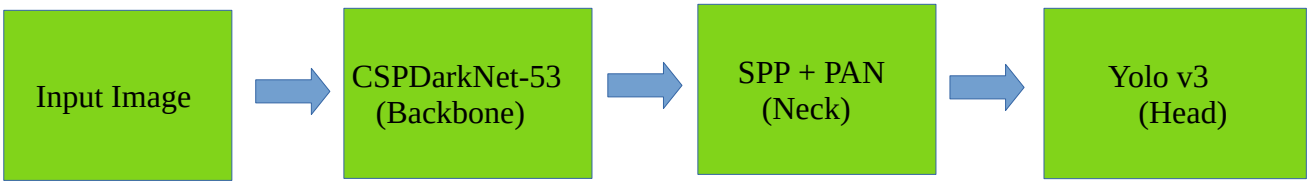
**Name : Panneerselvam N**
**Role : Research Intern – Inspect**
**Date : 03/06/2021**

# You Only Look Once(YOLO v4)

Yolo v4 is an Improved version Yolo v3 in terms of Accuracy and Speed.

**Blocks in Yolo v4:**

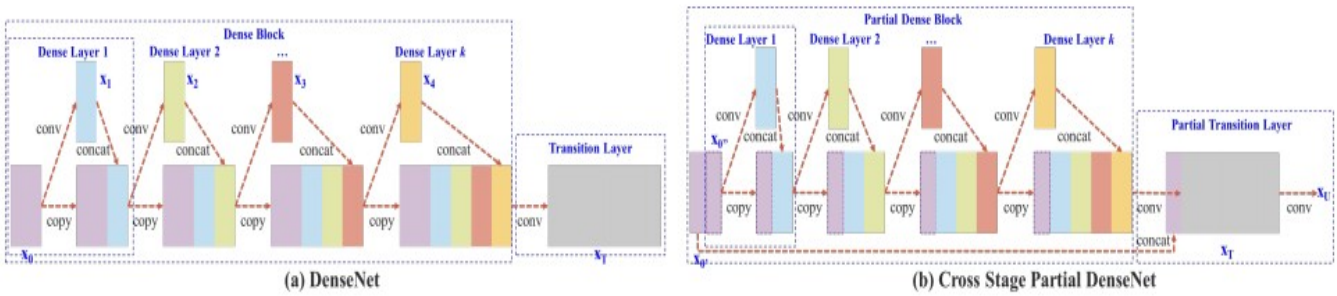



**Explanation:**

**Cross Stage Partial Darknet-53(CSPDarknet-53):**

**CSPDarknet-53** used as a Backbone for Yolo v4. It modified version Darknet-53 in Yolo v3.

In CSPDarknet-53 ,Cross Stage Partial DenseNet used instead of Residual Networks. DenseNet is a advanced version of ResNet. In DenseNet every layer outputs are added with next layers inputs as skip connections.




Cross Stage Partial Dense is little modified version of DenseNet.

**Working:**

- Initially Feature maps are separated into two parts.
- Second Part is given to convolution layer, Output of convolutional layer is concatenated with that input, this process has continued on all layers.
- Final Layer’s output is concatenated with initial input's first part.

