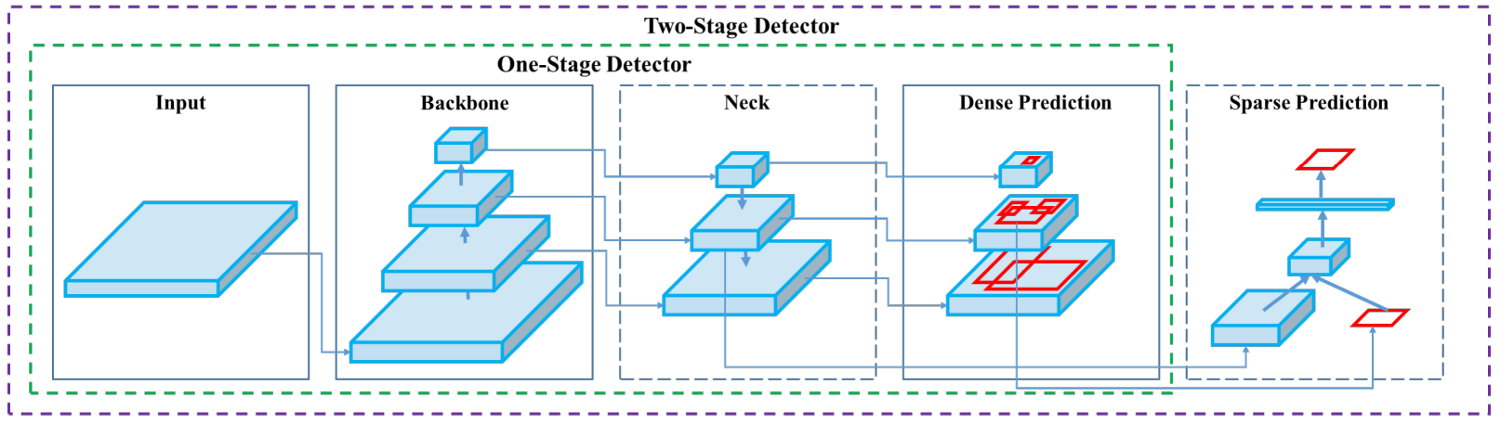
**STUDY ON YOLOv4 Architecture:**

**YOLOv4:**

1. YOLOv4 outruns the existing methods significantly in both terms “detection performance” and “superior speed”.
2. Due to “speedily operating” feature, object detector can be trained smoothly and used in production systems.
3. The main objective was “to optimize neural networks detector for parallel computations”, they also introduce various different architectures and architectural selections after attentively analyzing the effects on the performance of numerous detector, features suggested in the previous YOLO models.
4. The main goal of this work is designing a fast operating speed of an object detector in production systems and optimization for parallel computations, rather than the low computation volume theoretical indicator (BFLOP).
5. It proposes an efficient and powerful object detection model. It means everyone can use a 1080Ti or 2080Ti GPU to train a superfast and accurate object detector.
6. It verifies the influence of state-of-the-art Bag-of-Freebies and Bag-of-Specials methods of object detection during the detector training.
7. It modifies the state-of-the-art methods and make them more efficient and suitable for single GPU training, including CBN [89], PAN [49], SAM [85], etc.

Ref.: <https://arxiv.org/pdf/2004.10934.pdf>

**YOLOv4 Architecture:**



1. A modern detector is usually 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.
2. For those detectors running on GPU platform, their backbone could be VGG, ResNet, ResNeXt, or DenseNet.
3. For those detectors running on CPU platform, their backbone could be SqueezeNet, MobileNet, or ShuffleNet.
4. As to the head part, it is usually categorized into two kinds, i.e., one-stage object detector and two-stage object detector.
5. The most representative two-stage object detector is the R-CNN series, including fast R-CNN, faster R-CNN, R-FCN, and Libra R-CNN. It is also possible to make a two-stage object detector an anchor-free object detector, such as Rep Points.
6. As for one-stage object detector, the most representative models are YOLO, SSD, and RetinaNet.
7. Object detectors developed in recent years often insert some layers between backbone and head, and these layers are usually used to collect feature maps from different stages. We can call it the neck of an object detector. Usually, a neck is composed of several bottom-up paths and several top-down paths. Networks equipped with this mechanism include Feature Pyramid Network (FPN), Path Aggregation Network (PAN), BiFPN, and NAS-FPN.
8. some researchers put their emphasis on directly building a new backbone (DetNet [43], DetNAS) or a new whole model (SpineNet, HitDetector) for object detection.
9. To sum up, an ordinary object detector is composed of several parts:
10. **Input:** Image, Patches, Image Pyramid
11. **Backbones:** VGG16, ResNet-50, SpineNet, EfficientNet-B0/B7, CSPResNeXt50, CSPDarknet53.
12. **Neck:**

* **Additional blocks:** SPP, ASPP, RFB, SAM
* **Path-aggregation blocks:** FPN, PAN, NAS-FPN, Fully-connected FPN, BiFPN, ASFF, SFAM.

