YOLO

You Only Look Once

By,
Shruti Shah
Vinayak Kundu
Abhay Mathur
Aashish Charaya
Devansh Rathor
Nidhi Tiwari
For, Computer Vision Group Assignment

What is YOLO?

- YOLO, is a state-of-the-art, real-time object object detection and image segmentation model.
- Developed by Joseph Redmon and Ali Farhadi at the University of Washington.
- Launched in 2015, YOLO quickly gained popularity for its high speed and accuracy.

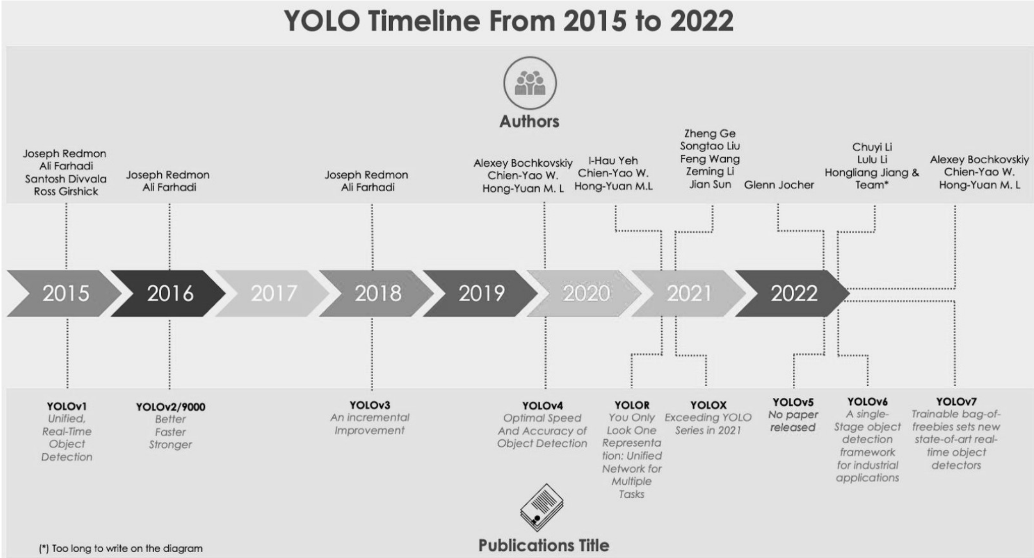
How does YOLO work?

- YOLO is an AI framework that supports multiple computer vision tasks.
- The framework can be used to perform detection, segmentation, obb, classification, and pose estimation.
- YOLO applies a single neural network to the full image.
- This network divides the image into regions and predicts bounding boxes and probabilities for each region.
- These bounding boxes are weighted by the predicted probabilities.
- These detections are thresholded to only see the highest scoring ones.

Where is YOLO used?

- **Security:** Monitors live feeds and identifies potential threats.
- **Healthcare:** Assists doctors by analyzing medical images in real-time.
- **Agriculture:** Scouts fields for crops, weeds, and diseases.
- **Self-Driving Cars:** Detects objects on the road like cars and pedestrians.
- **Robotics:** Enables robots to grasp objects and navigate their surroundings.

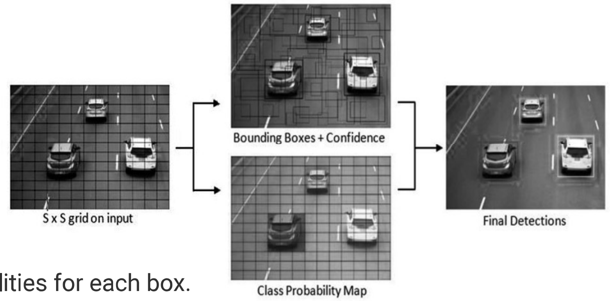
History of YOLO



Methodology & Architecture

Methodology

- Grid-based Detection: Divide image into grid cells.
- Single Forward Pass: Detect objects in one go.
- Bounding Box Prediction: Predict bounding boxes for each cell.
- Class Prediction: Assign class probabilities for each box.
- Confidence Score: Rate likelihood of object presence.
- Non-max Suppression: Remove duplicate detections.

