Ref research paper: [You Only Look Once: Unified, Real-Time Object Detection]

DPM(Deformable parts model) in easier terms:
Let's say you're at a costume party. You see a friend dressed as a pirate. The pirate outfit has a hat (the root), a sword (a part), and a parrot (another part). If your friend shifts around a bit, the hat might tilt, the sword might be held differently, but you still recognize them as a pirate because the key elements are there, and you can see how they relate to each other.
In contrast, R-CNN would be like a fancy camera that takes a snapshot of your friend and then analyzes the whole image to identify them as a pirate. It's effective, but it doesn't account for the small shifts in costume parts as flexibly as DPM does.

Technically:

1. Root Filter: This is like the main piece of the puzzle that gives you the general shape of the object. It defines the overall bounding box for the object.
2. Part Filters: These are smaller filters that represent various parts of the object. Each part can move around a bit, which is where the "deformable" aspect comes in. This flexibility allows DPM to recognize objects that might be in different poses or slightly occluded.
3. Spatial Arrangement: DPM uses a model to score how well the parts fit together relative to the root. It's like checking if the wheels fit nicely under the car body, even if they're a bit off-center.

R-CNN(Region-based convolutional network) in easier terms:

Imagine R-CNN as a movie director casting actors for a film. The director first looks at a large pool of audition tapes (the entire image) and selects a few promising candidates (region proposals). Each candidate is then called in for a closer audition (feature extraction). After evaluating their performances (classification), the director adjusts their roles to fit better in the film (bounding box regression). Finally, the director decides which actors to cast, ensuring there are no duplicates (NMS).

Technically:

1. Region Proposal Generation: R-CNN starts by generating around 2,000 region proposals using algorithms like Selective Search. This step identifies potential bounding boxes where objects might be located.
2. Feature Extraction: Each proposed region is resized to a fixed size (e.g., 224x224 pixels) and passed through a CNN (often AlexNet) to extract feature vectors. These vectors represent the content of the regions in a compact form.
3. Object Classification: The extracted features are then fed into a separate classifier (typically SVMs) trained for each object class. This classification step determines whether the region contains an object of interest.
4. Bounding Box Regression: Alongside classification, R-CNN also performs bounding box regression to refine the proposed bounding boxes, improving the accuracy of object localization.

5. Non-Maximum Suppression: After classification and bounding box adjustments, R-CNN applies NMS to filter out overlapping boxes, ensuring that only the most confident detections are retained.

Selective search:

1. Initial Segmentation: The algorithm begins by segmenting the image into many small regions based on color, texture, size, and shape. Imagine you're at a party and you start by grouping people into small clusters based on their outfits—like all the people in blue shirts together.
2. Combining Regions: Next, Selective Search iteratively merges these small segments into larger ones. It looks for similar regions (like people wearing similar outfits) and combines them to form larger bounding boxes. This is done based on similarity metrics such as color and texture.
3. Generating Proposals: After several iterations of merging, the algorithm produces around 2,000 region proposals. Each proposal is a potential bounding box that might contain an object. Think of it as creating a list of all the clusters of people you think might be interesting to talk to at the party.

YOLO (You Only Look Once) is a groundbreaking object detection system that simplifies the process by reframing detection as a single regression problem, predicting bounding boxes and class probabilities simultaneously in one pass.

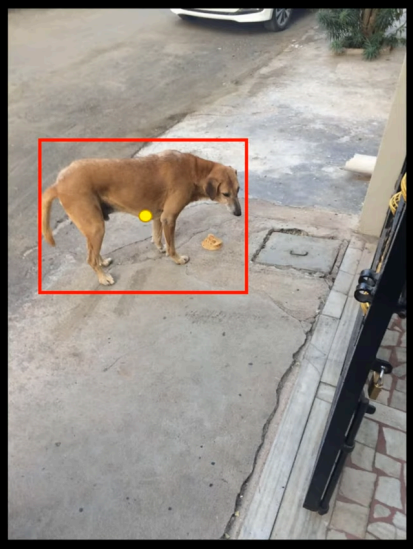
Speed and Efficiency: YOLO is extremely fast, running at 45 frames per second (fps) on a Titan X GPU, and a faster version can achieve 150 fps. This makes it capable of processing real-time video with less than 25 milliseconds of latency.

Global Context: Unlike traditional methods, YOLO considers the entire image during training and testing, allowing it to better understand the context and reduce background errors significantly compared to methods like Fast R-CNN.

Generalization: YOLO is highly generalizable and performs well across different domains, such as natural images and artwork, outperforming other detection methods in these scenarios.

Accuracy Limitations: While YOLO is fast, it struggles with precise localization, particularly for smaller objects, making it less accurate than some state-of-the-art detection systems.

