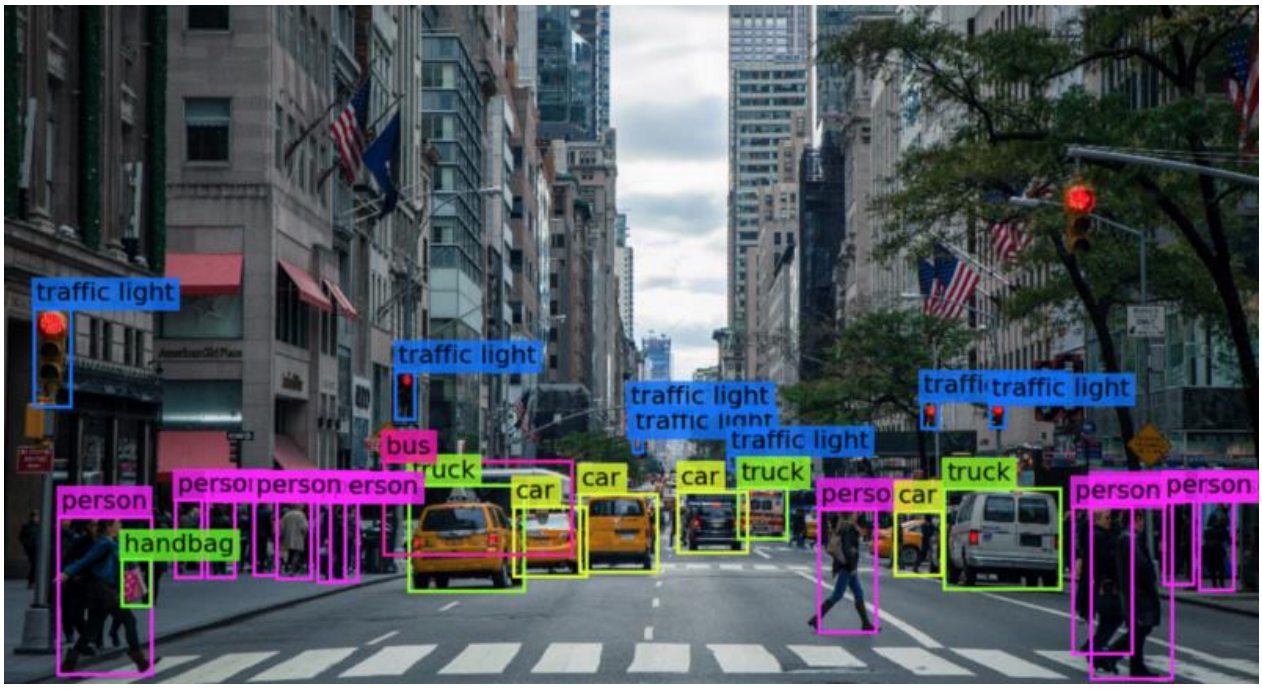


Computer Vision

You Only Look Once algorithm

Artificial Intelligent

Iheb Bouariche | YOLO algorithm

1. Computer Vision

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs and take actions or make recommendations based on that information. AI's computer vision algorithms enable computers to observe and understand.

Computer vision needs lots of data for AI machine learning. Two essential technologies are used to accomplish this: deep learning and convolutional neural network and recurrent neural network.

Machine learning uses algorithmic models that enable a computer to teach itself about the context of visual data. If enough data is fed through the model, the computer will look at the data and teach itself to tell one image from another. Algorithms enable the machine to learn by itself, rather than someone programming it to recognize an image.

CNN discerns hard edges and simple shapes, then fills in information as it runs iterations of its predictions. A CNN is used to understand single images. A recurrent neural network (RNN) is used in a similar way for video applications to help computers understand how pictures in a series of frames are related to one another.

From IBM.com

2. Object detection

Object detection is a computer vision technique for locating instances of objects in images or videos. Object detection algorithms typically leverage machine learning or deep learning to produce meaningful results. When humans look at images or video, we can recognize and locate objects of interest within a matter of moments. The goal of object detection is to replicate this intelligence using a computer.

3. YOLO (You only look once algorithm)

YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture in real-time. Object detection in YOLO is done as a regression problem and provides the class probabilities of the detected images.

YOLO proposes the use of an end-to-end neural network including convolutional layers that makes predictions of bounding boxes and class probabilities all at once.