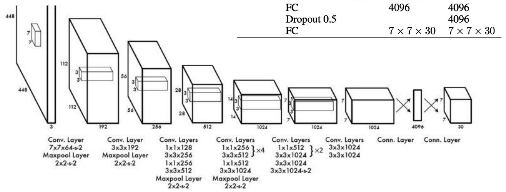


Architecture of YOLOv1

- First iteration of YOLO.
- Consisted of 24 convolution layers, 4 max-pooling layers and 2 fully connected layers at the end.
- Input size being 448*448 image.
- First Convolution layer of size 7 by 7
- Rest convolution layer are combinations of 1*1 and 3*3 size masks
- The Architecture uses Leaky-ReLu as activation function.
- Output for each class would be: (x,y,width,height,confidence)
- All the versions calculate 3 types of errors:
  - localization loss : Coordinates error
  - confidence loss : Probability calculation error
  - classification loss : Type of Class error

	Type	Filters	Size/Stride	Output
	Conv	64	7 × 7 / 2	224 × 224
	Max Pool		2 × 2 / 2	112 × 112
	Conv	192	3 × 3 / 1	112 × 112
	Max Pool		2 × 2 / 2	56 × 56
1×	Conv	128	1 × 1 / 1	56 × 56
	Conv	256	3 × 3 / 1	56 × 56
	Conv	256	1 × 1 / 1	56 × 56
	Conv	512	3 × 3 / 1	56 × 56
	Max Pool		2 × 2 / 2	28 × 28
4×	Conv	256	1 × 1 / 1	28 × 28
	Conv	512	3 × 3 / 1	28 × 28
	Conv	512	1 × 1 / 1	28 × 28
	Conv	1024	3 × 3 / 1	28 × 28
	Max Pool		2 × 2 / 2	14 × 14
2×	Conv	512	1 × 1 / 1	14 × 14
	Conv	1024	3 × 3 / 1	14 × 14
	Conv	1024	3 × 3 / 1	14 × 14
	Conv	1024	3 × 3 / 2	7 × 7
	Conv	1024	3 × 3 / 1	7 × 7
	Conv	1024	3 × 3 / 1	7 × 7
	FC		4096	4096
	Dropout 0.5			4096
	FC		7 × 7 × 30	7 × 7 × 30

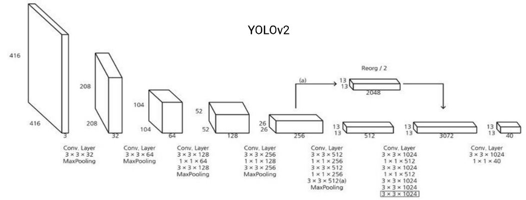
YOLOv1



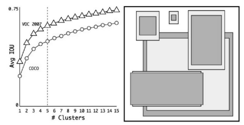
Architecture of YOLOv2

- Next iteration to YOLOv1, multiple changes made to the approach and architecture
- Input Image size is 416*416 instead of 448*448, to get 13*13 feature map
- Uses DarkNet19 that is 19 convolution layers instead of 24.
- 2 Fully connected layers replaced by convolution layers
- Anchoring boxes replaces bounding boxes by above action.
- Few features added like:
  - Batch Normalization
  - Breaking layers into smaller dimension for finer detail detection
  - K-clustering used for accurate box prediction

YOLOv2

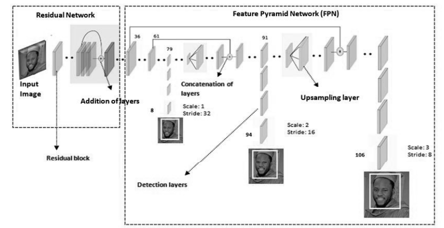


K-Clustering



Architecture of YOLOv3

- Similar to YOLOv2 in terms of working, here are the changes.
- DarkNet 19 is replaced by DarkNet 53, A deeper convolution network with 53 layers
- Uses Feature Pyramid Network (FPN) to get features at multiple scale.

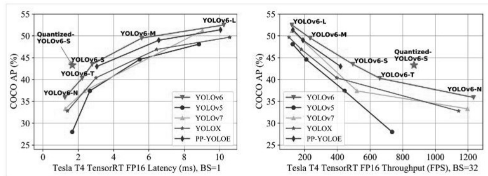


	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 1. **Darknet-53.**

Architecture changes in YOLOv4 to YOLOv7

- YOLOv4 replaced the Darknet 53 backbone with CSPDarkNet 53, which is 29 convolution layers with 27.6 million parameters Instead of incorporating FPN, it used PANnet for parameter aggregation for different level detection.
- YOLOv5, the one which wasn't documented like YOLOv1-4, first model based on pytorch, contained 4 sizes:
  - YOLOv5s (smallest)
  - YOLOv5m
  - YOLOv5l
  - YOLOv5x (largest).
- YOLOv6 and v7 were more improved versions, taking object detection to industrial levels



Demonstration

- [Weapon Detection Dataset & Deployment]
- [Weapon Detection]

Advantages & Limitations

Advantages

1. **Speed:** YOLO is known for its real-time processing capabilities, making it suitable for applications requiring rapid object detection.
2. **Simplicity:** Its single-stage architecture simplifies the process of object detection by directly predicting bounding boxes and class probabilities.
3. **Objectiveness:** YOLO considers the entire image at once, enabling it to detect objects in context and reducing false positives.
4. **Efficiency:** YOLO processes images in a single pass through the network, making it computationally efficient compared to other object detection methods.