1. **Heads:**

* **Dense Prediction (one-stage):** 
  + RPN, SSD, YOLO, RetinaNet (anchor based)
  + CornerNet, CenterNet , MatrixNet , FCOS (anchor free).
* **Sparse Prediction (two-stage):**
  + Faster R-CNN , R-FCN , Mask RCNN (anchor based)
  + Rep Points (anchor free)

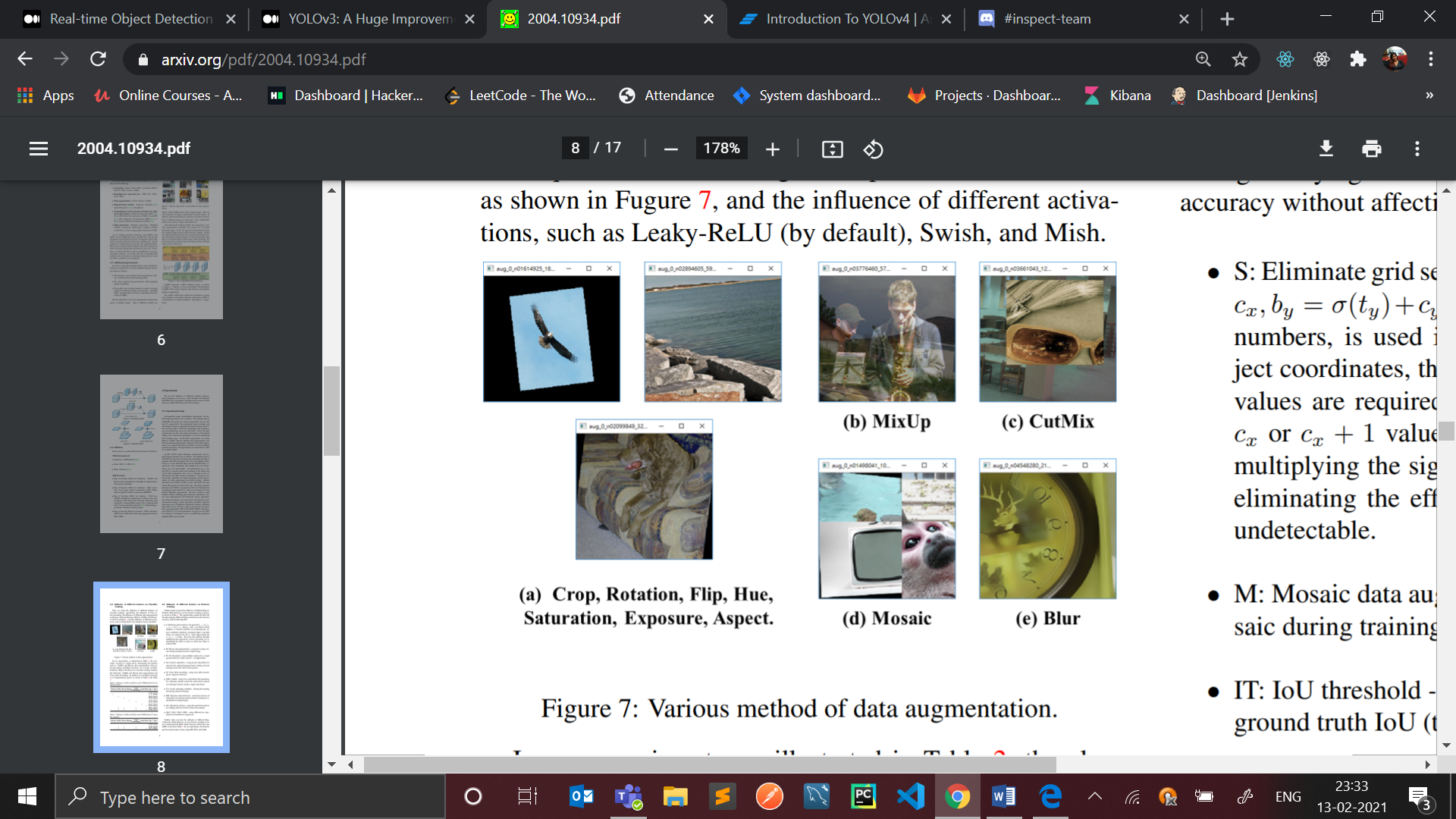
**Bag of Freebies:**

1. Usually, a conventional object detector is trained offline. Therefore, researchers always like to take this advantage and develop better training methods which can make the object detector receive a better accuracy without increasing the inference cost. We call these methods that only change the training strategy or only increase the training cost as “bag of freebies.”
2. What is often adopted by object detection methods and meets the definition of bag of freebies is data augmentation.
3. The purpose of data augmentation is to increase the variability of the input images, so that

the designed object detection model has higher robustness to the images obtained from different environments. For examples, photometric distortions and geometric distortions

are two commonly used data augmentation method and they definitely benefit the object detection task. In dealing with photometric distortion, we adjust the brightness, contrast, hue, saturation, and noise of an image. For geometric distortion, we add random scaling, cropping, flipping, and rotating.

1. The data augmentation methods mentioned above are all pixel-wise adjustments, and all original pixel information in the adjusted area is retained.
2. Random erase and CutOut can randomly select the rectangle region in an image and fill in a random or complementary value of zero.
3. As for hide-and-seek and grid mask, they randomly or evenly select multiple rectangle regions in an image and replace them to all zeros. If similar concepts are applied to feature maps, there are DropOut, DropConnect, and DropBlock methods.
4. In addition, some researchers have proposed the methods of using multiple images together to perform data augmentation. For example, MixUp uses two images to multiply and superimpose with different coefficient ratios, and then adjusts the label with these superimposed ratios.
5. As for CutMix, it is to cover the cropped image to rectangle region of other images, and adjusts the label according to the size of the mix area. In addition to the above mentioned methods, style transfer GAN is also used for data augmentation, and such usage can effectively reduce the texture bias learned by CNN.
6. The last bag of freebies is the objective function of Bounding Box (BBox) regression. The traditional object detector usually uses Mean Square Error (MSE) to directly perform regression on the center point coordinates and height and width of the BBox.
7. CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, Eliminate grid sensitivity, Using multiple anchors for single ground truth, Cosine annealing scheduler, Optimal hyper-parameters, Random training shapes.



**Bag of Specials:**

1. For those plugin modules and post-processing methods that only increase the inference cost by a small amount but can significantly improve the accuracy of object detection, we call them “bag of specials”.
2. Generally speaking, these plugin modules are for enhancing certain attributes in a model, such as enlarging receptive field, introducing attention mechanism, or strengthening feature integration capability, etc., and post-processing is a method for screening model prediction results.
3. Common modules that can be used to enhance receptive field are SPP, ASPP, and RFB .
4. The attention module that is often used in object detection is mainly divided into channel-wise attention and pointwise attention, and the representatives of these two attention models are Squeeze-and-Excitation (SE) and Spatial Attention Module (SAM), respectively. Although SE module can improve the power of ResNet50 in the ImageNet image classification task 1% top-1 accuracy at the cost of only increasing the computational effort by 2%, but on a GPU usually it will increase the inference time by about 10%, so it is more appropriate to be used in mobile devices. But for SAM, it only needs to pay 0.1% extra calculation and it can improve ResNet50-SE 0.5% top-1 accuracy on the ImageNet image classification task. Best of all, it does not affect the speed of inference on the GPU at all.
5. Since multi-scale prediction methods such as FPN have become popular, many lightweight modules that integrate different feature pyramid have been proposed. The modules of this sort include SFAM, ASFF, and BiFPN.
6. The main idea of SFAM is to use SE module to execute channel wise level re-weighting on multi-scale concatenated feature maps.
7. As for ASFF, it uses softmax as point-wise level reweighting and then adds feature maps of different scales.
8. In BiFPN, the multi-input weighted residual connections is proposed to execute scale-wise level re-weighting, and then add feature maps of different scales.
9. The post-processing method commonly used in deep learning-based object detection is NMS, which can be used to filter those BBoxes that badly predict the same object, and only retain the candidate BBoxes with higher response. The way NMS tries to improve is consistent with the method of optimizing an objective function.