Following a fundamentally different approach to object detection, YOLO achieves state-of-the-art results beating other real-time object detection algorithms by a large margin.

4. How YOLO work?

Yolo uses single neural network and can reasons globally about the full image and all the objects in the image.

First the yolo system divides the input image into an 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 box contains an object and also how accurate it thinks the box in the right position and

geometry. The confidence is defined as Pr(Object) $*$ IOU(truth/pred) . If no object exists in that cell, the confidence scores should be zero.

Otherwise, the confidence score must be equal the intersection over union (IOU) between the predicted box and the ground truth.

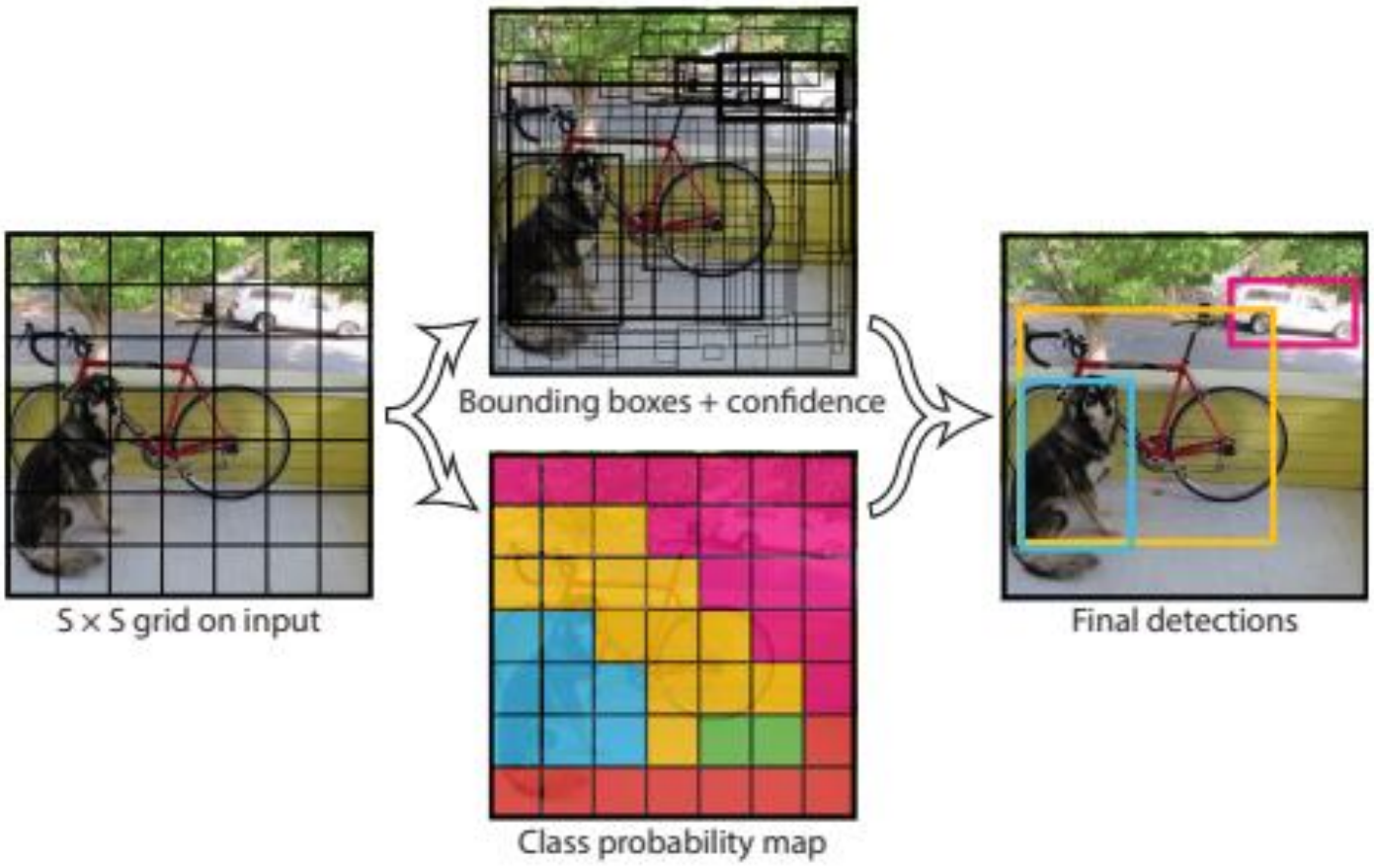
Each bounding box consists of 5 predictions: x, y, w, h, and confidence. The (x, y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. Finally, the confidence prediction represents the IOU between the predicted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

Each grid cell also predicts C conditional class probabilities, Pr(Classi|Object). These probabilities are conditioned on the grid cell containing a classification object.

