methods like DPM and

R-CNN by a wide margin. Since YOLO is highly generalisable it is less likely to break down when applied to new domains or unexpected input.

Unified Detection

The Network developers unify the separate components of object detection into a single neural network. This network uses features from the entire image to predict each bounding box. It also predicts all bounding boxes for an image simultaneously. This means This network reasons globally about the full image and all the objects in the image. The YOLO de- sign enables end-to-end training and real-time speeds while maintaining high average precision.

This system divides the input image into a S × 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 predicts B bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts. Formally the Network developers define confidence as $\Pr(\text{Object}) * \text{IOU}_{\text{pred}}$. If no

object exists in that cell, the confidence scores should be zero. Otherwise the Network developers want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.

Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally

the confidence prediction represents the IOU between the predicted box and any ground truth box.

Each grid cell also predicts C conditional class probabilities, $\Pr(\text{Class}_i|\text{Object})$. These probabilities are conditioned on the grid cell containing an object. The Network developers only predict one set of class probabilities per grid cell, regardless of the number of boxes B.

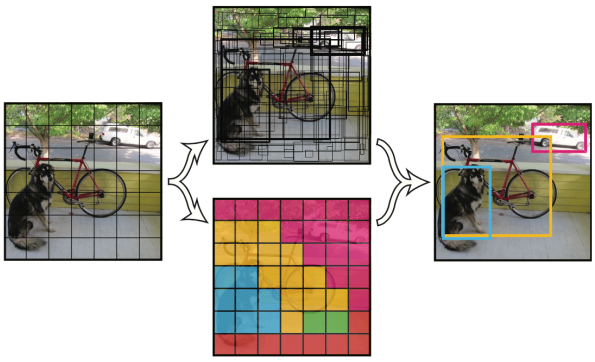

Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an even grid and simultaneously predicts bounding boxes, confidence in those boxes, and class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

At test time the Network developers multiply the conditional class probabilities and the individual box confidence predictions,

$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IOU}^{\text{truth}}_{\text{pred}} = \Pr(\text{Class}_i) * \text{IOU}^{\text{truth}}_{\text{pred}} \quad (1)$$

which gives us class-specific confidence

scores for each box. These scores encode both the probability of that class appearing in the box and how well the predicted box fits the object.

Design

The Network developers implement this model as a convolutional neural network and evaluate it on the PASCAL VOC detection dataset.The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates.

This network architecture is inspired by the GoogLeNet model for image classification. This network has 24 convolutional layers followed by 2 fully connected layers. However, instead of the inception modules used by GoogLeNet the Network developers simply use 1×1 reduction layers followed by 3×3 convolutional layers, similar to Lin et al. The full network is shown in Figure 3.

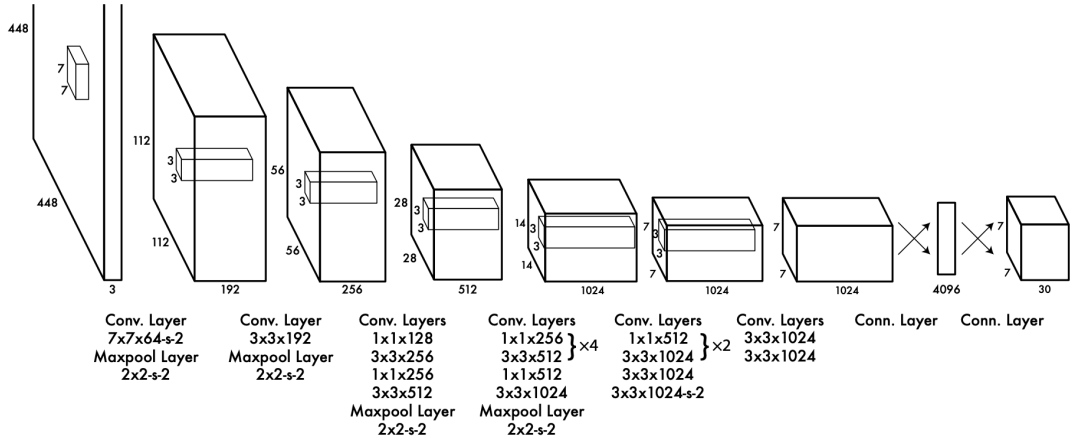

Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

The Network developers also train a fast version of YOLO designed to push the boundaries of fast object detection. Fast YOLO uses a neural network with fewer convolutional layers (9 instead of 24) and fewer filters in those layers. Other than the size of the network, all training and testing parameters are the same between YOLO and Fast YOLO.

