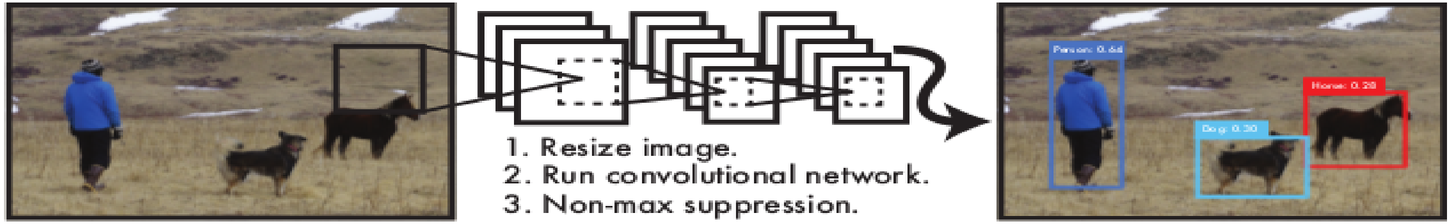
You Only Look Once: Unified, Real-Time Object Detection

Research paper: https://arxiv.org/pdf/1506.02640v5.pdf

Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi
University of Washington, Allen Institute for AI,Facebook AI Research http://pjreddie.com/yolo/

You Only Look Once:
Unified, Real-Time Object Detection

Joseph Redmon[*], Santosh Divvala[*†], Ross Girshick[¶], Ali Farhadi[*†]
University of Washington[*], Allen Institute for AI[†], Facebook AI Research[¶]
http://pjreddie.com/yolo/



Base YOLO model runs at 45 FPS. A smaller version, Fast YOLO, runs astounding 155 FPS second, outperforms DPM (deformable parts models) and R-CNN.

Figure 1: The YOLO Detection System. (1) resizes image to 448 × 448, (2) runs a convolutional network, and (3) thresholds the result by the model's confidence.

YOLO github Code and readme

https://github.com/hualili/opencv/blob/master/deep-learning-2022s/2022F-106-YOLO4-README-Tiny-Yolo4-GPU-YY-2022-9-28.pdf

Tutorial: https://pjreddie.com/darknet/yolo/

GitHub, Inc. (US)

```
git clone https://github.com/pjreddie/darknet
```

Harry Li, Ph.D.

1x1 Convolution for Dimension Reduction and Pooling

The 1x1 convolution enables dimension reduction by reducing the number of channels in convolution layers

1. Suppose the input layers is C⋆H⋆W, where C is its channels. The 1x1 convolution generates one average result in shape H⋆W. The 1x1 kernel (filter) is a vector of length C.
2. Now if you have F 1x1 filters, you get F layers of output, the output shape is F⋆H⋆W. For input layer C⋆H⋆W with F 1x1 convolution (with each of its channel is C), you will get F⋆H⋆W layers.