Object Localization



$$\begin{bmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 50 \\ 70 \\ 60 \\ 70 \\ 1 \\ 0 \end{bmatrix}$$

C_1 = *Dog class*
C2 = Person Class

$$\begin{bmatrix} 1 \\ 30 \\ 28 \\ 28 \\ 82 \\ 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ - \\ - \\ - \\ - \\ - \\ - \end{bmatrix}$$

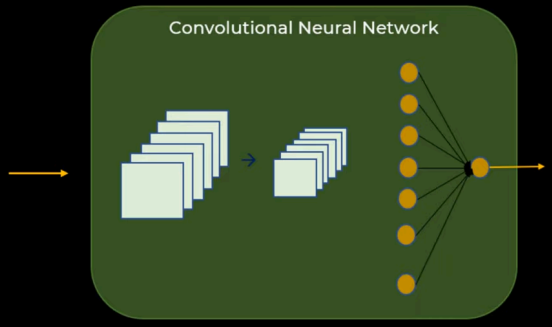
X_train

y_train

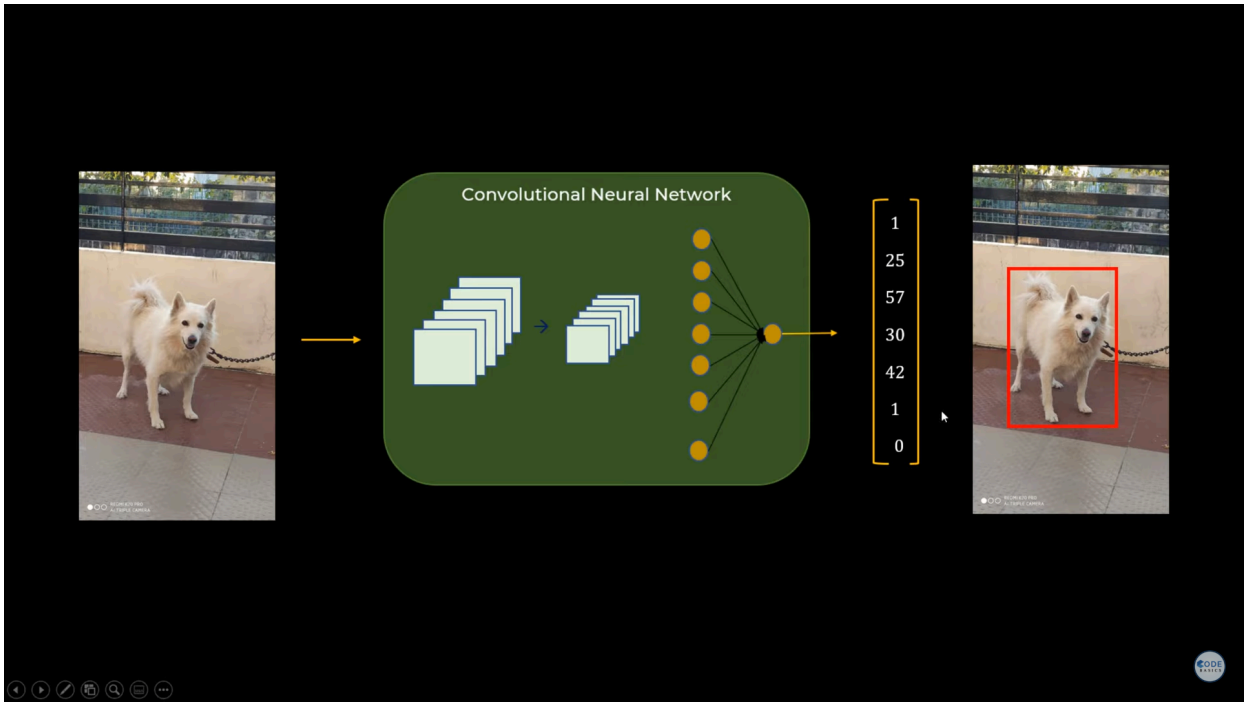
$$\begin{bmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{bmatrix} \quad \begin{bmatrix} 1 \\ 50 \\ 70 \\ 60 \\ 70 \\ 1 \\ 0 \end{bmatrix}$$