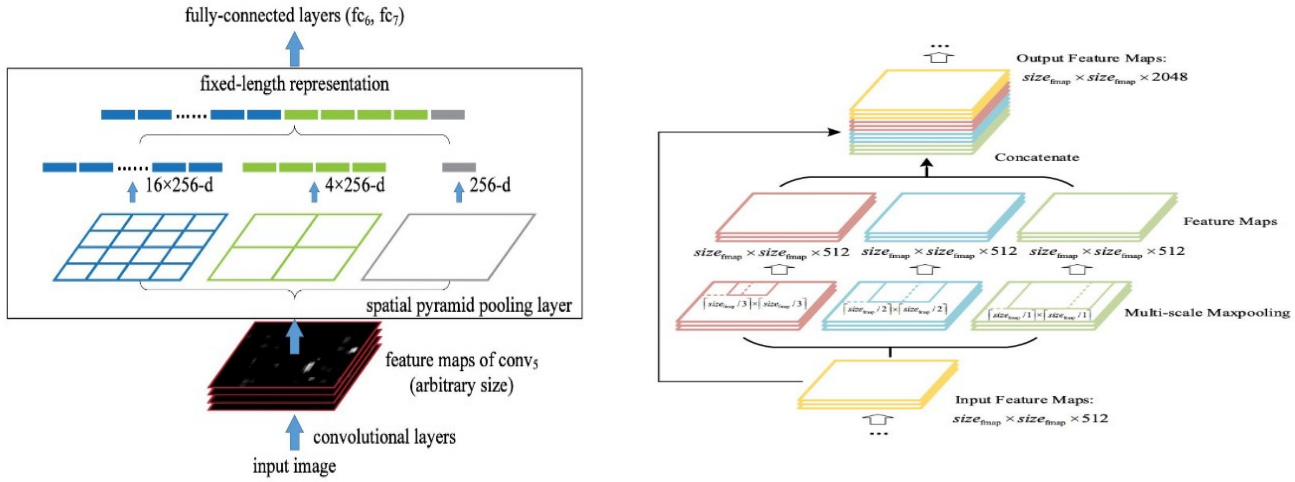
**Advantages :**

- Yolo v4 avoids gradient vanishing and exploding problem by using this Cross Stage Partial DenseNet(CSPD)

**Additional Block :**

It is a first part of Neck.

**Spatial Pyramid Pooling(SPP)** used as additional block for Yolo v4. In Yolo v4 modified SPP is used because of normal SPP has fully connected layer but in yolo v4 has only Convolutional layers.




In above fig, Right side image shows modified version of SPP.

CSPDarknet-53 extracts features and given the features to Spatial Pyramid Pooling .

**Working :**

- Output of CSPDarnet-53 has 1024 channel, its is resized 512 using 1x1(Point wise convolotional)
- First Input features are duplicated into 3 parts.
- Next,Three Multi-scale Pooling applied to this duplicated feature map.
- Feature maps are maintained same scaling size by using padding technique.
- Finally Three poling layers output and initial input of pooling layer are concatenated with single feature map, which has channel size of 2048.

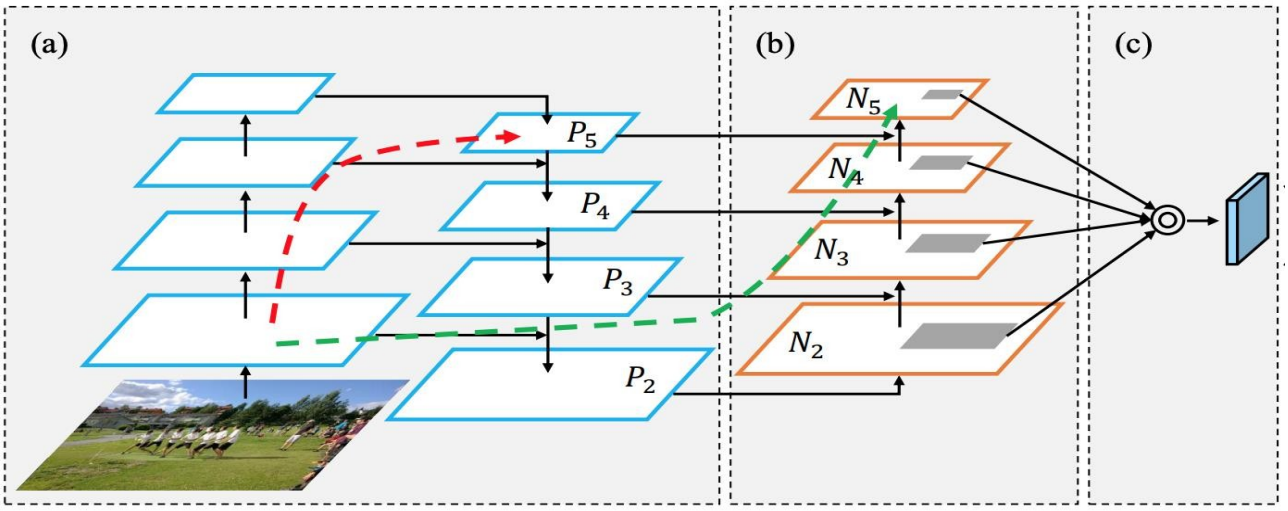
**Advantages :**

- Spatial Pyramid Network gives Fixed size of output where the input is any size.

- SPP uses three multi-scale max pooling , so the important feature maps easyly separated from low level features.
- By using concatenation of input features , the low level features also added to feature map. Because low level features are very important to detect small objects.

**Feature Aggregation :**

It is a second part of Neck block.
**Path Aggregation Network(PAN)** is used as Feature Aggregation Network of Yolo v4. It is improved version Feature Pyramid Network .