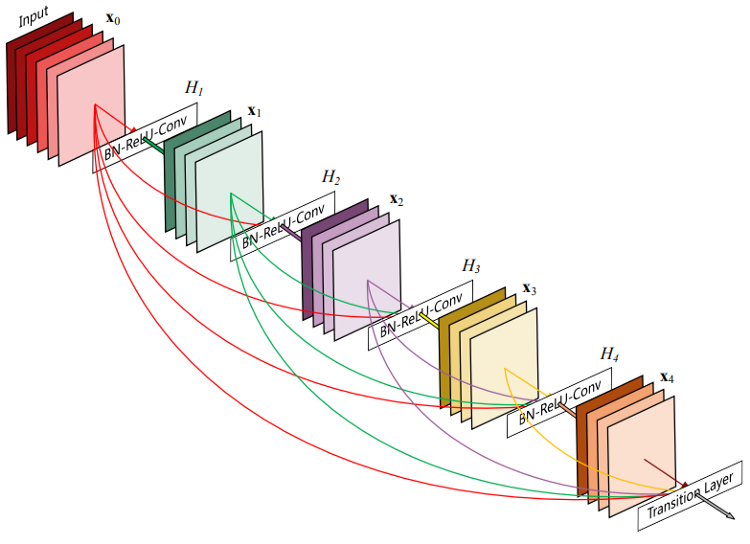
Selection of BoF and BoS:

For improving the object detection training, a CNN usually uses the following:

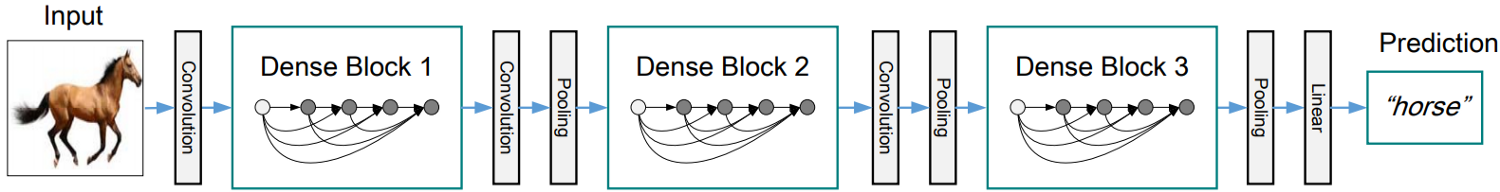
1. Activations: ReLU, leaky-ReLU, parametric-ReLU, ReLU6, SELU, Swish, or Mish.
2. Bounding box regression loss: MSE, IoU, GIoU, CIoU, DIoU.
3. Data augmentation: CutOut, MixUp, CutMix.
4. Regularization method: DropOut, DropPath, Spatial DropOut, or DropBlock.
5. Normalization of the network activations by their mean and variance: Batch Normalization (BN), Cross-GPU Batch Normalization (CGBN or SyncBN), Filter Response Normalization (FRN), or Cross-Iteration Batch Normalization (CBN).
6. Skip-connections: Residual connections, Weighted residual connections, Multi-input weighted residual connections, or Cross stage partial connections (CSP).

**Backbone:**

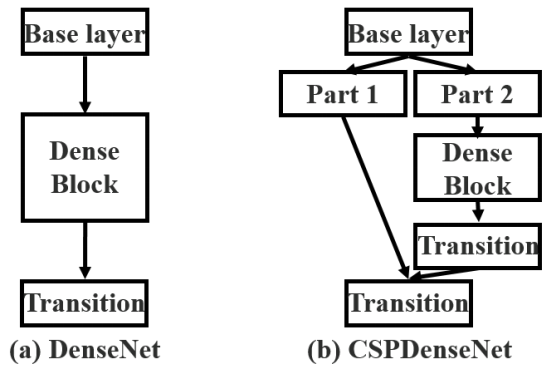
1. Backbone here refers to the feature-extraction architecture.
2. YOLOv3 has Darknet53 with 53 convolutional layers, so it’s more accurate but slower whereas uses the backbone which isn’t Darknet53 but CSPDarknet53.
3. Backbone is one of the ways where we can improve accuracy, we can design a deeper network to extend the receptive field and to increase model complexity.
4. And to ease the training difficulty, skip-connections can be applied.
5. We can expand this concept further with highly interconnected layers.
6. A Dense Block in the YOLOv4 backbone contains multiple convolution layers with each layer *Hi* composed of batch normalization, ReLU, and followed by convolution. Instead of using the output of the last layer only, *Hi* takes the output of all previous layers as well as the original as its input. i.e. *x₀, x₁, …, and xᵢ₋₁*. Each *Hi* below outputs four feature maps. Therefore, at every layer, the amount of feature maps is magnified by four — the rate of growth.
7. Then a DenseNet can be formed by composing multiple Dense Blocks with a transition layer in between that composed of convolution and pooling.
8. Dense block:

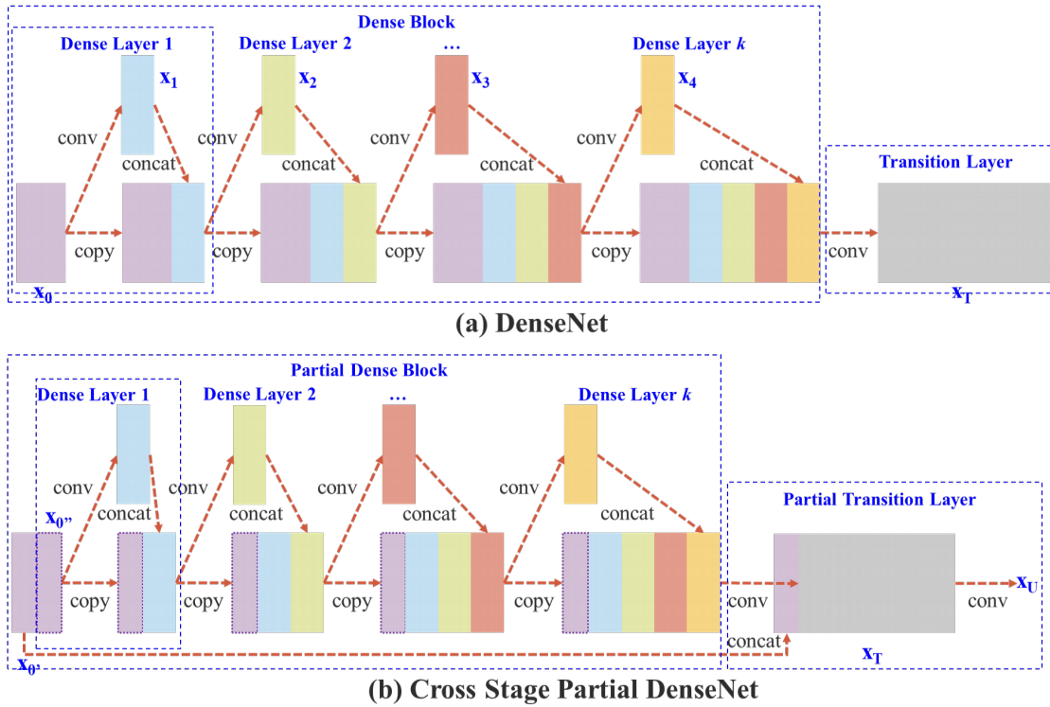


1. Dense Net:

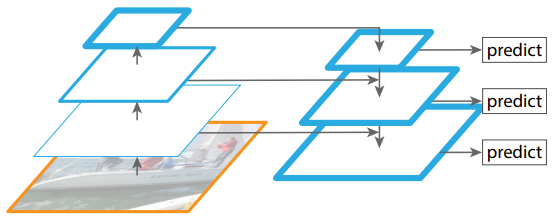
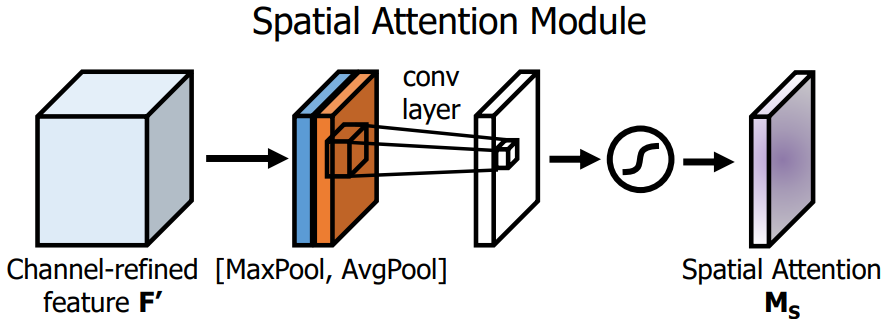