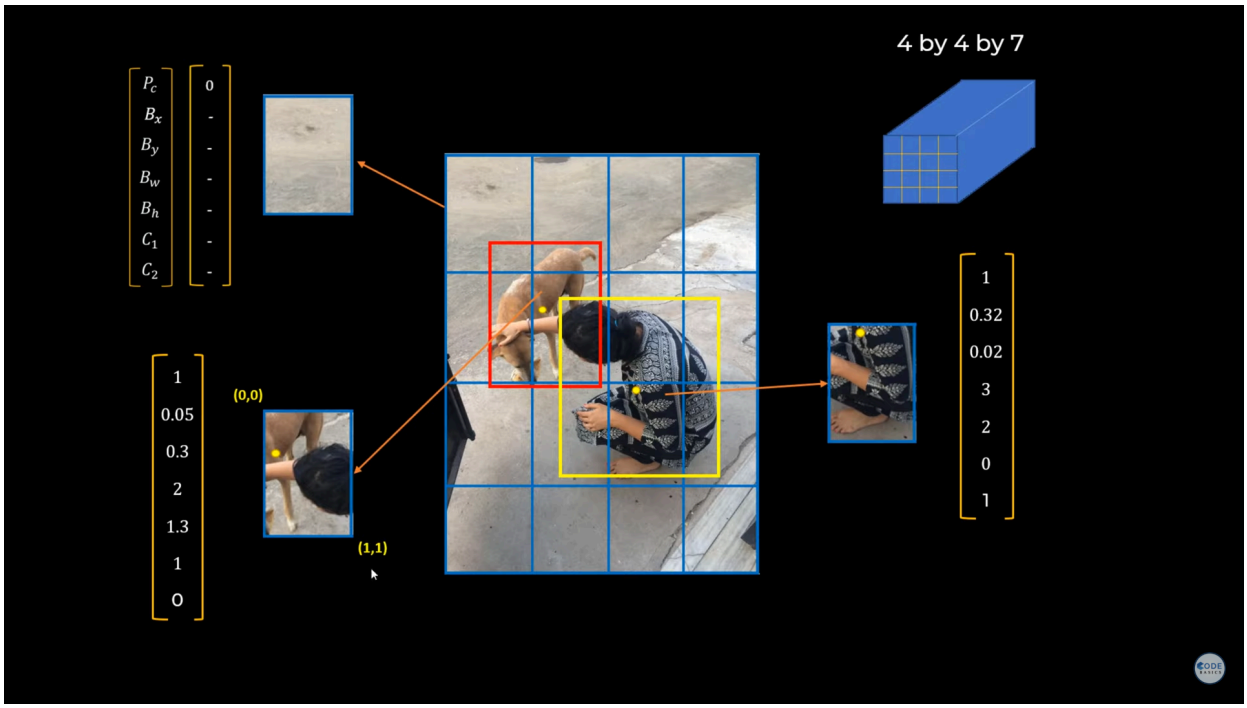
$$\begin{bmatrix} 1 \\ 30 \\ 55 \\ 28 \\ 82 \\ 0 \\ 1 \end{bmatrix}$$

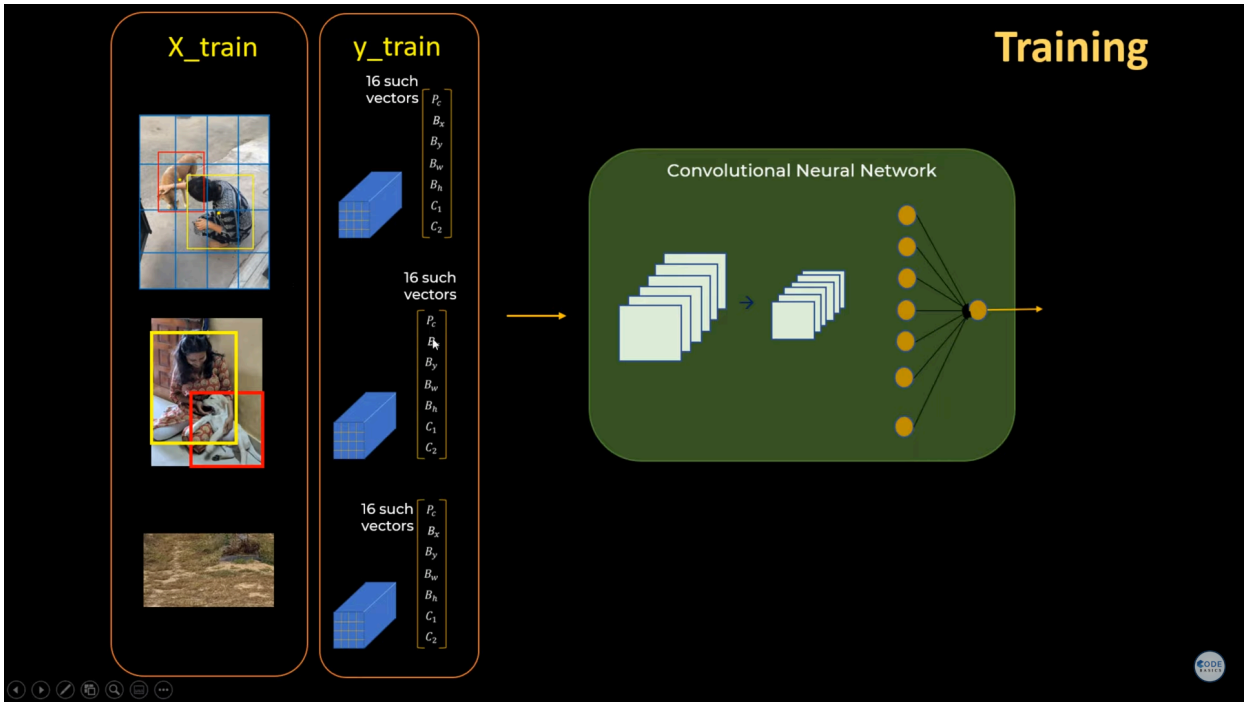
$$\begin{bmatrix} 0 \\ - \\ - \\ - \\ - \\ - \\ - \end{bmatrix}$$

