## Convolutional Neural Networks – an overview and terminology

There are number of tasks that CNN addresses:

* Classification – assigning label to the image.
* Localization – localizing an object at an image. It is usually combined with classification. It is addressed to localize a single object within an image.
* Object detection – detecting different categories of objects within the same image. It is both classification and localization task, however the number of objects within the same image in unknow. In case of videos, object detection task is performed on each frame of the video.
* Segmentation – determining the shape of a detected object. It creates a pixel-wise mask for the objects in an image which helps models to understand the shape of objects and their position in the image at a more granular level. Segmentation is divided into two categories:
  + Semantic Segmentation - In semantic segmentation, different instances of the same class are represented by the same colours.
  + Instance Segmentation – In instance segmentation, different instances of the same class are represented by different colours.

More sophisticated tasks include: Human pose estimation, image captioning.

The most popular **Object Detection Datasets** for testing performance:

* PASCAL VOC – contains 20 classes. It was used in early stage of object detection, now is too easy.
* COCO Microsoft – the most popular dataset for testing. 80 classes, 200 000 images. It uses 0.5 IoU based metrics as a standard.
* OpenImagesV4 – the largest dataset containing 500 classes, and 1,743,000 images. It was released in 2018.
* Many specialized datasets, including pedestrian datasets (INRIA), face datasets.

Metrics for object detection include evaluation of classification and detection tasks. The classification task only evaluates the probability of the class object appearing in the image. The object detection task localizes the object with a bounding box associated with its corresponding confidence score. The confidence score represents probability that the bounding box detected an object. The most popular method for evaluation confidence score is Intersection over Union (IoU).

**Intersection over union (IoU)** is the ratio of the overlapping area of **ground truth** and predicted bounding box to the total area. A ground-truth bounding boxes is a hand labelled area from the testing set that specify where in the image our object is. The bounding boxes are simply the (x, y)-coordinates of the object in the image. An Intersection over Union score > 0.5 is normally considered a “good” prediction. IoU is used as a confidence threshold in R-CNN, Faster R-CNN, YOLO.

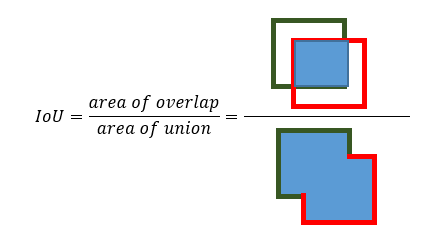


Figure 1 Intersection over Union

Detection predictions uses IoU as a threshold, usually 0.5. The predictions are classified into **True Positives (TP)**, **False Negatives (FN)**, and **False Positives (FP)**. The predicted object is True Positive if the predicted class is correct and the overlap with the ground truth is greater than a threshold. The predicted object is marked False Positive if it has overlap less than that threshold or same object instance is detected again, or it is misclassified, or a background is predicted as an object instance. False Negative is when the object exists, but a model does not predict it. **True Negative (TN)** occurs when empty boxes are correctly detected as non-object. Detection model predicts thousands of True Negative, making it not useful for object detection.

**Accuracy** is a metric commonly used in classification tasks. It is the percentage of correctly predicted examples out of all predictions: a sum of TP and TN over a sum of all predictions. Accuracy is misleading if dataset is imbalanced. Detection datasets are imbalanced. Hence it should not be used as metric for detectors.

**Precision** is the probability of the predicted bounding boxes matching actual ground truth boxes, also referred to as the positive predictive value. Precision scores range from 0 to 1. A high precision means that most detected objects match ground truth objects.

**Recall** is the true positive rate, also referred to as sensitivity. It measures the probability of ground truth objects being correctly detected. Recall ranges from 0 to 1 where a high recall score means that most ground truth objects were detected.

​

**Precision/Recall curve** metric is used when classes are heavily imbalanced. The precision/recall curve plots recall on the x-axis and precision on the y-axis, where each point in the curve represents recall and precision values for a certain confidence value. There is a trade-off between Precision and Recall that depends on the confidence threshold. With a high confidence threshold, the model is more severe to predict positive, hence there will be less False Positive and more False Negative. This improves Precision but decreases Recall.

**Average Precision (AP)** evaluates the performance of object detectors. It is a single number metric calculated by averaging precision across recall values from 0 to 1. AP is an area under the Precision-Recall Curve. In the case of a multi-class object detector precision and recall are computed for each object class separately. The metric **mean Average Precision (mAP)** is the mean of the average precision of each class.