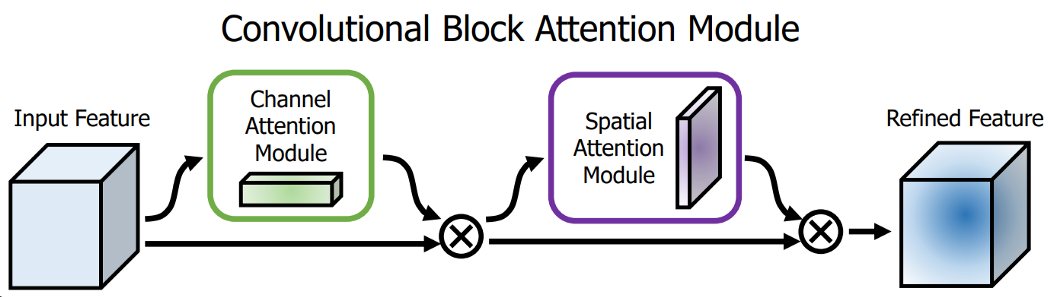
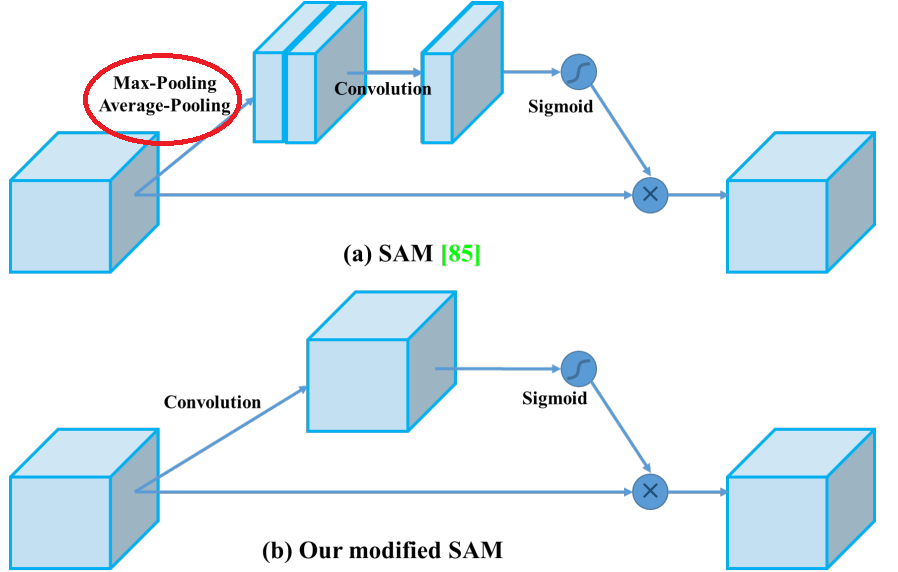
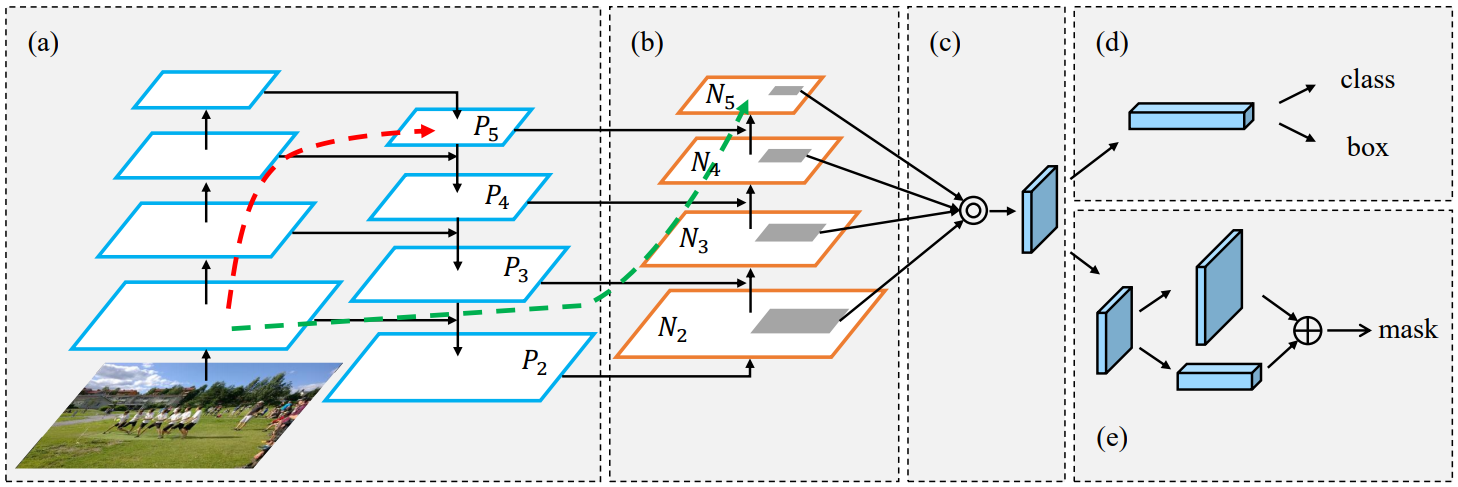