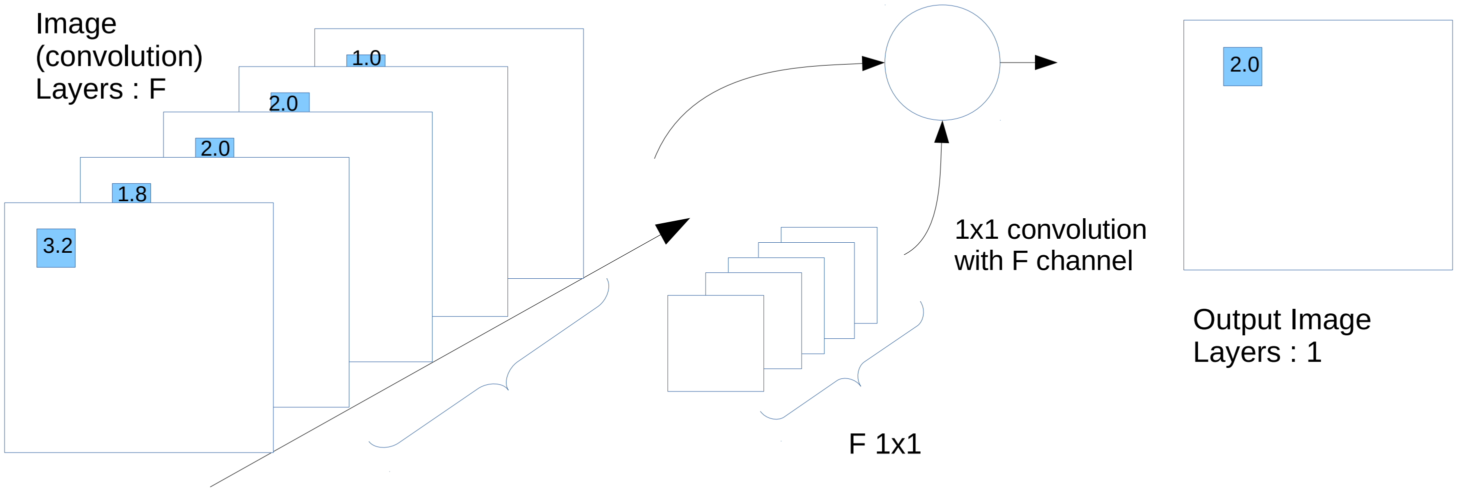


Image resolution
MxN

Average result:
(3.2 + 1.8 + 2.0 + 2.0 + 1.0) / 5 = 2.0

Harry Li, Ph.D.

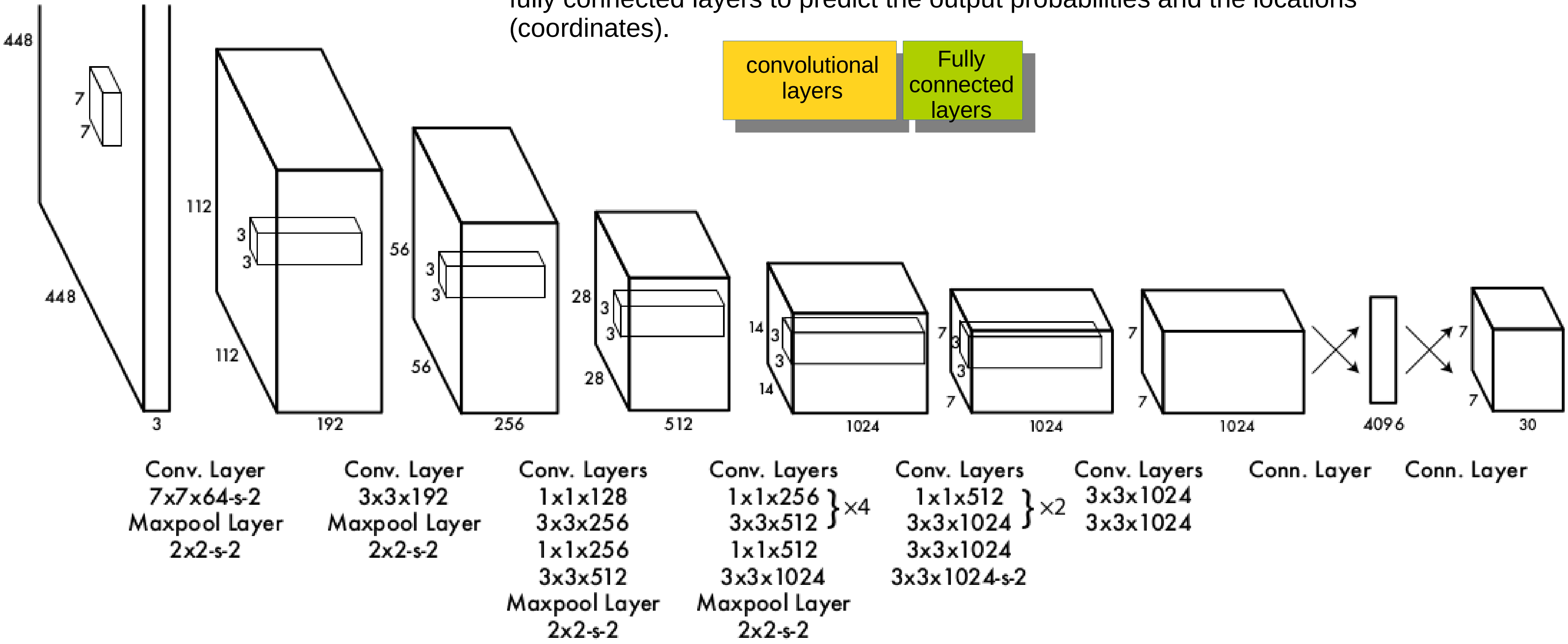
1. Input image size: 448x448x3; 2. Resolution reduction for feature extraction/abstraction Pooling and convolution with stride = 2;

Base Line Yolo Architecture

Design guideline: The block of convolutional layers to extract image features, the fully connected layers to predict the output probabilities and the locations (coordinates).

convolutional layers

Fully connected layers



Harry Li, Ph.D.