Limitations

1. **Localization Accuracy:** YOLO may struggle with accurately localizing small or closely packed objects due to its coarse grid output.
2. **Lower Recall:** It may miss detecting small objects or objects with low contrast in cluttered scenes compared to two-stage detectors.
3. **Class Imbalance:** YOLO may face challenges in scenarios with significant class imbalance, affecting the accuracy of less frequent classes.
4. **Fixed Grid Size:** The fixed grid structure of YOLO can limit its ability to detect objects at varying scales effectively.

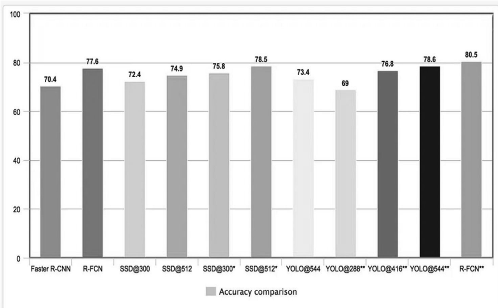
Comparative Analysis

Comparing YOLO with other models

FEATURES	**YOLO**	**FASTER R-CNN**	**SSD(Single Shot Multibox Detector)**	**RetinaNet**
TYPE	Single-stage detector	Two-stage detector	Single-stage detector	Single-stage detector
PROCESSING	Employs a single CNN to analyze the entire image at once	Two step process uses Regional Proposal Network(RPN) and Fast R-CNN detector	Analyzes the image by dividing it into grid of cells	Divides image into grid,predicts per cell using multi-scale features
KEY ELEMENTS	Single CNN predicts bounding boxes and class probabilities	RPN proposes regions and Faster R-CNN classifies objects and refines boxes	Uses multi-scale feature maps and default boxes	Improved rare object accuracy with focal loss and multi-scale detection with feature pyramid network(FPN)

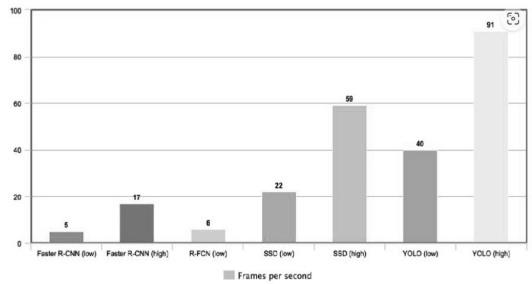
Accuracy comparison

- This graph compares the performance (mAP) of SSD, YOLO, Faster R-CNN and other object detection models on the PASCAL VOC 2012 dataset.
- For both SSD and YOLO, using higher resolution input images (e.g., 512x512 vs 300x300 for SSD) leads to better accuracy (higher mAP).
- At comparable image resolutions (e.g., 300x300), SSD generally achieves higher mAP compared to YOLO.



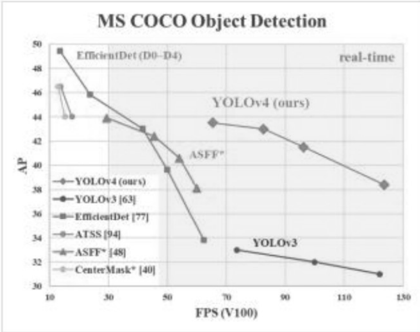
Real Time Object Detection

- The graph highlights a general trend where YOLO models are the fastest (highest FPS), followed by SSD, R-FCN, and Faster R-CNN.
- This is because YOLO utilizes a single-stage detection approach, making it computationally lighter compared to the two-stage approaches used by Faster R-CNN and R-FCN.
- Higher resolution input images lead to a decrease in FPS.



Speed vs. Accuracy Trade-Off in Real-Time Usage

- The graph represents comparison between different object detection models based on their mAP on the MS COCO dataset
- YOLO models (YOLOv4, YOLOv3, YOLO) achieve a good balance between speed and accuracy (mAP between 44% and 55%).
- EfficientDet models (EfficientDet-D7, EfficientDet-D0) offer higher mAP (around 59%) compared to YOLO models but may be slower.
- The real-time model (EfficientDer(DI-04)) achieves an mAP of 44%, similar to YOLO models. It likely offers faster performance than EfficientDet models but with lower accuracy.
- AVCVGG shows the highest mAP (around 76%) but may not be suitable for real-time applications due to its complex architecture. This model prioritizes accuracy over speed.



Thank You