Above image shows the PANet architecture.
**Working:**

- Normal FPN merges the high level features to low level features in Top-Bottom opration. So the features of middel and low are modified . This gives difficulties for detect small objects.

- For overcome above problem, the PANet comes with additional layer pyramid of Bottom-top , this gives very efficient low and middle level features.

- Finally the all pyramid features are concatenated with Region of Interest Align(ROIAlign) .

**Head of Yolo v4:**

Convolutional head is used to generate bounding box and classification as like Yolov3.

The output is in the form of Three dimensional vector.

The shape of detection kernel is 1 x 1 x (B x (5 + C)). Here B is the number of bounding boxes a cell on the feature map can predict, '5' is for the 4 bounding box attributes and one object confidence and C is the no. of classes.

**New Concepts :**
In Yolo v4 Research Paper introduced two new concepts the following,

- Bag of Freebies
- Bag of Specials

This Bags contains many methods and techniques to optimize Yolo v4 algorithm.

**Bag of Freebies:**

**For backbone:** CutMix and Mosaic data augmentations, DropBlock regularization, Class labelsmoothing.

**For detector (neck and head):** Complete Intersection over Union (CIoU-loss), Cross-mini–Batch Norm (CmBN), DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training (SAT), Eliminate grid sensitivity, using multiple anchors for a single ground truth, Cosine annealing scheduler, Optimal hyper-parameters, Random training shapes.

**Bag of Specials :**

**For backbone:** Mish activation, Cross-stage partial connections (CSP), Multi-input weighted residual connections (MiWRC).

**For detector (neck and head):** Mish activation, Spatial Pyramid Pooling block (SPP-block), Spatial Attention Module (SAM-block), Path Aggregation Network (PANet), Distance Intersection over Union Non-Maximum Suppression (DioU-NMS).

**Performance:**
Combination of CSPDarknet-53 -SPP-PANet gives good accuracy with compare to other architectures.

Table 6: Using different classifier pre-trained weightings for detector training (all other training parameters are similar in all models) .

| Model (with optimal setting) | Size | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|
| **CSPResNeXt50-PANet-SPP** | 512x512 | 42.4 | 64.4 | 45.9 |
| **CSPResNeXt50-PANet-SPP** (BoF-backbone) | 512x512 | 42.3 | 64.3 | 45.7 |
| **CSPResNeXt50-PANet-SPP** (BoF-backbone + Mish) | 512x512 | 42.3 | 64.2 | 45.8 |
| **CSPDarknet53-PANet-SPP** (BoF-backbone) | 512x512 | 42.4 | 64.5 | 46.0 |
| **CSPDarknet53-PANet-SPP** (BoF-backbone + Mish) | 512x512 | 43.0 | 64.9 | 46.5 |

Table 9: Comparison of the speed and accuracy of different object detectors on the MS COCO dataset (test-dev 2017). (Real-time detectors with FPS 30 or higher are highlighted here. We compare the results with batch=1 without using tensorRT.)