The final output of This network is the $7 \times 7 \times 30$ tensor of predictions.

Training

The Network developers pretrain This convolutional layers on the ImageNet 1000-class competition dataset. For pretraining the Network developers use the first 20 convolutional layers from Figure 3 followed by a average-pooling layer and a fully connected layer. The Network developers train this network for approximately a week and achieve a single crop top-5 accuracy of 88% on the ImageNet 2012 validation set, comparable to the GoogLeNet models in Caffe's Model Zoo.

The Network developers then convert the model to perform detection. Ren et al. show that adding both convolutional and connected layers to pretrained networks can improve performance. Following their example, the Network developers add fThis convolutional lay- ers and two fully connected layers with randomly initialized weights. Detection often requires fine-grained visual infor- mation so the Network developers increase the input resolution of the network from 224 × 224 to 448 × 448.

This final layer predicts both class probabilities and bounding box coordinates. The Network developers normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. The Network developers parametrize the bounding box x and y coordinates to be offsets of a particular grid cell loca- tion so they are also bounded between 0 and 1.

The Network developers use a linear activation function for the final layer and all other layers use the following leaky rectified linear activation:

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise} \end{cases}$$

The Network developers optimize for sum-squared error in the output of this model. The Network developers use sum-squared error because it is easy to optimize, however it does not perfectly align with this goal of maximizing average precision. It weights localization er- ror equally with classification error which may not be ideal. Also, in every image many grid cells do not contain any object. This pushes the “confidence” scores of those cells towards zero, often overpowering the gradient from cells that do contain objects. This can lead to model instability, causing training to diverge early on.

To remedy this, the Network developers increase the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don’t contain objects. The Network developers use two parameters, λ_{coord} and λ_{noobj} to accomplish this. The Network developers set $\lambda_{coord} = 5$ and $\lambda_{noobj} = .5$.

Sum-squared error 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 this the Network developers predict the square root of the bounding box width and height instead of the width and height directly.

YOLO predicts multiple bounding boxes per grid cell. At training time the Network developers only want one bounding box predictor to be responsible for each object. The Network developers assign one predictor to be “responsible” for predicting an object based on which prediction has the highest current IOU with the ground truth. This leads to specialization between the bounding box predictors. Each predictor gets better at predicting certain sizes, aspect ratios, or classes of object, improving overall recall.

During training the Network developers optimize the following, multi-part loss function:

$$\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2$$

$$+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i\right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i\right)^2$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2 \qquad (3)$$

where 1_i denotes if object appears in cell i and 1_{ij} denotes that the jth bounding box predictor in cell i is “responsible” for that prediction.

Note that the loss function only penalizes classification error if an object is present in that grid cell (hence the conditional class probability discussed earlier). It also only penalizes bounding box coordinate error if that predictor is “responsible” for the ground truth box (i.e. has the highest IOU of any predictor in that grid cell).

The Network developers train the network for about 135 epochs on the train- ing and validation data sets from PASCAL VOC 2007 and 2012. When testing on 2012 the Network developers also include the VOC 2007 test data. Throughout training the Network developers use a batch size of 64, a momentum of 0.9 and a decay of 0.0005.