Yolo predict one set of class probabilities per grid cell, regardless of the number of boxes B. At test time we multiply the conditional class probabilities and the individual box confidence predictions.

$$\Pr(\text{Classi}|\text{Object}) * \Pr(\text{Object}) * \text{IOU}(\text{truth/pred}) = \Pr(\text{Classi}) * \text{IOU}(\text{truth/pred})$$

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.



From https://arxiv.org/pdf/1506.02640.pdf

You Only Look Once

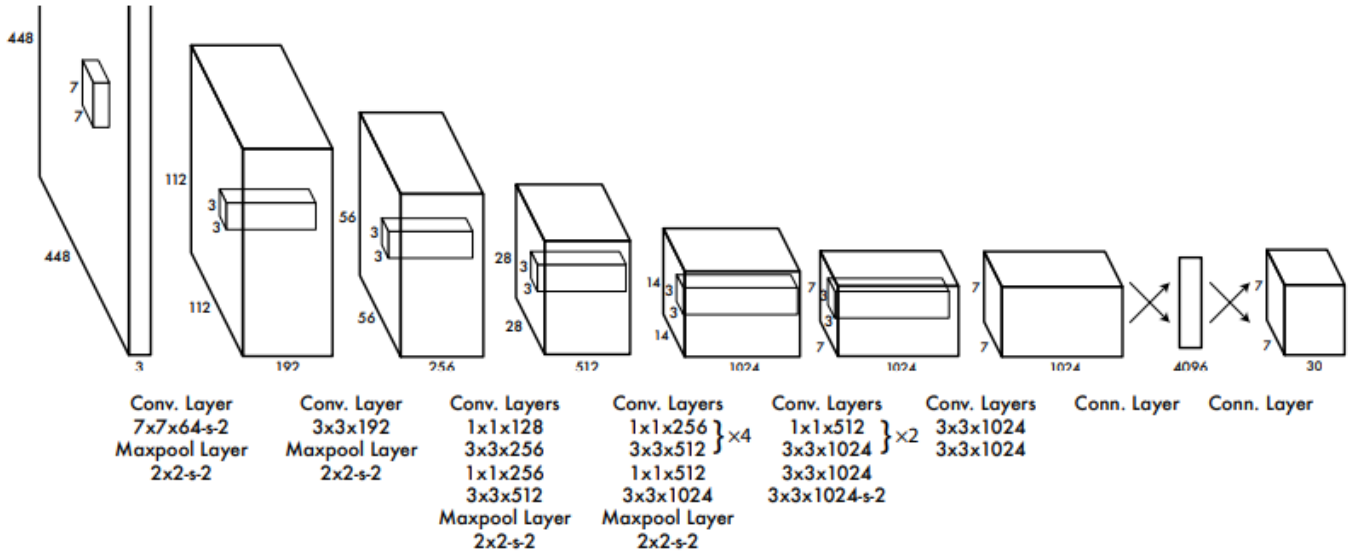
Unified, Real-Time Object Detection

Yolo model detection as a regression problem. It divides the image into an S × S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an S × S × (B ∗ 5 + C) tensor.

5. Network design

The model contains a convolutional and fully connected neural network layers. The initial convolutional layers of the network extract feature from the image while the fully connected layers predict the output probabilities and coordinates.

Network architecture is inspired by the GoogLeNet model for image classification. The model has 24 convolutional layers followed by 2 fully connected layers. Instead of the inception modules used by GoogLeNet, the model uses 1 × 1 reduction layers followed by 3 × 3 convolutional layers.



Architecture of the detection network designed with 24 convolutional layers followed by 2 fully connected layers. Alternating 1 × 1 convolutional layer reduce the features space from preceding layers. The convolutional network is pretrained on the ImageNet classification task at half of the resolution (224 × 224 input image) and then double the resolution for detection.

6. Training

The first 20 convolutional network layers followed by an average-pooling layer and a fully connected layer was pretrained on the ImageNet 1000-class competition dataset for approximately a week.