On COCO dataset competition uses the flavour of Average Precision. They are denoted as AP@α – i.e., AP50 and A75 means AP at IoU threshold of 0.5 and 0.75 respectively. AP@[.50:.5:.95] denotes that AP is calculated for a given class as an average over 10 different IoU ranging from 50% to 95% at 5% step-size.

**Average Recall (AR)** is evaluated as an average recall over IoU thresholds from 0.5 to 1. This metric is mainly used for evaluating models that are optimized for high recall, meaning low False Negative cases. [1]

**Gradient Descent** is an iterative optimization algorithm for finding the minimum of a function. It minimises a loss function of the neural network. To achieve its goal, it computes the slope (gradient) and moves-in the opposite direction of the slope increase from the current point. In case of Neural Networks, weights are updated every iteration of Gradient Descent. Gradient Descent or Batch Gradient Descent uses he whole training data per iteration**. Stochastic Gradient Descent** use only one item or a subset of training data per iteration. If a subset is used, an algorithm is also called **Mini-Batch Gradient Descent**.

The terms **normalization** and **standardization** are sometimes used interchangeably, but they usually refer to different things. Normalization usually means to scale a variable to have values between 0 and 1, while standardization transforms data to have a mean of zero and a standard deviation of 1. Machine learning algorithms like linear regression, logistic regression, neural network, etc. that use gradient descent as an optimization technique require data to be scaled.

**Batch normalization** is a technique for training very deep neural networks that standardizes the inputs to a layer for each mini-batch. Inputs within a Neural Networks are results of activation function. Layer’s inputs are normalized by using the mean and variance of the values in the current mini-batch. Batch normalization optimizes neural network training by improving the speed, performance, and stability. It improves convergence, eliminates need for other form of regularization (i.e., weight initialization), allows to reduce some of dropout without overfitting.

**Logistic regression** – logistic regression is on of the techniques used for classification. The method of logistic regression is a binary classifier by nature. However, it can also be used for multi-class classifications. The **softmax regression** (**multinomial logistic regression**) is a form of logistic regression that normalizes an input value into a vector of values that follows a probability distribution whose total sums up to 1. The output values are between the range [0,1] which enables to avoid binary classification and accommodate many classes. Whether to use softmax regression or logistic regressions depends on the nature of the classes. The method of softmax regression is suitable if the classes are mutually exclusive and independent, as assumed by the method.

The most popular **Image Classification models** (backbones) used in object detection are VGG16, GoogleNet (also codenamed Inception\_v1), ResNet and Darknet.

VGG16 uses 16 layers convolutional layers with multiple 3×3 kernel-sized filters and Max Pooling layer of size 2 × 2. The model achieves 92.7% top-5 test accuracy in ImageNet.