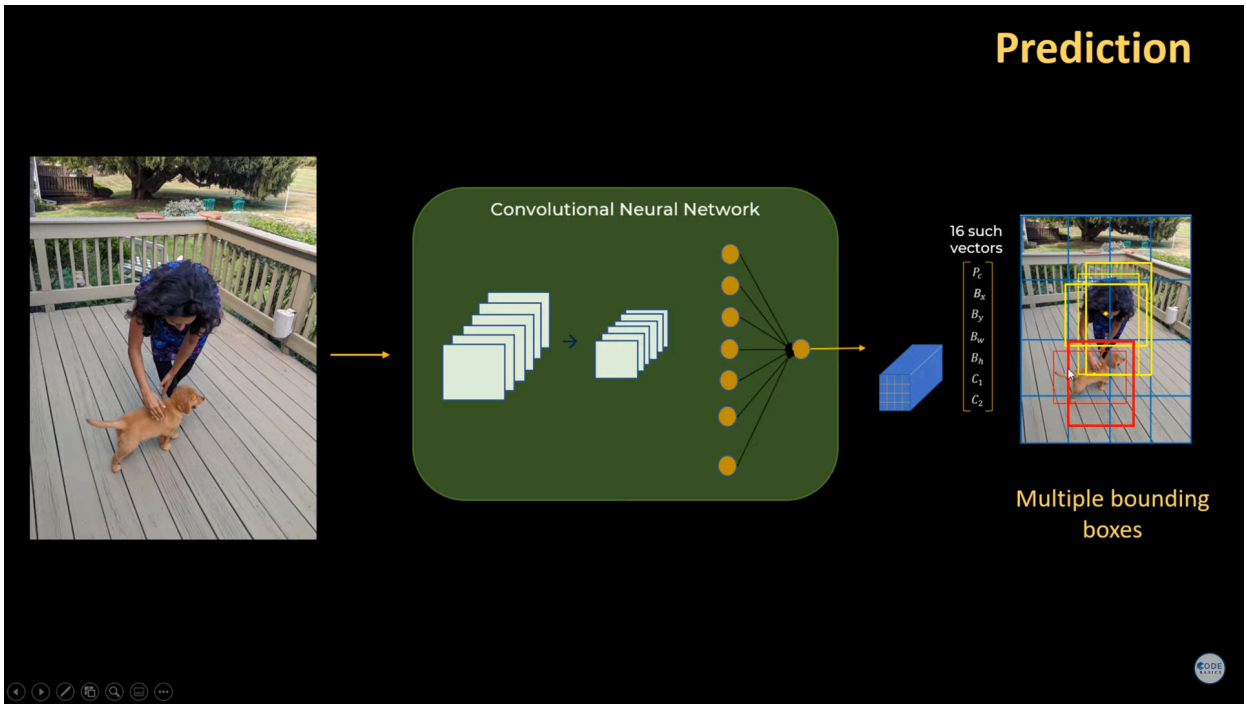


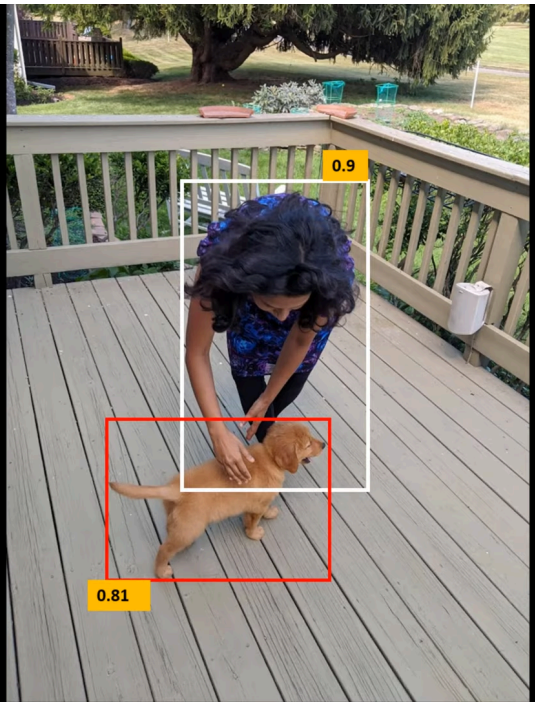
$$\begin{bmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{bmatrix}$$


Convolutional Neural Network
1
25
57
30
42
1
0


4 by 4 by 7
P_c
B_x
B_y
B_w
B_h
C_1
C_2
0
-
-
-
-
-
-
1
0.05
0.3
2
1.3
1
0
(0,0)
(1,1)
1
0.32
0.02
3
2
0
1

