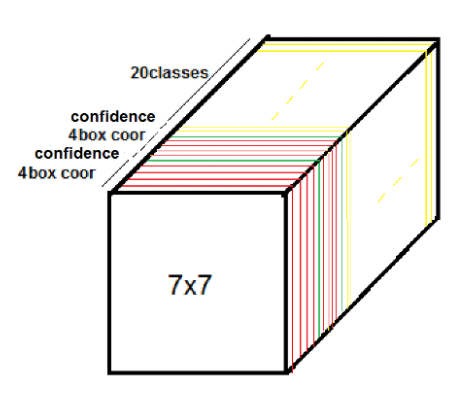
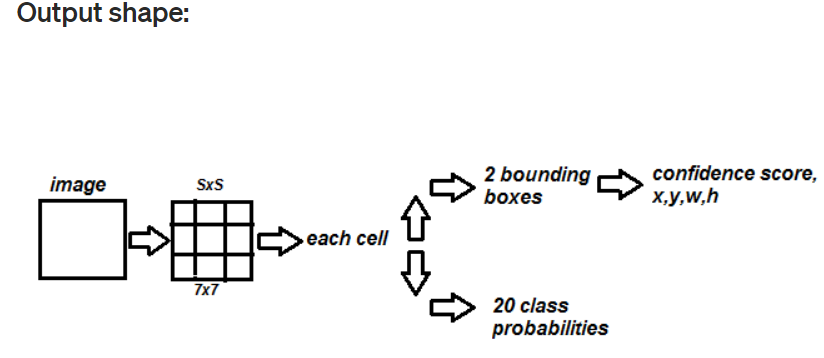
**Yolo History:**

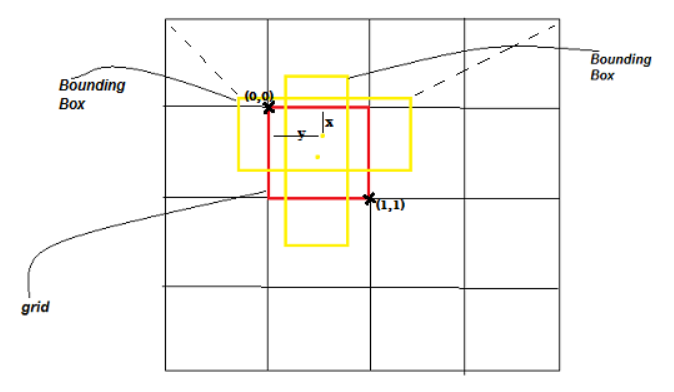
Yolov1:

* It is a custom network based on Googlenet architecture (not the commonly used VGG-16 with 30.6 billion floating point operations for a single pass over a single image of 224)
* Uses features from the entire image to predict each grid at the same time
* Unlike DPM (deformable parts model), it predicts all BB across all classes for an image
* Input image is divided into S\*S grids and each grid predicts B bounding boxes
* If the center of the obj. falls into a grid, that grid is responsible for detecting that obj. ( 1 obj. per grid while each grid has two BB prediction)
* Confidence of each BB = Pr(Object) \* IOU
* Each BB has 5 predictions: x,y,w,h,confidence
* Output tensor = S\*S\*(B\*5 + C)