Then the model converted to perform detection. And adding four convolutional layers and two fully connected layers with randomly initialized weights can improve performance. And increasing the input resolution of the network from 224 × 224 to 448 × 448 (detection often requires fine-gained visual information)

The final layer predicts both class probabilities and bounding box coordinates. And do normalization to the bounding box width and height by the image width and height so that they fall between 0 and 1. And parametrize the bounding box x and y coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1.

A linear activation function is used for the final layer and all other layers use the following leaky rectified linear activation:

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

Using the sum-squared error to optimize this model does not perfectly align with our goal of maximizing the average precision. Because, 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.

The solution is by increasing the loss from bounding box coordinate predictions and decrease the loss from confidence predictions for boxes that don't contain objects. two parameters are used (λcoord=5) and (λnoobj=0.5) to accomplish this.

Sum-squared error also equally weights errors in large boxes and small boxes. The error metric should reflect that small deviations in large boxes matter less than in small boxes. The solution is by predicting the square root of the bounding box width and height instead of the width and height directly.

The Loss function:

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

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

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

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

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

Where 1(obj,i) denotes if object appears in cell "i", and 1(obj,ij) denotes that the j th 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. 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 model is trained for about 135 epochs on the training and validation data sets from PASCAL VOC 2007 and 2012.it train on a batch size of 64 and a optimizer momentum of 0.9.

The learning rate schedule is as follows: For the first epochs the learning rate start from 10−3 to 10−2. If we start at a high learning rate our model often diverges due to unstable gradients. Then it continues training with 10−2 for 75 epochs, then 10−3 for 30 epochs, and finally 10−4 for 30 epochs.

To avoid overfitting, dropout and extensive data augmentation are used. A dropout layer with rate = 0.5 and data augmentation by introducing random scaling and translations of up to 20% of the original image size. And we also randomly adjust the exposure and saturation of the image by up to a factor of 1.5 in the HSV color space.

7. YOLO Family

Originally introduced by Joseph Redmon in Darknet, YOLO has come a long way. Here are a few things that made the YOLO's first version break the competition over R-CNN and DPM:

- Real-time frames processing at 45 fps
- Less false positive on the background
- Higher detection accuracy (although lower accuracy on localization)

The algorithm has continued to evolve ever since its initial release in 2016. Both YOLOv2 and YOLOv3 were written by Joseph Redmon. After YOLOv3, there came new authors who anchored their own goals in every other YOLO release.

YOLOv2: Released in 2017, this version earned an honorable mention at CVPR 2017 because of significant improvements on anchor boxes and higher resolution.

YOLOv3: The 2018th release had an additional objectivity score to the bounding box prediction and connections to the backbone network layers. It also provided an improved performance on tiny objects because of the ability to run predictions at three different levels of granularity.

YOLOv4: April's release of 2020 became the first paper not authored by Joseph Redmon. Here Alexey Bochkovski introduced novel improvements, including mind activation, improved feature aggregation, etc.

YOLOv5: Glenn Jocher continued to make further improvements in his June 2020 release, focusing on the architecture itself.

8. Application of YOLO

As expected, YOLO's applications expand as far as the use cases of object detection, including to the following:

- Autonomous driving: here object detection is required to avoid accidents on the road since there is no human managing the wheel. In this case, YOLO helps detect people, cars, or any external hazards appearing on the road.
- Wildlife detection: this is as applicable for trees and biodiversity as it is for different species of animals to track their growth and migration.
- Robotics: depending on the industry the robot operates in, some robots do require computer vision to detect objects on their path and perform a particular instruction.

- Retail: visual product search or reverse image search are becoming increasingly popular in retail, which wouldn't have been possible without object detection algorithms like YOLO.
- Security: YOLO can also be used in security systems to enforce security in an area. Let's assume that people have been restricted from passing through a certain area for security reasons. If someone passes through the restricted area, the YOLO algorithm will detect him/her, which will require the security personnel to take further action.

And others…

9. Conclusion

YOLO is the state-of-the-art real-time object detection algorithm as it is much faster compared to other algorithms while being able to maintain a good accuracy. YOLO network understands generalized object representation however, the spatial constraints limit the accuracy in case of nearby and smaller objects. We have newer versions of this algorithm, we have YOLOv4 which addresses this problem and it's more accurate and faster. Overall, YOLO's speed and accuracy makes it a widely used algorithm for real-time object detection. There are others versions of YOLO until the latest one YOLOv7 (released in 2022).