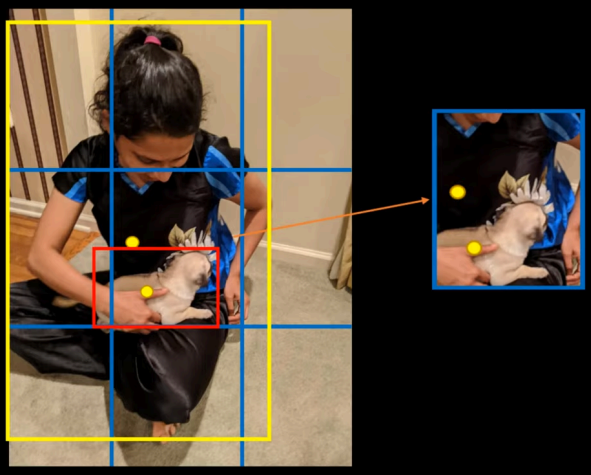

X_train
y_train
16 such vectors
P_c
B_x
B_y
B_w
B_h
C_1
C_2
16 such vectors
16 such vectors
Convolutional Neural Network
Training


Prediction
Convolutional Neural Network
16 such vectors
P_c
B_x
B_y
B_w
B_h
C_1
C_2
Multiple bounding boxes



Non max suppression

What if one grid cell has center of two objects?



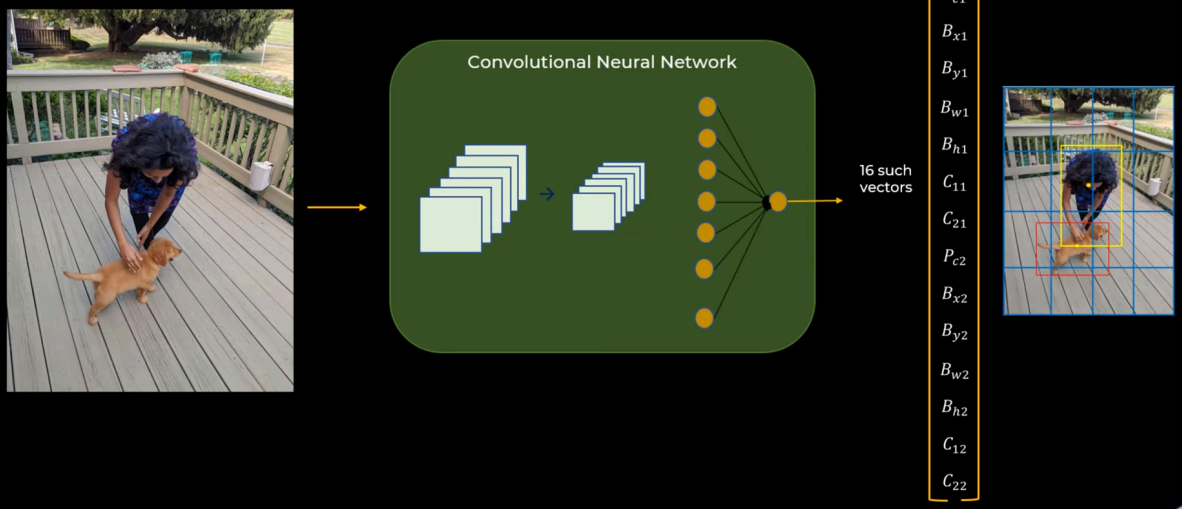
$$\begin{bmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{bmatrix}$$