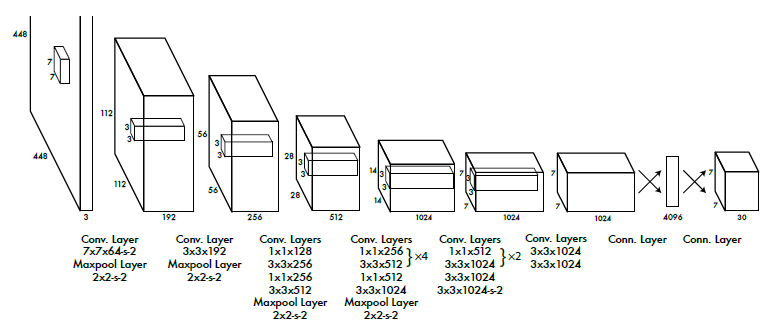
For B=2, C=20, S=7

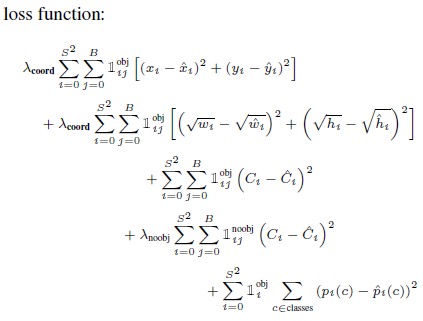


* The model outputs a confidence score for objectness of each BB in each grid (the goal is to prevent the model to detect the background)
* the class predictions predict the conditional probability of that class given that there is an object
* Thus if no object is in the grid, the objectness is zero. Otherwise, it is equal to IOU. Why? Because we IOU is calculated from the ground truth box, therefore, we are sure that there is an object inside manually drawn box. The higher the IOU, the higher the possibility that the predicted BB contains an object → higher objectness score
* BB width and height are normalized by the image w,h → BBw,h fall between 0,1
* BB dimension start with a prior and can be learnt by the network
* The same with x,y like last point ↑

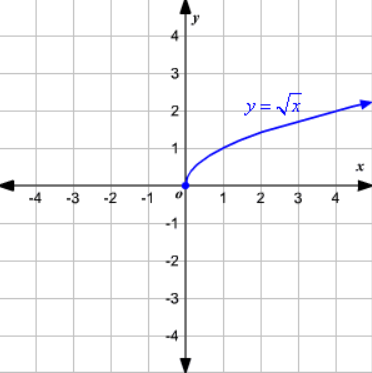


* It predicts the coordinates of BB directly using FC layers on top of the convolutional feature extractor.
* It has significant number of localization error and relatively low recall compared to region proposal-based methods.
* Since it learns to predict bounding boxes from data, it struggles to generalize to objects in new or unusual aspect ratios or configurations
* It also uses relatively coarse features for predicting bounding boxes since our architecture has multiple downsampling layers from the input image.
* Downsampling of factor 32 → 7\*7\*2BB = 98 BB in total
* Yolo is fast since everything is done in a single network
* Trained for 20 classes





* 1(obj)ij =1 if an object appears in cell i + box j for this cell is responsible for that object ,otherwise 0.
* “1(obj)ij=1 only if the box contain an object and responsible for detect this object (higher IOU)“
* Why square root of w and h? Our error metric should reﬂect that small deviations in large boxes matter less than in small boxes. To partially address this YOLO predicts the square root of the bounding box width and height instead of the width and height directly.



* λcoord=5 to give more importance to localization (more punishment w.r.t. classification)
* 3rd Line: Tries to make the confidence score equal to IOU (C=1), when there is an obj.
* 4th line: Tries to make the confidence equal to one, where there is no obj.
* 1(noobj)ij = 1 if (there is no object inside cell i) or (there is an object ,but the box j for this cell is not responsible for that object) ,otherwise 0.
* λnoobj =0.5 to prevent the network jump to the conclusion (diverge) based on the fact that mostly the grids do not contain any object
* last line of the loss function sums the errors for all the classes probabilities for the 49 grid cells.

Training:

* First they pretrained the convolutional layers of the network for classification on the ImageNet 1000-class competition dataset. For pretraining they used the ﬁrst 20 convolutional layers from the network we talked about previously followed by a average-pooling layer and a 1x1000 fully connected layer with input size of 224×224 .This network achieve a top-5 accuracy of 88%.
* Then they removed the 1x1000 fully connected layer and added four convolutional layers and two fully connected layers with randomly initialized weights and increased the input resolution of the network from 224×224 to 448×448. After that they trained the model for detection.
* Change from 224\*224 to 448 for detection → (difficult for the network to switch to learning object detection and adjust to the new input resolution)

NMS:

* For each class (cars, pedestrians, cats,….) do:

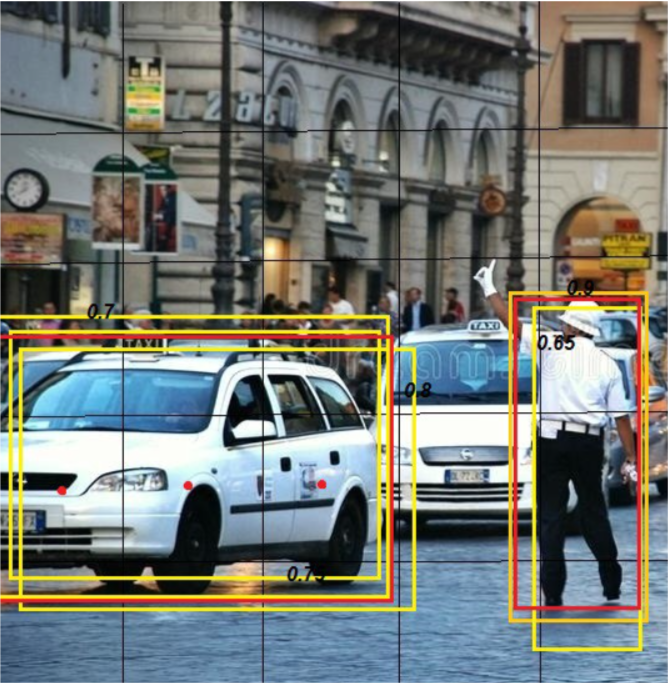
1-Discard all boxes with confidence C<C -threshold (for example C<0.5)

2- Sort the predictions starting form the highest confidence C.

3-Choose the box with the highest C and output it as a prediction .

4-Discard any box with IOU>IOU-threshold with the box in the previous step .

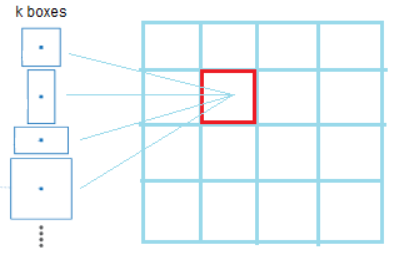
5-Start again from step (3) until all remaining predictions are checked.



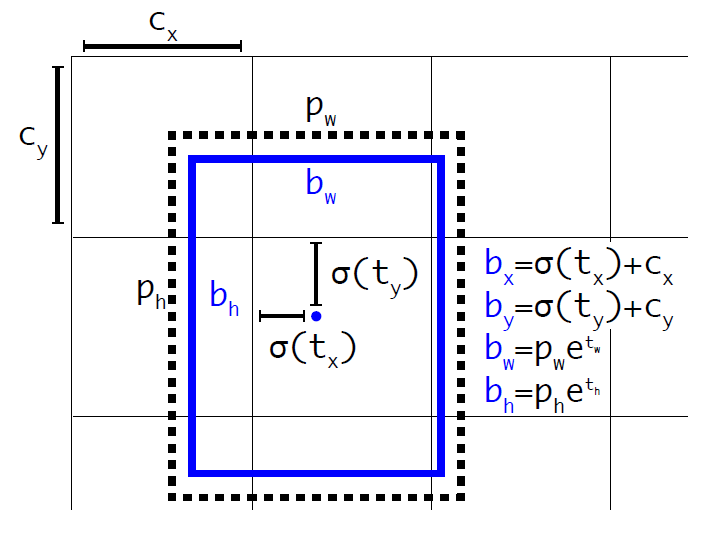
Yolov2:

* Yolov1 makes a signiﬁcant number of localization errors. Furthermore, YOLO has relatively low recall. Thus in the second version of YOLO they focused mainly on improving recall and localization while maintaining classiﬁcation accuracy. To achieve better performance they used some ideas:

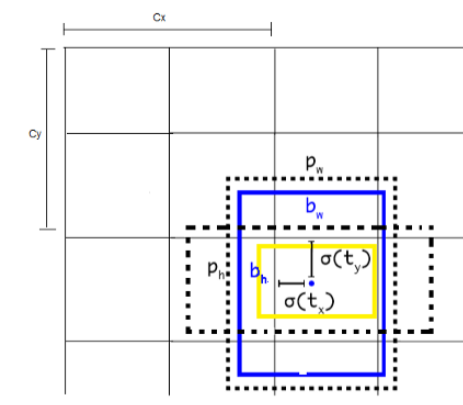
1. Batch Normalization: 2% improvement in mAP.
2. High Resolution Classiﬁer: For Yolov2 they initially trained the model on images at 224×224, then they ﬁne tune the classiﬁcation network at the full 448×448 resolution for 10 epochs on ImageNet before training for detection.
3. Convolution with Anchor Boxes: In Yolov1 any grid cell can only detect one object → problem! To solve this, the authors tried to allow the grid cell detect more than one object using k bounding box.