This learning rate schedule is as follows: For the first epochs the Network developers slowly raise the learning rate from 10^-3 to 10^-2.The Network developers continue training with 10^{-2} for 75 epochs, then decrease to 10^{-3} for 30 epochs, and finally decrease again to 10^{-4} for 30 epochs

To avoid overfitting the Network developers use dropout and extensive data augmentation. A dropout layer with rate = .5 after the first connected layer prevents co-adaptation between layers [18]. For data augmentation the Network developers introduce random scaling and translations of up to 20% of the original image size. The Network developers also randomly adjust the exposure and saturation of the im- age by up to a factor of 1.5 in the HSV color space.

Inference

Just like in training, predicting detections for a test image only requires one network evaluation. On PASCAL VOC the network predicts 98 bounding boxes per image and class probabilities for each box. YOLO is extremely fast at test time since it only requires a single network evaluation, un- like classifier-based methods.

The grid design enforces spatial diversity in the bound- ing box predictions. Often it is clear which grid cell an object falls in to and the network only predicts one box for each object. However, some large objects or objects near the border of multiple cells can be well localized by multiple cells. Non-maximal suppression can be used to fix these multiple detections

Limitations of YOLO

YOLO imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class. This spatial constraint lim- its the number of nearby objects that this model can pre- dict. This model struggles with small objects that appear in groups, such as flocks of birds.

Since this model learns to predict bounding boxes from data, it struggles to generalize to objects in new or unusual aspect ratios or configurations. This model also uses relatively coarse features for predicting bounding boxes since this architecture has multiple downsampling layers from the input image.

Finally, while the Network developers train on a loss function that approximates detection performance, this 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. This main source of error is incorrect localisations.

YOLOv3: An Incremental Improvement

We present some updates to YOLO! We made a bunch of little design changes to make it better. We also trained this new network that's pretty swell. It's a little bigger than last time but more accurate. It's still fast though, don't worry. At 320 × 320 YOLOv3 runs in 22 ms at 28.2 mAP, as accurate as SSD but three times faster. When we look at the old .5 IOU mAP detection metric YOLOv3 is quite good. It achieves 57.9 AP50 in 51 ms on a Titan X, compared to 57.5 AP50 in 198 ms by RetinaNet, similar performance but 3.8× faster

The Deal

So here's the deal with YOLOv3: We mostly took good ideas from other people. We also trained a new classifier network that's better than the other ones. We'll just take you through the whole system from scratch so you can understand it all

1. Bounding Box Prediction
2. Class Prediction
3. Predictions Across Scales
4. Feature Extractor

5. Training :- We still train on full images with no hard negative mining or any of that stuff. We use multi-scale training, lots of data augmentation, batch normalisation, all the standard stuff

FINAL SUMMERY

YOLO for Object Detection

Object detection is a computer vision task that involves both localizing one or more objects within an image and classifying each object in the image.

It is a challenging computer vision task that requires both successful object localization in order to locate and draw a bounding box around each object in an image, and object classification to predict the correct class of object that was localized.

The "*You Only Look Once*," or YOLO, family of models are a series of end-to-end deep learning models designed for fast object detection, developed by Joseph Redmon, et al. and first described in the 2015 paper titled "You Only Look Once: Unified, Real-Time Object Detection."

The approach involves a single deep convolutional neural network (originally a version of GoogLeNet, later updated and called DarkNet based on VGG) that splits the input into a grid of cells and each cell directly predicts a bounding box and object classification. The result is a large number of candidate bounding boxes that are consolidated into a final prediction by a post-processing step.

There are three main variations of the approach, at the time of writing; they are YOLOv1, YOLOv2, and YOLOv3. The first version proposed the general architecture, whereas the second version refined the design and made use of predefined anchor boxes to improve bounding box proposal, and version three further refined the model architecture and training process.

Although the accuracy of the models is close but not as good as Region-Based Convolutional Neural Networks (R-CNNs), they are popular for object detection because of their detection speed, often demonstrated in real-time on video or with camera feed input.

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.

— You Only Look Once: Unified, Real-Time Object Detection, 2015.

Object Detection With YOLOv3

The keras-yolo3 project provides a lot of capability for using YOLOv3 models, including object detection, transfer learning, and training new models from scratch.

In thisProject, we will use a pre-trained model to perform object detection on an unseen photograph.