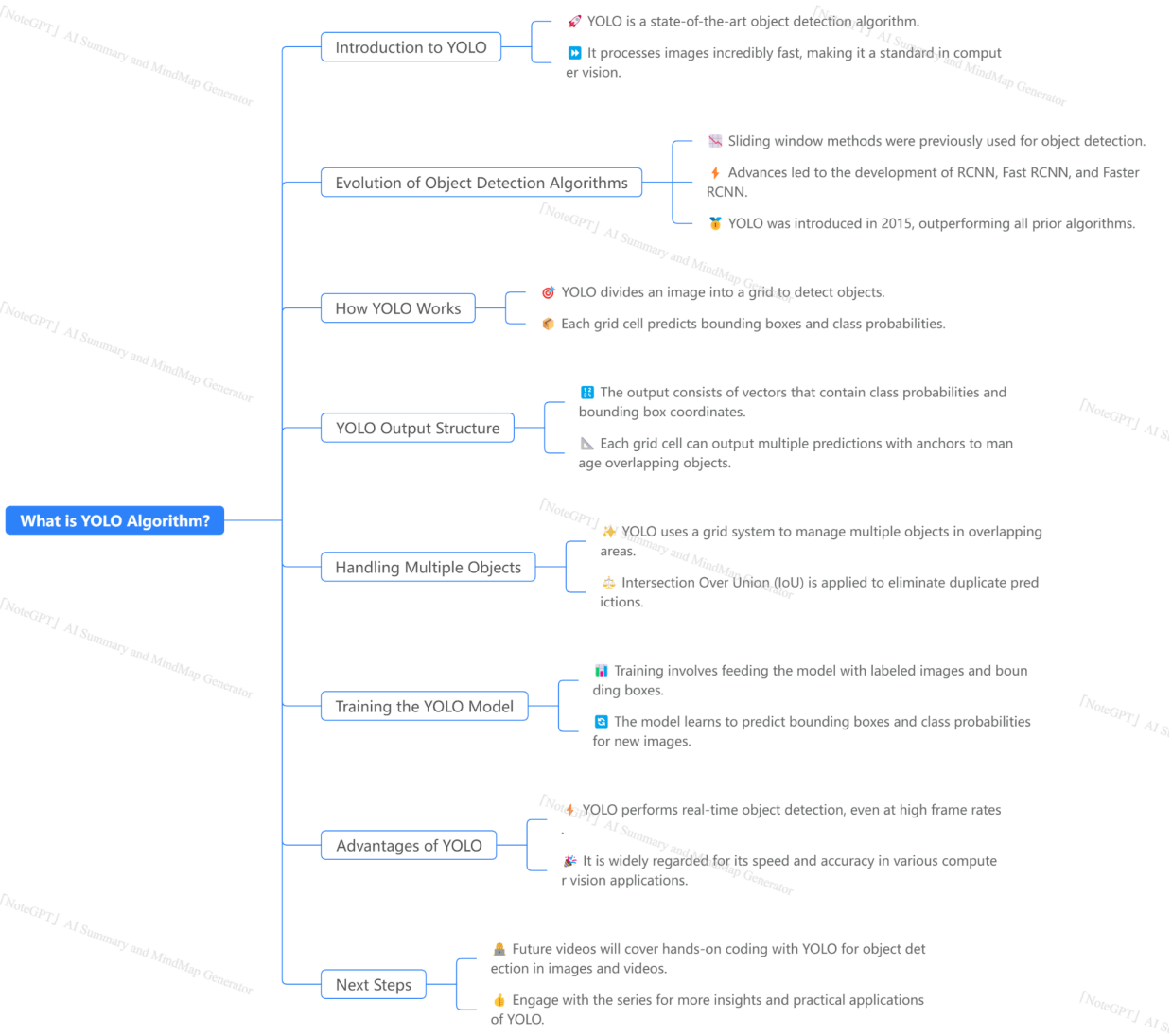
$$\begin{bmatrix} 1 \\ 0.22 \\ 0.45 \\ 1 \\ 0.7 \\ 1 \\ 0 \end{bmatrix}$$ Dog

$$\begin{bmatrix} 1 \\ 0.13 \\ 0.38 \\ 2.1 \\ 3 \\ 0 \\ 1 \end{bmatrix}$$ Person

$$\begin{bmatrix} 1 \\ 0.22 \\ 0.45 \\ 1 \\ 0.7 \\ 1 \\ 0 \\ 1 \\ 0.13 \\ 0.38 \\ 2.1 \\ 3 \\ 0 \\ 1 \end{bmatrix}$$

CNN with Two anchor boxes





Ref video : What is YOLO algorithm? | Deep Learning Tutorial 31 (Tensorflow, Keras & Pytho...

Architecture:

YOLO's convolutional neural network architecture: The initial convolutional layers extract features from the image, while the fully connected layers predict output probabilities and coordinates. Focusing on optimizing the convolutional layers can lead to significant improvements in the model's performance.

Network depth: YOLO uses 24 convolutional layers, while Fast YOLO uses only 9 layers. Striking the right balance between depth and computational efficiency is crucial for real-time object detection.

Filter count: The number of filters in each convolutional layer also impacts the model's performance. Fast YOLO uses fewer filters compared to YOLO, prioritizing speed over accuracy.

Training on the PASCAL VOC dataset: The PASCAL VOC dataset is a widely used benchmark for object detection, segmentation, and classification. Optimizing the model's performance on this dataset can lead to significant improvements in its overall effectiveness.

Predicting output probabilities and coordinates: The final output of the YOLO network is a tensor of predictions, which includes both class probabilities and bounding box coordinates. Ensuring the accuracy of these predictions is essential for effective object detection.

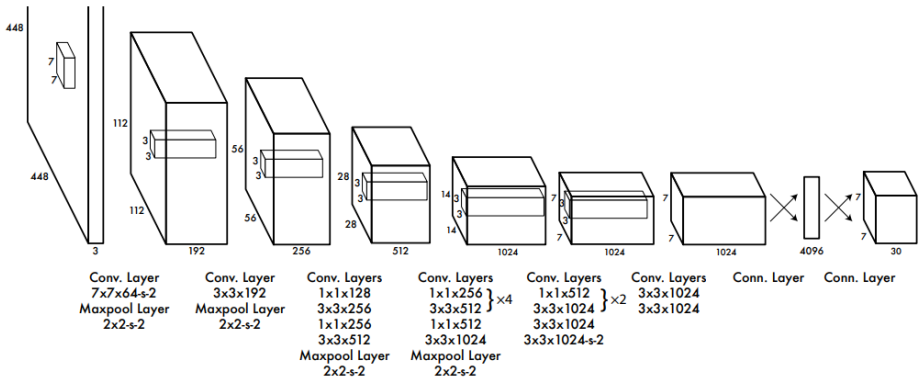


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

Detailed Arch:

1. Feature Extraction:

- The first few convolutional layers (1-5) extract low-level features such as edges, lines, and basic shapes from the input image. These features are then passed through a Rectified Linear Unit (ReLU) activation function to introduce non-linearity and a max-pooling layer to reduce the spatial dimensions of the feature maps while retaining important information.