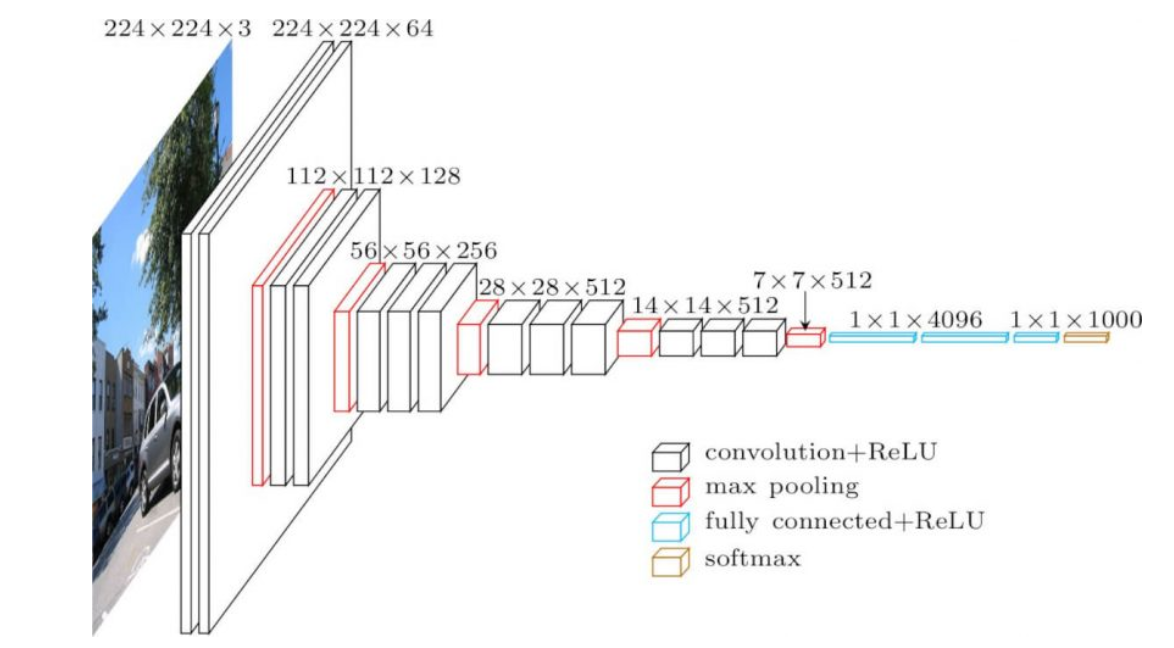


Figure 2 VGG16 architecture. [2]

He et al, the authors of ResNet, came to conclusion that simple adding more convolutional layers to a model leads to higher error. Evidence shows that the best ImageNet models using traditional convolutional and fully-connected layers typically contain between 16 and 30 layers. The main reason is vanishing gradient and degradation problem.

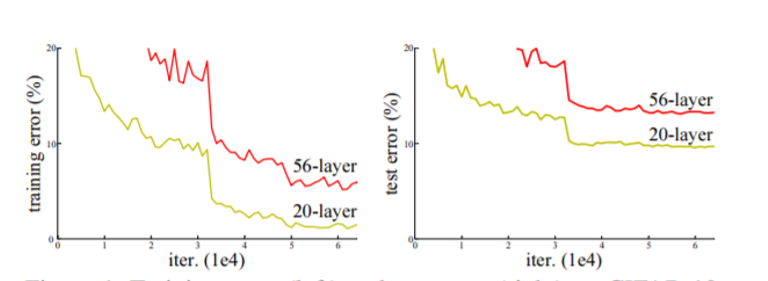


Figure 3 Training error (left) and test error (right) onCIFAR-10 with 20-layer and 56-layer “plain” networks. The deeper network has higher training error, and thus test error. [3]

Vanishing gradient problem is solved by normalized initialization and intermediate normalization layers (batch normalization), which enable networks with tens of layers to start converging for stochastic gradient descent (SGD) with backpropagation.

Degradation problem occurs when the network depth increases, causing accuracy to get saturated and then degrades rapidly.

To solve the degradation problem, He et al introduced a new neural network layer — The Residual Block.

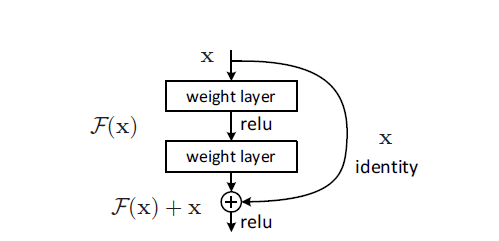


Figure 4 The Residual Block [3]

The most important modification to CNN is adding shortcut connections. Shortcut connections are those skipping one or more layers. In case of ResNet, the shortcut connections simply perform identity mapping, and their outputs are added to the outputs of the stacked layers. Identity mapping just takes the layer outputs from earlier layers and passes it down.

He et al experimented with a CNN of more than 1000 layers. The training error is similar to the ResNet of 110 layers. However, the testing result is worse than 110-layer network. [3]

Darknet was introduced by Reymond et al, authors of YOLO. The purpose of developing Darknet was to increase the speed of CNN classifier. Darknet-19 uses 19 convolutional layers with 5 Max Pooling and batch normalization. The main achievement was a drop in numbers of floating operation to 5.58 billion and keeping a high accuracy - 91.2% top-5 accuracy on ImageNet. Darknet-53, used from YOLOv3, has 53 convolutional networks and shortcut connections. It is much more powerful but slower than Darknet-19 but still more efficient than ResNet-101 or ResNet-152. [4]

Table 1 Comparison of backbones. Accuracy, billions of operations, billion floating point operations per second, and FPS for various networks.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Backbone** | **Top-1** | **Top-5** | **Bn Ops** | **BFLOP/s** | **FPS** |
| Darknet-19 | 74.1 | 91.8 | 7.29 | 1246 | 171 |
| ResNet-101 | 77.1 | 93.7 | 19.7 | 1039 | 53 |
| ResNet-152 | 77.6 | 93.8 | 29.4 | 1090 | 37 |
| Darknet-53 | 77.2 | 93.8 | 18.7 | 1457 | 78 |

There are two main categories of object detectors: **two-stage** and **one-stage** detectors. The R-CNN family are the main representatives of two-stage object detectors. The first version has the following workflow: take an image as the input, extract region proposals, run CNN for each proposal to compute feature map, and then classify each region. The next versions improved this workflow by adding i.e., Region of Interest Pooling Layer (RoI), Anchor Boxes, Region Proposal Network [5]. The main idea stays the same, which is to generate region proposal in the first stage and then use regression (detection task) and classification methods in the second stage. The disadvantage of two-stage detectors is their complexity, resulting in the long time spent in handling different components. They are not useful for real-time applications.