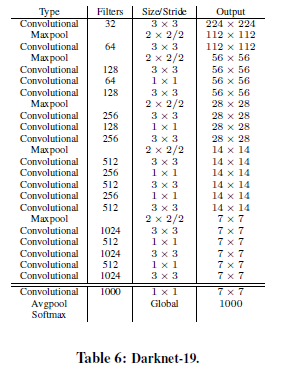
* Anchor box: An anchor box is a width and height, which we can predict the bounding box relative to it instead of predicting the box relative to the all image .Using this idea it will be easier for the network to learn. Using only convolutional layers (without fully connected layers). Faster R-CNN predicts offsets and conﬁdences for anchor boxes.
* Yolov1 predicts the coordinates of bounding boxes directly using fully connected layers on top of the convolutional feature extractor.



* “t” stands for target value → output of the network
* “p” stands for prior value → handmade values which in case of Yolo are calculated by k-means over ImageNet boxes
* Sigmaoid function serves the fact that the values remain between zero and one. We also have exponential parameterization to ensure that bw and bh are non-negative. However, bw and bh could be greater than one.
* The blue box is the final detection of the network
* For example if we use two anchor boxes the grid cell (2, 2) in the image below will output 2 boxes (the blue and the yellow boxes). Let the black dotted boxes represent the two anchor boxes for that cell.



* Now consider only the blue box, instead of assigning the predicted blue box to the grid cell only as in YOLO, YOLOv2 assigns the blue box not only to the grid cell but also to one of the anchor boxes and that will be the one that has the highest IOU with the ground truth box. YOLOv2 uses the above equations to assign the blue box to the grid and the anchor box.
* With anchor boxes it predicts more than thousand boxes (obj. is paired with [grid, anchor box] not only a grid like in Yolov1)
* Using anchor boxes increased recall without change of mAP
* Darknet-19: Less complex with 19 convolutional layers and 5 maxpooling layers. It achieved 91.2% top-5 accuracy on ImageNet, which is better than VGG (90%) and YOLO network (88%)
* Output shape: 13x13x(b(1+4+20)) → if b=5 and number of classes = 20 → 31x13x125
* Using new network reduced computation by 33%. It is called Darknet-19



Training:

* The model was first trained for classification then it was trained for detection.

1. Classification: they trained Darknet-19 network on the standard ImageNet 1000 class classiﬁcation dataset with input shape 224x224 for 160 epoch. After that they fine tune the network at large input size 448x448 for 10 epoch. This give them a top-1 accuracy of 76.5% and a top-5 accuracy of 93.3%
2. Detection: After training for classification they removed the last convolutional layer from Darknet-19 and instead they added three 3 × 3 convolutional layers and a 1x1 convolutional layer with the number of outputs we need for detection (13x13x125). Also a passthrough layer was added so that our model can use ﬁne grain features from previous layers. Then they trained the network for 160 epochs on detection datasets (VOC and COCO datasets).

* Input image is shrinked from 448 to 416 to have a center grid for detection.
* Multi-Scale Training: Instead of ﬁxing the input image size, they changed the network every few iterations. After every 10 batches the network randomly chooses a new image dimension size from the dimensions set {320,352,384,…,608}(multiply of 32). Then they resize the network to that dimension and continue training. This means the same network can predict objects at different resolutions(input shapes). Maybe we can do the same for both sensors. We develop a network whose input image has sometimes 3 channel (RGB) and sometime only 1 (IR), thus the network is trained to work with both inputs. In another word, the filters are trained to extract features from IR and the Optic camera on their different layers. Since the features are very similar, it wouldn’t be a very difficult task for the kernels to adapt themselves and provide good features for both data types. Suggestion: V1 network → we preprocess the FLIR data and resize it in the above-mentioned range. Since we will do data augmentation, we can produce a lot more training data out of what FLIR has provided us. Then we not only change the input size every 10 batches, but we change the number of channels as well and feed IR and RGB images interchangeably on all the data available in the internet. Obviously, the network will be much better on RGB as we have much more data for that but it will be a good starting point for both sensors.

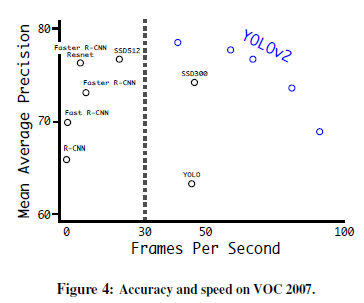
V2 network → We can use the weights and structure of the V1 as a backbone for two separate networks if we wish for redundancy of IR and RGB network. This way, we can buy/find more data (ex. A huge dataset from FLIR cost 4000€) and fine-tune the V1 network for IR to have a separate network.