- The middle convolutional layers (6-15) build upon the low-level features and extract more complex patterns and shapes, such as object parts and textures. These layers are also followed by ReLU activations and max-pooling layers.

- The final convolutional layers (16-24) capture high-level semantic features and object-level information. These features are crucial for accurately identifying and classifying objects within the image.

2. Increased Model Capacity:

- More layers increase the model's capacity to learn from data. With 24 convolutional layers, YOLO can learn a richer representation of the data, which is particularly important for complex tasks like object detection where variations in object appearance, scale, and context can be significant.

3. Global Reasoning:

- The architecture allows the model to reason globally about the entire image rather than focusing on localized regions. This is achieved through the use of convolutional layers that

process the whole image in a single forward pass, enabling the model to make predictions based on the overall context of the scene, which is critical for accurate detection.

4. Reduction of Overfitting:

- The depth of the network, combined with techniques like batch normalization and dropout, helps in regularizing the model. This reduces the risk of overfitting, allowing the model to generalize better to unseen data.

5. Efficient Processing:

- The design of YOLO, including the number of convolutional layers, is optimized for speed and efficiency. The architecture is structured to allow for real-time processing, which is a significant advantage over traditional two-stage detectors that require multiple passes through the network.

Functions of Convolutional Layers:

- Convolution Operation: Each convolutional layer applies a set of filters (kernels) to the input image or the output of the previous layer. This operation helps in detecting specific features in the image.

- Activation Function: After the convolution operation, an activation function (typically Leaky ReLU in YOLO) is applied to introduce non-linearity into the model, enabling it to learn complex mappings from inputs to outputs.

- Pooling Layers: Although not all layers are pooling layers, the architecture includes max-pooling layers that down-sample the feature maps, reducing their spatial dimensions while retaining important information. This helps in making the model more computationally efficient.

- Final Output: The last few layers of the network are fully connected layers that take the high-level features extracted by the convolutional layers and convert them into predictions for bounding box coordinates and class probabilities.

Training done:

Pretraining on ImageNet

1. Dataset: The model is pretrained on the ImageNet dataset, which consists of 1,000 classes and over 1.2 million images. The initial architecture includes 20 convolutional layers, followed by an average pooling layer and a fully connected layer.

2. Training Accuracy: After training for about a week, the model achieves a top-5 accuracy of 88% on the ImageNet validation set. This indicates that for 88% of the images, the correct class label is among the top five predictions made by the model.

Model Architecture for Detection

1. Adding Layers: To adapt the pretrained model for object detection, additional layers are added:

- Four convolutional layers

- Two fully connected layers

2. Input Size: The input resolution is increased from 224x224 pixels to 448x448 pixels to capture more detail, which is crucial for detecting small objects.

3. Output Layer: The final layer outputs both class probabilities and bounding box coordinates. The bounding box dimensions are normalized by the image dimensions to ensure they are between 0 and 1.

$$\text{Normalized Width} = \frac{\text{Width}}{\text{Image Width}}$$

$$\text{Normalized Height} = \frac{\text{Height}}{\text{Image Height}}$$

4. Bounding Box Coordinates: The x and y coordinates of the bounding box are defined as offsets from a grid cell location, also normalized to the range [0, 1].

Activation Functions

1. Activation Functions: The final layer uses a linear activation function, while all other layers use a leaky rectified linear unit (ReLU):

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

Loss Function

1. **Sum-Squared Error:** The model optimizes for sum-squared error (SSE) in its predictions. The loss function is structured to consider both localization (bounding box) and classification errors:

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

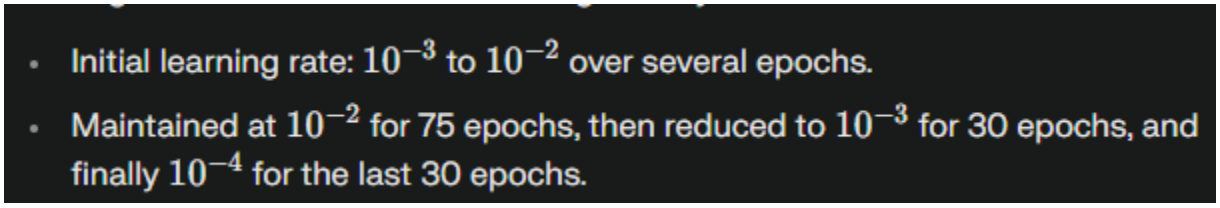
Here, $1_{\text{obj}}(i)$ indicates whether an object is present in grid cell i, and B is the number of bounding boxes predicted per cell.

2. **Weighting Factors:** The parameters λ_{coord} and λ_{noobj} adjust the importance of bounding box predictions versus confidence predictions. For example, $\lambda_{\text{coord}} = 5$ and $\lambda_{\text{noobj}} = 0.5$ means localization errors are prioritized.