One stage detector i.e., YOLO or SSD, achieve high inference speed. They are stripped out from the first stage. Bounding Boxes are predicted directly from input images, without region proposal step. They are time efficient and can be used for real-time object detection tasks. [6]

## YOLO

Adarsh et al show how deep learning algorithms are growing rapidly with an improved object detection performance. In their paper the authors present an overview of object detection methods by including two classes of object detectors. In two stage detector covered algorithms are RCNN, Fast RCNN, and Faster RCNN, whereas in one stage detector YOLO v1, v2, v3, and SSD are covered. Two stage detectors focus more on accuracy, whereas the primary concern of one stage detectors is speed. [7]

### YOLO version 1

Radmond et al introduced YOLO, which acronym stands for You Only Look Once, in 2016. The authors of YOLO incorporated core ideas from previous work on MultiGrasp detection system used for robotic grasp. [8] The two main ideas were to divide an image into grid cells that contain bounding-boxes, and to use single-stage regression to bounding-boxes instead of sliding window or region proposal techniques. By using this approach, a Neural Network simultaneously performs classification and localization in a single step.

Prior to YOLO, the two-stage object detectors were explored. The most advanced, a family of R-CNN detectors, in the first step selects a set of region proposal. Then regression and classification are processed for every single Region of Interest. This approach is computationally expensive. The further works concentrated on optimizing individual components of a large detection pipeline.

The authors of YOLO took a different approach. They use regression to solve the object detection problem. Regression is carried on two variables: spatially separated bounding boxes and associated class probabilities. A single convolutional network simultaneously predicts multiple bounding boxes (localization) and class probabilities for those boxes (classification).

The concept of YOLO is as follows:

* The input image is divided into an S × S grid. A grid cell may contain the centre of an object. If a grid cell holds the centre of an object, that grid cell is responsible for detecting that object. A single grid cell contains no more than one object.
* Perform localization (bounding boxes + confidence):
  + Each cell predicts B bounding boxes. A bounding box contains the following parameter to describe location of it within an image: (x, y, w, h), where (x ,y) are coordinates of the centre of an image, ‘w’ is the width and ‘h’ is the height of an image. Coordinates of the centre of an image are within a grid cell, whereas the width and the hight of an image may expand over the grid cell. That part is similar to a Region of Interest used in R-CNN family detectors.
  + The authors of YOLO added one more parameter to be predicted by a grid cell, which is confidence score. A confidence score is the conditional probability. The condition is that the cell contains an object. If no object exists in that cell, the confidence scores should be zero. Otherwise, the confidence score equals the Intersection over Union (IOU) between the predicted box and the ground truth. The confidence score formula is as follows: Pr(containing an object) x IoU(pred, truth); where Pr = probability and IoU = Intersection over Union.
* Perform classification (class probability map):
* If a grid cell contains an object, it predicts C conditional class probabilities: Pr(Classi|Object). Otherwise, a grid cell belongs to the background class.
* Final detection:
  + One image contains S×S×B bounding boxes, each box corresponding to 4 location predictions, 1 confidence score, and C conditional probabilities for object classification. The total prediction values for one image are S × S × (B ∗ 5 + C), which is the tensor shape of the final convolutional layer of the model. YOLO is trained on a loss function that directly corresponds to detection performance and the entire model is trained jointly.

YOLO runs 45 FPS, which is faster than real-time (30 FPS is considered to be a real-time). It may be connected to a webcam and it maintains real-time performance, including the time to fetch images from the camera and display the detections. It processes every image from video streaming individually. Because of its speed, it is possible to simulate a tracking system of objects on a video streaming, when objects move around.

The speed goes with a price of accuracy. On PASCAL 2007 it reached 63 mAP, which was lower than Fast R-CNN and Faster R-CNN. It lacks especially with detecting of small objects. On other hand, it makes less background errors.