| Method | Backbone | Size | FPS | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|---|---|
| **YOLOv4: Optimal Speed and Accuracy of Object Detection** | | | | | | | | | |
| **YOLOv4** | CSPDarknet-53 | 416 | 54 (P) | 41.2% | 62.8% | 44.3% | 20.4% | 44.4% | 56.0% |
| **YOLOv4** | CSPDarknet-53 | 512 | 43 (P) | 43.0% | 64.9% | 46.5% | 24.3% | 46.1% | 55.2% |
| **YOLOv4** | CSPDarknet-53 | 608 | 33 (P) | **43.5%** | **65.7%** | **47.3%** | **26.7%** | **46.7%** | 53.3% |
| **CenterMask: Real-Time Anchor-Free Instance Segmentation [40]** | | | | | | | | | |
| CenterMask-Lite | MobileNetV2-FPN | 600× | 50.0 (P) | 30.2% | - | - | 14.2% | 31.9% | 40.9% |
| CenterMask-Lite | VoVNet-19-FPN | 600× | 43.5 (P) | 35.9% | - | - | 19.6% | 38.0% | 45.9% |
| CenterMask-Lite | VoVNet-39-FPN | 600× | 35.7 (P) | 40.7% | - | - | 22.4% | 43.2% | **53.5%** |
| **Enriched Feature Guided Refinement Network for Object Detection [57]** | | | | | | | | | |
| EFGRNet | VGG-16 | 320 | 47.6 (P) | 33.2% | 53.4% | 35.4% | 13.4% | 37.1% | 47.9% |
| EFGRNet | VG-G16 | 512 | 25.7 (P) | 37.5% | 58.8% | 40.4% | 19.7% | 41.6% | 49.4% |
| EFGRNet | ResNet-101 | 512 | 21.7 (P) | 39.0% | 58.8% | 42.3% | 17.8% | 43.6% | 54.5% |
| **Hierarchical Shot Detector [3]** | | | | | | | | | |
| HSD | VGG-16 | 320 | 40 (P) | 33.5% | 53.2% | 36.1% | 15.0% | 35.0% | 47.8% |
| HSD | VGG-16 | 512 | 23.3 (P) | 38.8% | 58.2% | 42.5% | 21.8% | 41.9% | 50.2% |
| HSD | ResNet-101 | 512 | 20.8 (P) | 40.2% | 59.4% | 44.0% | 20.0% | 44.4% | 54.9% |
| HSD | ResNeXt-101 | 512 | 15.2 (P) | 41.9% | 61.1% | 46.2% | 21.8% | 46.6% | 57.0% |
| HSD | ResNet-101 | 768 | 10.9 (P) | 42.3% | 61.2% | 46.9% | 22.8% | 47.3% | 55.9% |
| **Dynamic anchor feature selection for single-shot object detection [41]** | | | | | | | | | |
| DAFS | VGG16 | 512 | 35 (P) | 33.8% | 52.9% | 36.9% | 14.6% | 37.0% | 47.7% |
| **Soft Anchor-Point Object Detection [101]** | | | | | | | | | |
| SAPD | ResNet-50 | - | 14.9 (P) | 41.7% | 61.9% | 44.6% | 24.1% | 44.6% | 51.6% |
| SAPD | ResNet-50-DCN | - | 12.4 (P) | 44.3% | 64.4% | 47.7% | 25.5% | 47.3% | 57.0% |
| SAPD | ResNet-101-DCN | - | 9.1 (P) | 46.0% | 65.9% | 49.6% | 26.3% | 49.2% | 59.6% |
| **Region proposal by guided anchoring [82]** | | | | | | | | | |
| RetinaNet | ResNet-50 | - | 10.8 (P) | 37.1% | 56.9% | 40.0% | 20.1% | 40.1% | 48.0% |
| Faster R-CNN | ResNet-50 | - | 9.4 (P) | 39.8% | 59.2% | 43.5% | 21.8% | 42.6% | 50.7% |
| **RepPoints: Point set representation for object detection [87]** | | | | | | | | | |
| RPDet | ResNet-101 | - | 10 (P) | 41.0% | 62.9% | 44.3% | 23.6% | 44.1% | 51.7% |
| RPDet | ResNet-101-DCN | - | 8 (P) | 45.0% | 66.1% | 49.0% | 26.6% | 48.6% | 57.5% |
| **Libra R-CNN: Towards balanced learning for object detection [58]** | | | | | | | | | |
| Libra R-CNN | ResNet-101 | - | 9.5 (P) | 41.1% | 62.1% | 44.7% | 23.4% | 43.7% | 52.5% |
| **FreeAnchor: Learning to match anchors for visual object detection [96]** | | | | | | | | | |
| FreeAnchor | ResNet-101 | - | 9.1 (P) | 43.1% | 62.2% | 46.4% | 24.5% | 46.1% | 54.8% |
| **RetinaMask: Learning to Predict Masks Improves State-of-The-Art Single-Shot Detection for Free [14]** | | | | | | | | | |
| RetinaMask | ResNet-50-FPN | 800× | 8.1 (P) | 39.4% | 58.6% | 42.3% | 21.9% | 42.0% | 51.0% |
| RetinaMask | ResNet-101-FPN | 800× | 6.9 (P) | 41.4% | 60.8% | 44.6% | 23.0% | 44.5% | 53.5% |
| RetinaMask | ResNet-101-FPN-GN | 800× | 6.5 (P) | 41.7% | 61.7% | 45.0% | 23.5% | 44.7% | 52.8% |
| RetinaMask | ResNeXt-101-FPN-GN | 800× | 4.3 (P) | 42.6% | 62.5% | 46.0% | 24.8% | 45.6% | 53.8% |
| **Cascade R-CNN: Delving into high quality object detection [2]** | | | | | | | | | |

Above image shows the comparison of different architecture with coco dataset. Yolo v4 gives best performance with accuracy of 41.2 AP and 54 FPS speed on MSCOCO Dataset.