1. CSP stands for Cross-Stage-Partial connections. The idea here is to separate the input feature maps of the Dense Block into two parts, one that will go through a block of convolutions, and one that won’t. Then, we aggregate the results.
2. For example:



1. This new proposed design reduces the complexity of the computation by separating the input into two parts — only one goes through the Dense Block.
2. CSPNet separates the feature map of the base layer into two-part, one part will go through a dense block and a transition layer; the other part is then combined with a transmitted feature map to the next stage.
3. 

**Neck:**

1. The main purpose of the neck block is to add additional layers between the backbone and the so-called head (dense prediction block).
2. In the early days of convolutional neural networks, everything was very linear. Networks were evolving in more recent versions, there appeared middle blocks, skip connections, and aggregations of data between layers.
3. Thus, to enrich the information that enters the head, neighbouring feature maps coming from the bottom-up and the top-down stream are combined together element-wise or concatenated before entering the head.
4. Therefore, the network head’s input will contain spatial rich information from the bottom-up stream and the semantic rich information from the top-down stream. This part of the system is named a neck.
5. **Feature Pyramid Networks (FPN)**:
   1. While making predictions for a particular scale, FPN upsamples (2×) the previous top-down stream and add it with the neighboring layer of the bottom-up stream (see the diagram below). The result is passed into a 3×3 convolution filter to reduce upsampling artifacts and create the feature maps P4 below for the head..
   2. In YOLOv4, the FPN concept is gradually implemented/replaced with the modified SAM, PAN, and SPP.
   3. 
6. **Spatial Attention Module (SAM)**:
   1. Attention mechanisms have widely adopted in deep learning, and especially in recurrent neural network designs.
   2. With SAM, the maximum pool and the average pool are applied separately by entering feature maps to create two sets of feature maps.
   3. The results are passed into a convolution layer followed by a sigmoid function to create spatial attention.
   4. 
   5. This above spatial attention module is applied to the input feature to output the refined feature maps.
   6. 
   7. But differently from the original SAM implementation, in YOLOv4, a modified SAM is used without applying the maximum and average pooling.
   8. 
7. **Path Aggregation Network (PAN)**:
   1. Another technique used is a modified version of the PANet (Path Aggregation Network). The concept is once more to aggregate information data to get higher accuracy.
   2. Thus, the design of the model during the early Deep Learning ages is quite simple (picture below). Each layer takes input from the previous layer.
   3. The early layers extract localized texture and pattern information to build up the semantic information needed in later layers. However, when we go to the right, the localized information that may be needed to fine-tune the prediction may be lost.
   4. In the image below, taken from Path Aggregation Network (PAN) paper, the path from bottom to top (b) is widened to make it easier to spread low-level information to the top.
   5. In FPN, the localized spatial information travels upward with a red arrow. This is not clearly demonstrated in the image, but the red path goes through around 100+ layers.
   6. PAN introduced a short-cut path (the green path) that requires only about 10 layers to transition to the top. These short-circuit concepts allow fine-grain localized information to be obtained by the upper layers.
   7. 

**Head (Dense Prediction):**

1. Here, we have the same process as in YOLOv3. The network detects the bounding box coordinates (X, Y, W, H) together with the confidence score for a class.
2. The goal of YOLO is to divide the image into a grid of multiple cells and then for each cell to predict the probability of having an object using anchor boxes. The output is a vector with bounding box coordinates and probability classes. Also in the end there is used post-processing techniques such as non-maxima suppression.

Ref.: <https://medium.com/analytics-vidhya/introduction-to-yolov4-object-detection-fcba8bb72449>