Loss Function for YOLO

$$\lambda_{\mathbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\mathbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

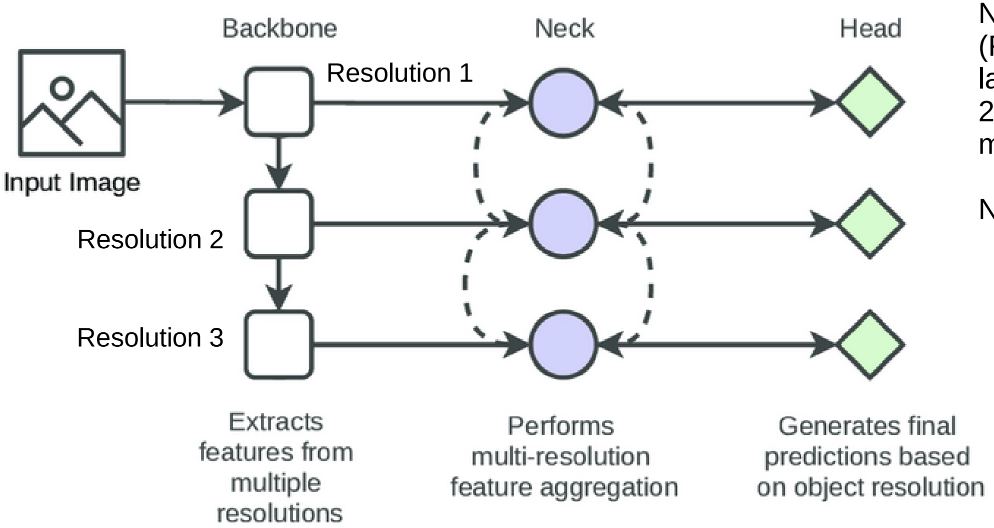
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2$$

Backbone, Neck, Head Module of CNN Architecture

1. A backbone module is the module that extract image features, usually at different resolutions as shown below;

2. A neck module fuses the features of different resolutions.

3. Finally, multiple head modules perform the detection of objects in different resolutions.



https://www.researchgate.net/figure/A-detection-model-contains-a-backbone-neck-head-module-The-backbone-module-exploits_fig1_356638292

Note: Object detection by YOLO, SSD, Mask RCNN and RetinaNet.

Implement a YOLO (v3) object detector from scratch in PyTorch

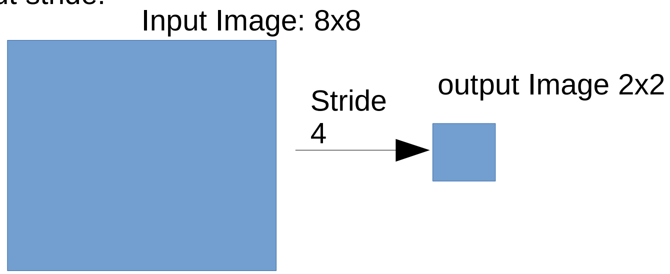
https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/

https://github.com/ayooshkathuria/YOLO_v3_tutorial_from_scratch

/ **YOLO_v3_tutorial_from_scratch**

Note: 1. YOLO only uses convolutional layers, e.g., fully convolutional network (FCN). It has 75 convolutional layers, with skip connections and upsampling layers.
2. No pooling, a convolutional layer with stride 2 for downsampling the feature map.

Note about stride:



Note 1. Images in batches can be processed in parallel by the GPU for speed gain. All images must have the same resolution to be concatenate multiple into a large batch (concatenating many PyTorch tensors into one);
Note 2. YOLO prediction is done by using a convolutional layer which uses 1 x 1 convolutions.

Harry Li, Ph.D.