The other limitation of YOLO is due to spatial constraints on bounding box predictions. Each grid cell can have only one class and uses small number of bounding boxes. In result, YOLO does not detects overlapping or nearby objects. It predicts just one of them. [9]

### YOLO version 2

Radmond et al concentrated on two aspects of improving YOLO model. YOLOv1 was fast, but lag behind other object detectors in terms of localization errors and low recall. The second problem they tackle was about enabling detection systems to classify more categories.

The following solutions were implemented into YOLOv2 to solve problems with localization errors and low recall:

* Batch Normalization – batch normalization is added on all the convolutional layers. It improved the mAP score by more than 2%. Using Batch Normalization allowed to remove dropout from the model without overfitting.
* Higher Resolution – YOLOv2 trains the classifier on ImageNet dataset. For first 10 epoch, images are resized to 448 x 448. This method improves detection of smaller objects. This increased the mAP score by almost 4%. Other detectors, including YOLOv1, uses classifier pre-trained on input images smaller than 256 x 256, and then fine tune the model for detection. If such model is used with higher resolution to detect smaller objects, the model must learn object detection and adjust its classification part to the new input resolution.
* Anchor Boxes – the idea of anchor boxes was incorporated from Faster R-CNN, that uses a set of 9 predefined anchor boxes.
  + In YOLOv2, the size and the number of anchor boxes is automatically found by running k-means clustering on the training set bounding boxes. The k-means algorithm to compute the number and dimensions of the anchor boxes uses IOU scores. It gave a set of 5 anchor boxes as optimum. There are fewer short, wide boxes and more tall, thin boxes.
  + YOLOv2 predicts class and objectness for every anchor box. In YOLOv1 a class was bounded to the grid cell. In effect, YOLOv2 it able to predict overlapping objects.
  + The authors notice that usually objects are at the centre of an image. For that reason, it is good to have a single location at the centre. The input image is 416 instead of 448×448. The output of convolutional layers is 13x13, if used 416 as an input. In that way there is a centre in the feature map.

Radmond et al developed Darknet-19 as backbone for YOLOv2. The aim of developing new classification model was to speed up the process. Darknet-19 contains 19 convolutional layers and 5 maxpooling layers. However, the number of operations to process an image is only 5.58 billion, whereas i.e., VGG-16 require 30.69 billion operations.

For it’s time YOLOv2 was the fastest, and also one of the most accurate object detectors. However, YOLOv2 often struggled with small object detections. [10]

### YOLO version 3

Radmond et al released YOLOv3 that has improved accuracy compared to its predecessor. It was achieved by implementing more deeper convolutional layer and using more anchor boxes. It uses Darknet-53 as its backbone. Darknet-53 has 53 layers, containing residual blocks. It is trained on ImageNet. YOLO v3 uses 9 anchor boxes, three for each scale. Each set is selected using K-Means clustering. Increasing the number of anchor boxes from 5 to 9 results in predicting 10 times more anchor boxes that YOLO v2. Using more anchor boxes and more complex Darknet makes YOLOv3 slower than previous version. However, it increased accuracy.

YOLOv3 detects objects at 3 different scales. The concept is taken from feature pyramid networks. The multilevel detection layers are added after convolutional layers. Radmond et al use upsampling technique. The upsampled layers are concatenated with the previous layers from convolution part. The 13 x 13 layer is responsible for detecting large objects, whereas the 52 x 52 layer detects the smaller objects, with the 26 x 26 layer detecting medium objects. The shape of the detection kernel is 1 x 1. This architecture is also described as composed of two parts: a backbone which is pre-trained on ImageNet and a head which is used to predict classes and bounding boxes of objects.