3. Specialization of Predictors: Each bounding box predictor is assigned based on the highest Intersection over Union (IoU) with the ground truth, allowing the model to specialize in predicting certain sizes or classes of objects.

Training Procedure

1. Epochs and Learning Rate: The model is trained for approximately 135 epochs with a learning rate schedule that starts low, gradually increases, and then decreases:



2. Batch Size and Regularization: A batch size of 64 is used, along with dropout (rate = 0.5) and data augmentation techniques to prevent overfitting.

YOLO Model Limitations

1. Spatial Constraints

- Bounding Box Predictions: Each grid cell can predict only two bounding boxes and one class.

- Impact: This restriction limits the model's ability to detect nearby objects, especially in crowded scenes.

2. Challenges with Small Objects

- Group Detection: The model struggles with small objects that appear in clusters (e.g., flocks of birds).

- Generalization Issues: YOLO has difficulty adapting to new or unusual object shapes and aspect ratios due to its training data.

3. Coarse Features

- Downsampling Layers: The architecture includes multiple downsampling layers, leading to coarse features for bounding box predictions.

- Effect: This can reduce the model's precision in detecting smaller objects.

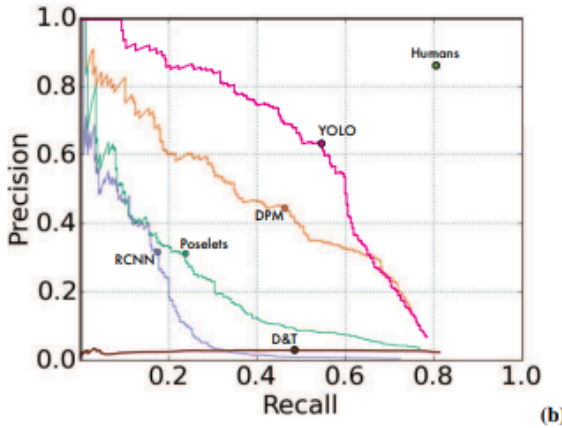
4. Loss Function Limitations

- Equal Weighting of Errors: The loss function treats errors in small and large bounding boxes equally.

- Impact on IOU: A small error in a large box is less critical, while a small error in a small box significantly affects the Intersection over Union (IoU) metric.

5. Main Source of Error

- Incorrect Localizations: The primary issue leading to detection errors is incorrect localizations of objects.



(a) Picasso Dataset precision-recall curves.

	VOC 2007	Picasso		People-Art
	AP	AP	Best F_1	AP
YOLO	**59.2**	**53.3**	**0.590**	**45**
R-CNN	54.2	10.4	0.226	26
DPM	43.2	37.8	0.458	32
Poselets [2]	36.5	17.8	0.271	
D&T [4]	-	1.9	0.051	

(b) Quantitative results on the VOC 2007, Picasso, and People-Art Datasets. The Picasso Dataset evaluates on both AP and best F_1 score.

Figure 5: Generalization results on Picasso and People-Art datasets.

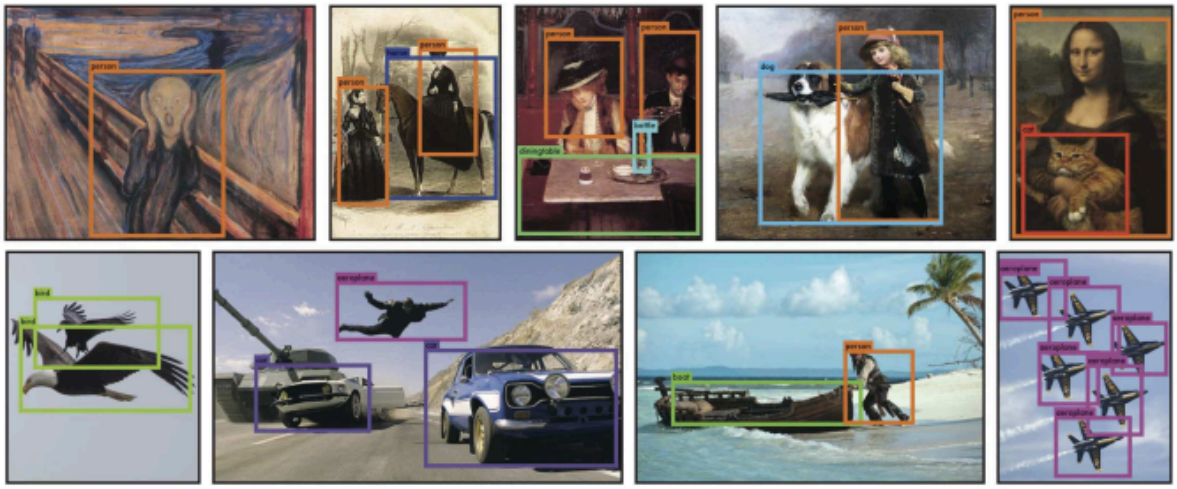


Figure 6: Qualitative Results. YOLO running on sample artwork and natural images from the internet. It is mostly accurate although it does think one person is an airplane.