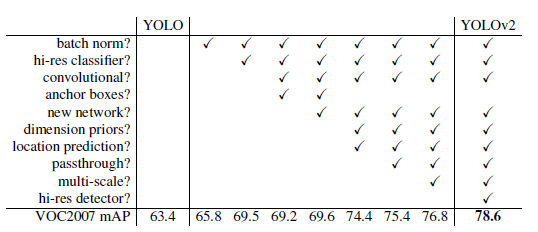
Note: The IR trained kernel will be of a help for RGB network as well.

Scientific approach:

1. Take the best yolo and test it on IR data data → write the result
2. Modify the structure of yolo to extract features from IR inputs and concatenate them to the main pipeline → write the result
3. Do it as I mentioned above (without modifying the network to avoid introducing more layers, complexity and parameters), but apply a new training technique → write the result
4. Compare

* The data augmentation is same as Yolov1 and SSD with random crops, color shifting, etc
* Largest model runs on 40 FPS on a Gefore GTX Titan X (12 GB GDDR5) which is still above real time



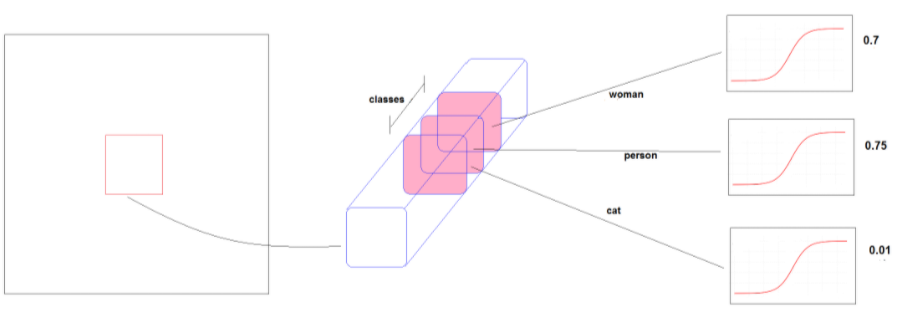


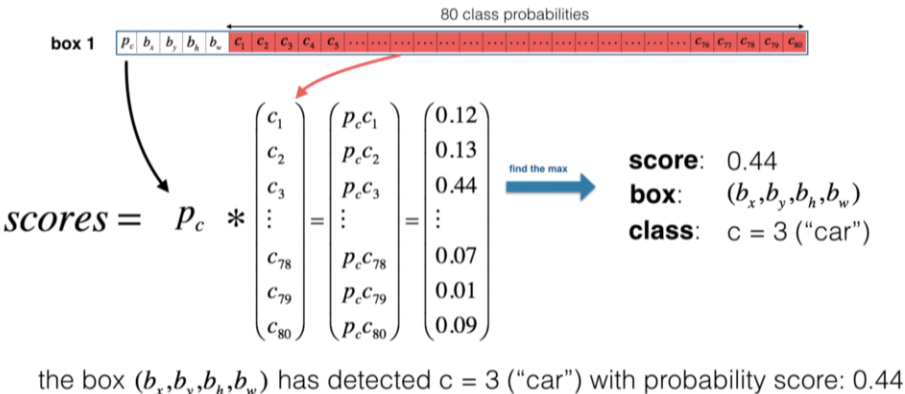
Yolo9000:

* The authors propose a mechanism for jointly training on classiﬁcation and detection data. During training, they mix images from both detection and classiﬁcation datasets. When the network sees an image labeled for detection, we can backpropagate based on the full YOLOv2 loss function. When it sees a classiﬁcation image we only backpropagate loss from the classiﬁcation speciﬁc parts of the architecture → It has its own complexities to produce a coherent labeled dataset from two different dataset like COCO (rough class detection) and ImageNet (fine class classification). For more detail read the paper

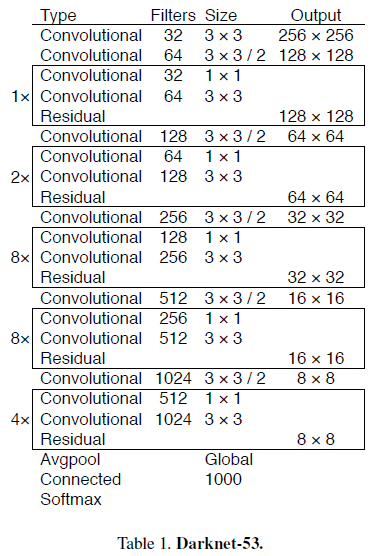
Yolov3:

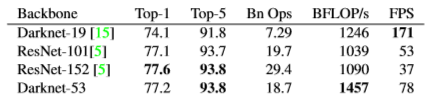
* BB prediction is modified. YOLOv3 also predicts an objectness score (confidence) for each bounding box using logistic regression. This should be one if the bounding box prior overlaps a ground truth object by more than any other bounding box prior.
* The system only assigns one bounding box prior for each ground truth object. If a bounding box prior is not assigned to a ground truth object it incurs no loss for coordinate or class predictions, only objectness.
* If the IOU is less than 0.5 (threshold), we ignore the prediction
* Multi labels prediction: each box predicts the classes which, the bounding box may contain using multi-label classification. An image can be a person and a woman at the same time. Binary cross-entropy loss is used instead of softmax for Yolov3 (independent logistic classifier)
* In other words, Yolov3 doesn’t use Softmax to assign the probability for the class of the object residing in a bounding box. Instead, it applies logistic regression to calculate the score for each class and removes the predictions that are below a specific threshold. In this way, Yolov3 provides us with a multilabel classification for the objects detected as softmaxing relies on the assumption that we have exclusive classes. Therefore, YOLOv3 uses binary cross-entropy loss for each class along with the localization loss(errors between the predicted box & the ground truth box) and the confidence loss(objectness of the box).



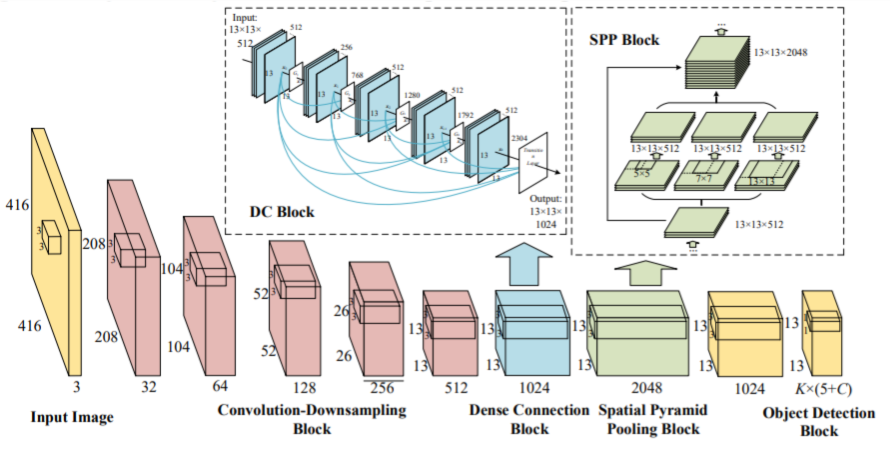


* YOLO struggles with small objects. However, with YOLOv3 we see better performance for small objects, and that because of using short cut connections. Using these connections method allows us to get more ﬁner-grained information from the earlier feature map. However comparing to the previous version, YOLOv3 has worse performance on medium and larger size objects.
* The first 53 layers of the network were trained on Imagenet for the classification task. 53 more layers are stacked on the existing layers to eventually obtain 106 layers fully ConvNet architecture for Yolov3. In Yolov3, we don’t have any pooling layers, it makes downsampling by using stride=2.
* Darknet-53 as backbone:

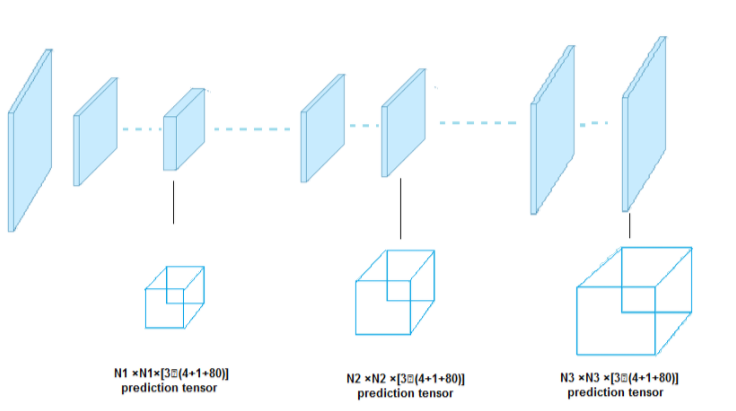




* To make sure that the feature extraction network is better than ResNet’s group. They have compare the results on ImageNet classification dataset. (Comparison of backbones)
* This network is a hybrid approach between the network used in YOLOv2 (Darknet-19), and the residual network, so it has some short cut connections.
* Darknet-53 also achieves the highest measured floating point operations per second. This means the network structure better utilizes the GPU, making it more efficient to evaluate and thus faster. That is mostly because ResNets have just way too many layers and aren’t very efficient.
* Remember that Yolov3 uses downsampling (stride=2) in Convolutional layers but Yolov3-spp uses downsampling (stride=2) in Convolutional layers + gets the best features in Max-Pooling layers



* [SPP (Spacial pyramid pooling)](https://becominghuman.ai/explaining-yolov4-a-one-stage-detector-cdac0826cbd7), which is originated from spatial pyramid matching (SPM) is modified to be used for CNN in order to increase the receptive field of the backbone.
* After training on classification the fully connected layer is removed from Darknet-53.
* Prediction across scales: Unlike YOLO and YOLO2, which predict the output at the last layer, YOLOv3 predicts boxes at 3 different scales as illustrated in the below image.
* One of the most significant improvements made by Yolov3 is that it makes a prediction at 3 different scales. The first prediction is made by the 82nd layer giving a feature map of 19x19x((4 + 1 + C)\*B). Second prediction is made by 94th layer giving a feature map of 38x38x((4 + 1 + C)\*B). The third prediction is made by the 106th layer giving a feature map of 76x76x((4 + 1 + C)\*B). These scaling operations are realized by upsampling=2 and merging specific layers. The large objects are detected in the first Yolo layer while the small ones are detected in the third Yolo layer.
* At each scale YOLOv3 uses 3 anchor boxes and predicts 3 boxes for any grid cell. Each object still only assigned to one grid cell in one detection tensor.



Summary:

The Original YOLO - YOLO was the first object detection network to combine the problem of drawing bounding boxes and identifying class labels in one end-to-end differentiable network.

YOLOv2 - YOLOv2 made a number of iterative improvements on top of YOLO including BatchNorm, higher resolution, and anchor boxes.

YOLOv3 - YOLOv3 built upon previous models by adding an objectness score to bounding box prediction, added connections to the backbone network layers, and made predictions at three separate levels of granularity to improve performance on smaller objects.

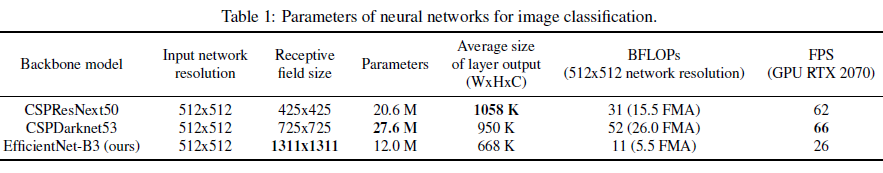


Sources:

1. <https://medium.com/@amrokamal_47691/yolo-yolov2-and-yolov3-all-you-want-to-know-7e3e92dc4899>
2. <https://medium.com/swlh/object-detection-on-thermal-images-4f3410a89db4>
3. Official papers

Yolov4:

* Just like batch normalization and residual connections; WRC (weighted residual con.), CSP (cross-staged-partial-con.), CmBN( cross mini-batch normalization), SAT (self-adversarial-training) and Mish-activation are applicable to majority of models, tasks and datasets.
* [Mosaic data augmentation](https://blog.roboflow.com/a-thorough-breakdown-of-yolov4/), CIoU loss and their combination is also used in this version
* It aims for a network with capability of an object detection in production (not research only) and optimization for parallel computations, rather than low computation volume theoretical indicator (BFLOP)
* It verify the influence of [Bag-of-Freebies and Bag-of-Specials](https://blog.roboflow.com/a-thorough-breakdown-of-yolov4/) methods for the object detection network during training
* The state of the art methods are modified to be more efficient and suitable for a single GPU training. CBN (cross-iteration batch normalization), PAN (path aggregation Network), SAM (spatial attention module)
* The network which is designed for GPU use small number of groups (1-8) in convolution layers CSPDarknet53 (CSPNet + Darknet53)
* Note: CSPDarknet53 is better in object detection on COCO and CSPResNext50 is better in classification on ImageNet
* In contrast to the classifier, the detector requires the following:
  + Higher input network size (resolution) – for detecting multiple small-sized objects
  + More layers – for a higher receptive field to cover the increased size of input network
  + More parameters – for greater capacity of a model to detect multiple objects of different sizes in a single image
* They take the network with the larger receptive field size (larger number of conv. Layers 3\*3) and larger number of parameters as the backbone → CSPDarknet53 it is!