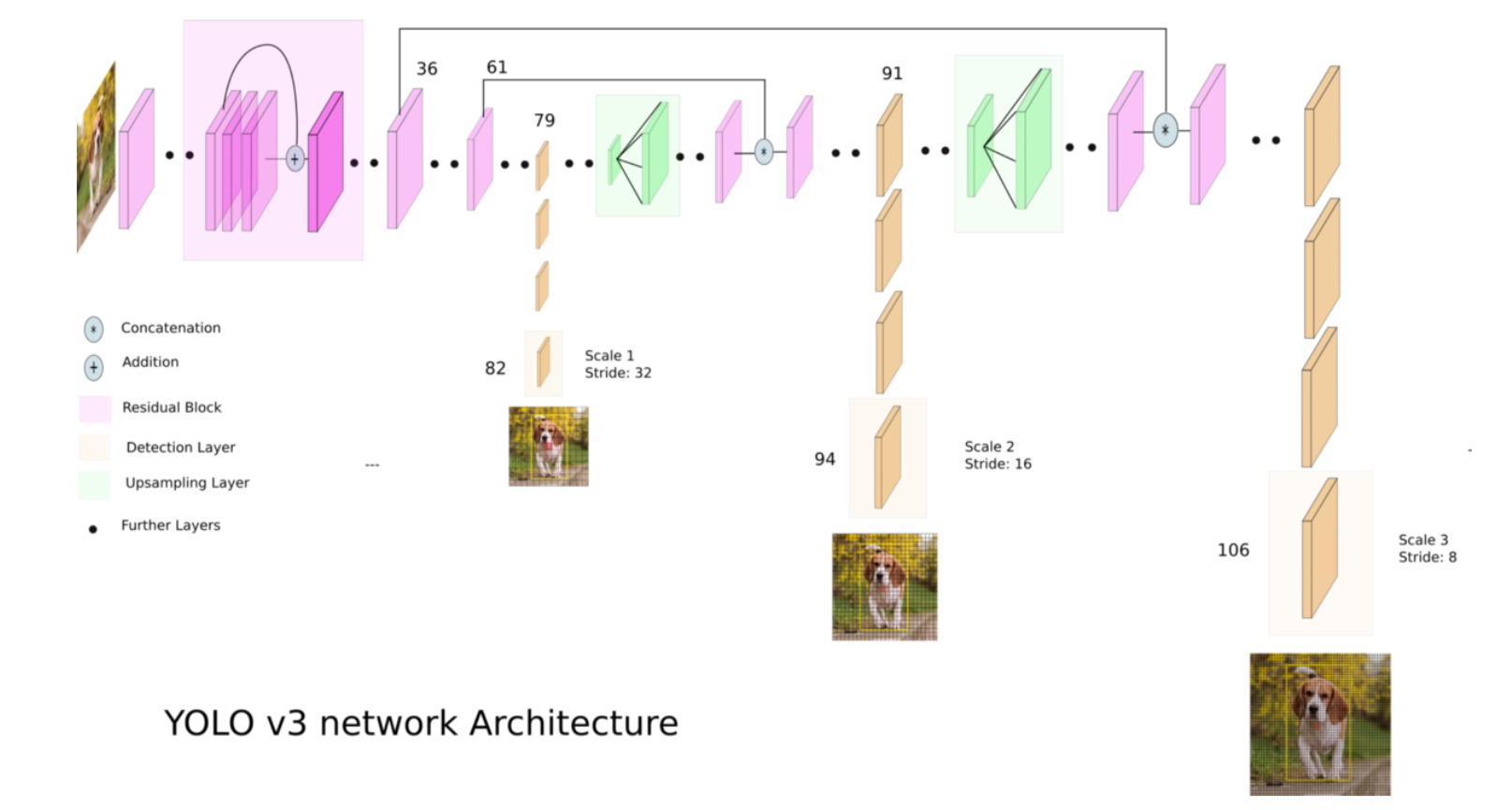


Figure 5 YOLOv3 network architecture (copied form <https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b> )

YOLO v3 performs multilabel classification for objects detected in images. YOLOv2 uses softmax that assumes classes are mutually exclusive. In YOLO v3, object confidence and class predictions are predicted through logistic regression and a threshold is used to predict multiple labels for an object. Classes with scores higher than this threshold are assigned to the box.

Table 2 Average Precision of different object detection models. [4]

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Backbone** | **AP** | **AP50** | **AP75** | **APS** | **APM** | **APL** |
| ***Two-stage method*** | | | | | | | |
| Faster R-CNN+++ | ResNet-101-C4 | 34.9 | 55.7 | 37.4 | 15.6 | 38.7 | 50.9 |
| Fastet R-CNN w FPN | ResNet-101-FPN | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 |
| Faster R-CNN by G-RMI | Inception-ResNet-v2 | 34.7 | 55.5 | 39.7 | 13.5 | 38.1 | 52.0 |
| Faster R-CNN w TDM | Inception-ResNet-v2-TDM | 36.8 | 57.7 | 39.2 | 16.2 | 39.8 | 52.1 |
| ***One-stage methods*** | | | | | | | |
| **YOLOv2** | DarkNet-19 | 21.6 | 44.0 | 19.2 | 5.0 | 22.4 | 35.5 |
| SSD13 | ResNet-101-SSD | 31.2 | 50.4 | 33.3 | 10.2 | 34.5 | 49.8 |
| DSSD513 | ResNet-101-DSSD | 33.2 | 53.3 | 35.2 | 13.0 | 35.4 | 51.1 |
| RetinaNet | ResNet-101-FPN | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 |
| **RetinaNet** | **ResNeXt-101-FPN** | **40.8** | **61.1** | **44.1** | **24.1** | **44.2** | **51.2** |
| **YOLOv3 608x608** | Darknet-53 | 33.0 | 57.9 | 34.4 | 18.3 | 35.4 | 41.9 |