* [SPP block (Spatial Pyramid Pooling)](https://becominghuman.ai/explaining-yolov4-a-one-stage-detector-cdac0826cbd7) is added to CSPDarknet53 since it significantly increase [the receptive field](https://medium.com/mlreview/a-guide-to-receptive-field-arithmetic-for-convolutional-neural-networks-e0f514068807). It separates out the most significant image features with almost no computational cost
* Instead of [FPN (Feature Pyramid Network)](https://medium.com/clique-org/panet-path-aggregation-network-in-yolov4-b1a6dd09d158) used in Yolov3, [PANet(Path Aggregation Network)](https://medium.com/clique-org/panet-path-aggregation-network-in-yolov4-b1a6dd09d158) is used as the method of parameter aggregation
* Cross-GPU Batch Normalization (CGBN or SyncBN) or expensive specialized devices are not used. This allows anyone to reproduce the state-of-the-art outcomes on a conventional graphic processor.
* Drop-Block has shown outstanding performance in compared to other regularization methods and therefore it is taken.
* Additional improvements toward training on single GPU:
  + They introduce a new method of data augmentation Mosaic, and Self-Adversarial Training (SAT)
  + They select optimal hyper-parameters while applying genetic algorithms
  + They modify some exsiting methods to make our design suitble for efficient training and detection – modified [SAM(Spatial Attention Module)](https://becominghuman.ai/explaining-yolov4-a-one-stage-detector-cdac0826cbd7), modified PAN, and Cross mini-Batch Normalization (CmBN)
* [Mosaic](file:///\\isnasv117.in.audi.vwg\EFS\EFS-GX6\4130_Arbeitsgruppen\4137_KHO_Performance\Studentische_Themen\Sam\Papers\Detection&Tricks\YOLO\Yolov4.pdf) help the detection outside context and also reduce the mini-batch size, since every batch normalization calculates activation statistics from 4 different images on each layer
* [Self-Adversarial Training (SAT)](file:///\\isnasv117.in.audi.vwg\EFS\EFS-GX6\4130_Arbeitsgruppen\4137_KHO_Performance\Studentische_Themen\Sam\Papers\Detection&Tricks\YOLO\Yolov4.pdf) which operates in 2 forward backward stages. First stage change the image instead of the weights and in the second stage the network has to detect from the image which contains deception
* [Genetic Algorithm](https://medium.com/cindicator/genetic-algorithms-and-hyperparameters-weekend-of-a-data-scientist-8f069669015e) is used for hyperparameter tunning. For more detail on training parameters, read the paper’s page7!
* Label smoothing is used and a bunch of stuff are done for the classifier → page 8
* A whole lots of stuff are done also to improve the detector (ex. Grid sensitivity problem)→ page 8

Yolov4-Scaled:

* In RegNet they have found that optimal depth of CNN is 60, bottleneck ratio is 1 and width increase rate of cross-stage is 2.5
* Through analysis of state of the art object detector, we found that CSPDarknet53 satisfy all the optimal architecture features
* Steps are as follows: Yolov4 → Yolov4-CSP → Scaled-Yolov4
* Traditional scaling method is to add more depth to the network
* Here Synergic compound scaling is considered
* The computation cost w.r.t. depth, width and scaling size is calculated
* They could reduce the computational cost of famous models like ResNet, ResNeXt and Darknet by 23.5%, 46.7% and 50% by applying CSPNet method. Therefore, they decided to use CSP-ized models for performing model scaling
* In one-stage object detector, the feature vector corresponding to each location is used to predict the category and size of an object at that location. The ability to better predict the size of an object basically depends on the receptive field of the feature vector. In the CNN architecture, the thing that is most directly related to receptive field is the stage, and the feature pyramid network (FPN) architecture tells us that higher stages are more suitable for predicting large objects.
* Yolov4-P6 can reach real time 30 FPS when the width scaling factor is equal to 1