YOLOv3 has improved significantly comparing to its previous version. It is much better on detecting small objects. The figure above shows that RetinaNet performed the best at the time of realising YOLOv3. However, it was almost 4 times slower than YOLOv3. It makes YOLOv3 much better choice for real-time object detection. [4]

### YOLO version 4

The paper announcing YOLOv4 was released in 2020. During the time between developing the new version and the previous one, many new detection models were developed. Those models still are grouped into two-stage and one-stage object detectors. Most of state-of-the-art object detections uses universal features such as batch-normalization and residual-connections.

Bochkovskiy et al stated that the goal of developing YOLOv4 was to create a real-time object detector with high accuracy that operates and is trained on conventional GPU, like 1080 Ti or 2080 Ti.

YOLOv4 uses architecture of his predecessor. The authors of new version concentrated on developing the model by optimizing components within YOLO. They tested many variants. They divided different methods into two categories:

* Bag-of-Freebies – methods used during training of a model for improving the overall accuracy of an object detection model.
* Bag-of-Specials – plugins modules and post-processing methods that can significantly improve the accuracy of object detection but increase the inference cost only by a small amount.

YOLO v4 uses among others:

* Bag of Freebies for backbone or detector: CutMix and Mosaic data augmentation, Self-Adversarial Training (SAT), Random training shapes, DropBlock regularization, CIoU-loss, CmBN.
* Bag of Specials for backbone or detector: Mish activation, Cross Stage Partial connections (CSP), Multiinput Weighted Residual Connections (MiWRC), SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS.

Here are short descriptions of those methods:

* Mosaic data augmentation is a data augmentation method that mixes 4 training images instead of a single image.
* Self-adversarial training (SAT) is a data augmentation technique that adds noise to training data in order to keep a model more robust.
* DropBlock regularization is a better regularization method for CNN.
* Cross mini-Batch Normalization (CmBN) is a variant of a CBN which assumes a batch contains four mini-batches.
* Weighted-Residual-Connections (WRC) is a type of residual block.
* Mish-activation is a new self regularized non-monotonic deep learning activation function. Is better that Relu. It improves accuracy and training stability.
* Cross Stage Partial Connections (CSP) is a new technique used in backbone that can enhance learning capability of CNN.
* CIoU loss achieves better accuracy and convergence speed on the Bounding Box regression problem.
* Spatial Pyramid Pooling layer (SPP) is used before fully connected layer. It uses multiple Maxpooling of different sizes. Next feature maps are concatenated.
* PAN path which aggregates information to improve accuracy. In YOLOv4 feature maps are concatenated, whereas in original PAN the feature maps were added.

They also tested various backbone options. The tests proved that CSPDarknet53 is less accurate on classification task than CSPResNeXt-50, however it is better on detection tasks. Therefore, the CSPDarknet53 was chosen.

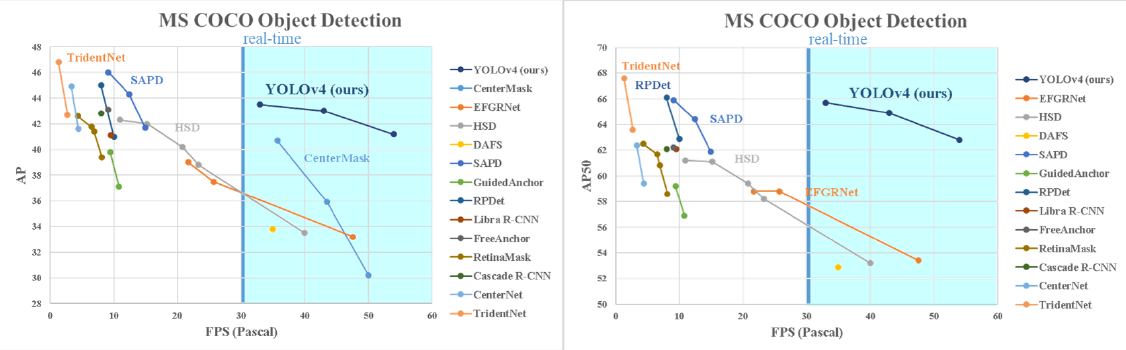


Figure 6 Comparison of the speed and accuracy of different object detectors running on PASCAL GPU [11]

YOLOv4 obtained an AP value of 43.5 percent and AP50 of 65.7 percent on the MS COCO dataset, and achieved a real-time speed above 30 FPS on the PASCAL GPU. Pascal GPU has CUDA, but do not have Tensor Cores. When run on Volta GPU, which is optimised for AI, it achieved 65FPS. It is the fastest and most accurate detectors in terms of both speed and accuracy. In addition, compared with YOLOv3, the AP and FPS have increased by 10 percent and 12 percent, respectively. [11]

Table 3 Comparison of different versions of YOLO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Train/test** | **GPU model** | **mAP** | **FPS** |
| YOLO [9] | PASCAL VOC |  | 63.4 | 45 |
| Fast YOLO [9] | PASCAL VOC |  | 52.7 | 155 |
| YOLOv2 (416x416) [10] | PASCAL VOC |  | 76.8 | 67 |
| YOLOv2 (416x416) [4] | COCO (UoI 0.5) |  | 44.0 | 40 |
| YOLOv3-320 [4] | COCO | Nvidia TITAN X Pascal | 51.5 | 45 |
| YOLOv3-416 [4] | COCO | Nvidia TITAN X Pascal | 55.3 | 34 |
| YOLOv3-608 [4] | COCO | Nvidia TITAN X Pascal | 57.9 | 17 |
| YOLOv4-416 [11] | COCO (UoI 0.5) | Nvidia TITAN X Pascal | 62.8 | 54 |
| YOLOv4-512 [11] | COCO (UoI 0.5) | Nvidia TITAN X Pascal | 64.9 | 43 |
| YOLOv4-608 [11] | COCO (UoI 0.5) | Nvidia TITAN X Pascal | 65.7 | 33 |

The table above includes mean Average Performance and speed of different versions of YOLO. YOLO has improved strongly. First and second versions were tested on PASCAL VOC. It is much easier test dataset. Version 2 was tested both on Pascal Voc and COCO dataset which is much more demanding. Hence the drop is mean Average Performance on COCO dataset. Versions 3 and 4 have different flavours depending on input size. The bigger dimension of image the better mean Avera precision but worst inference time. Bochkovskiy et al noted in theirs paper that during testing YOLOv4 run on Nvidia Titan X Pascal without using tensorRT, which is a GPU optimizer for running machine learning tasks.

# References

|  |  |
| --- | --- |
| [1] | G. Zeng, “blog.zenggyu.com,” 2018. [Online]. Available: https://blog.zenggyu.com/en/post/2018-12-16/an-introduction-to-evaluation-metrics-for-object-detection/. [Accessed 10 02 2021]. |
| [2] | H. -P. La, M. -T. Ha, H. -L. Nguyen and M. -T. Nguyen, “Vehicle Counting: Survey and Experiments,” *2020 7th NAFOSTED Conference on Information and Computer Science (NICS),* pp. pp. 350-355, 2020. |
| [3] | K. He, X. Zhang, S. Ren and J. Sun, “Deep Residual Learning for Image Recognition.,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. pp. 770-778, 2016. |
| [4] | J. Redmon, A. Farhadi, “YOLOv3: An Incremental Improvement,” 2018. |
| [5] | S. Ren, K. He, R. Girshick and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6,* pp. pp. 1137-1149, 1 June 2017. |
| [6] | L. Jiao et al, “A Survey of Deep Learning-Based Object Detection.,” *IEEE Access, vol. 7,* pp. pp. 128837-128868, 2019. |
| [7] | P. Adarsh, P. Rathi and M. Kumar, “YOLO v3-Tiny: Object Detection and Recognition using,” in *6th International Conference on Advanced Computing & Communication Systems* , 2020. |
| [8] | J. Redmon and A. Angelova, “Real-time grasp detection using convolutional neural networks,” *IEEE International Conference on Robotics and Automation (ICRA),* pp. pp. 1316-1322, 2015. |
| [9] | J. Redmon, S. Divvala, R. Girshick and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection.,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. pp. 779-788, 2016. |
| [10] | J. Redmon and A. Farhadi, “YOLO9000: Better, Faster, Stronger,” *IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. pp. 6517-6525, 2017. |
| [11] | A. Bochkovskiy, C. Wang, H. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection.,” 2020. |