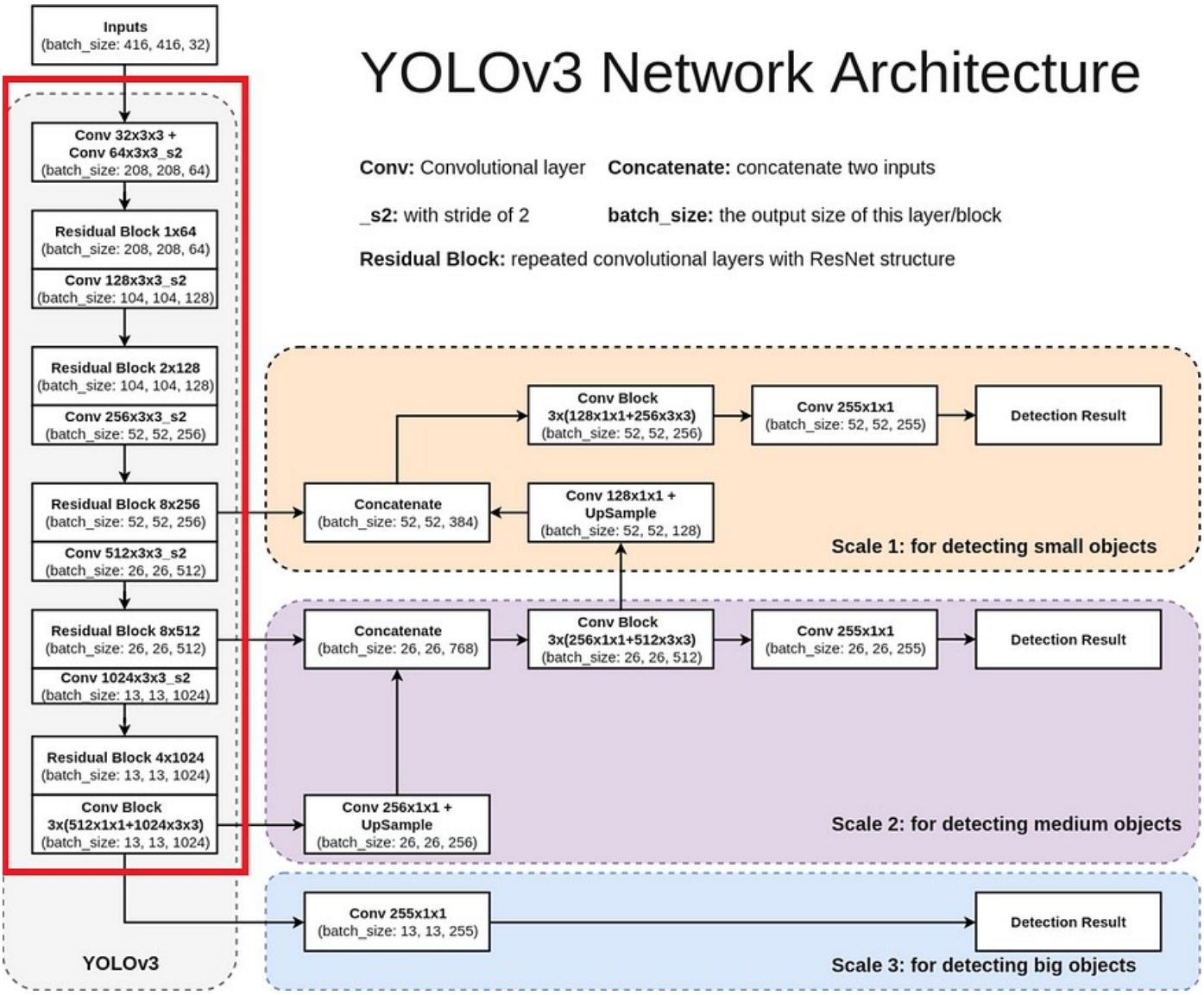
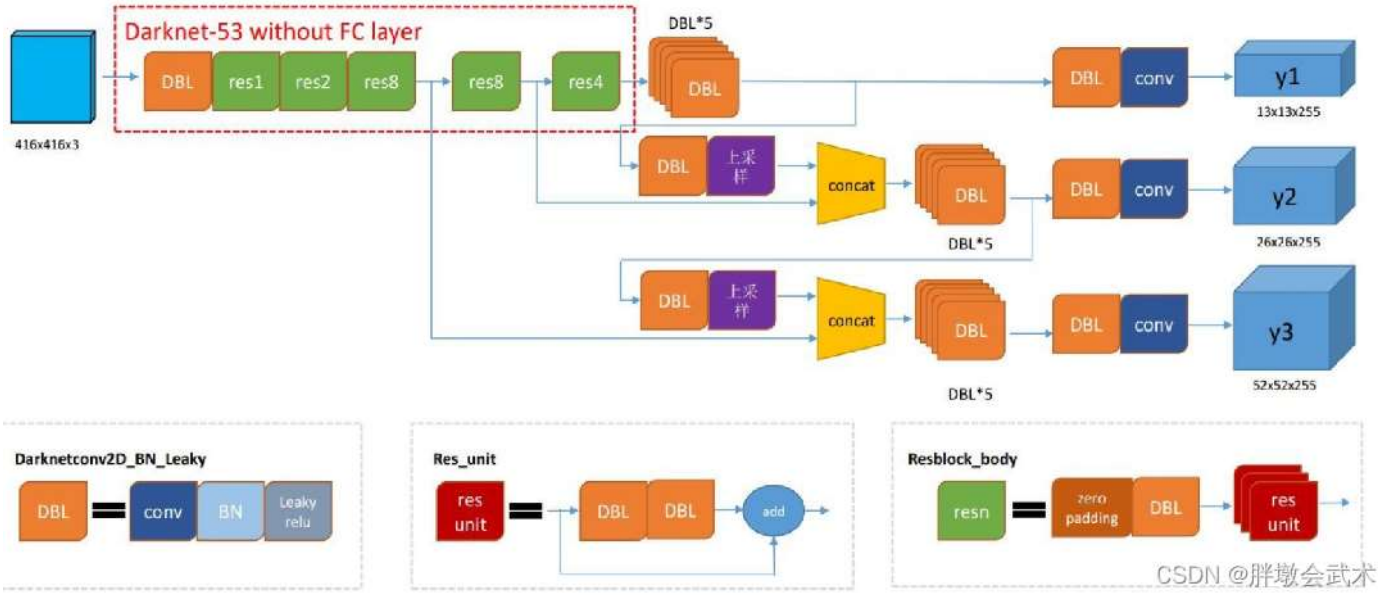
YOLOv3


YOLOv3 Network Architecture
Conv: Convolutional layer
Concatenate: concatenate two inputs
_s2: with stride of 2
batch_size: the output size of this layer/block
Residual Block: repeated convolutional layers with ResNet structure
Inputs
(batch_size: 416, 416, 32)
Conv 32x3x3 +
Conv 64x3x3_s2
(batch_size: 208, 208, 64)
Residual Block 1x64
(batch_size: 208, 208, 64)
Conv 128x3x3_s2
(batch_size: 104, 104, 128)
Residual Block 2x128
(batch_size: 104, 104, 128)
Conv 256x3x3_s2
(batch_size: 52, 52, 256)
Residual Block 8x256
(batch_size: 52, 52, 256)
Conv 512x3x3_s2
(batch_size: 26, 26, 512)
Residual Block 8x512
(batch_size: 26, 26, 512)
Conv 1024x3x3_s2
(batch_size: 13, 13, 1024)
Residual Block 4x1024
(batch_size: 13, 13, 1024)
Conv Block
3x(512x1x1+1024x3x3)
(batch_size: 13, 13, 1024)
YOLOv3
Conv Block
3x(128x1x1+256x3x3)
(batch_size: 52, 52, 256)
Conv 255x1x1
(batch_size: 52, 52, 255)
Detection Result
Concatenate
(batch_size: 52, 52, 384)
Conv 128x1x1 +
UpSample
(batch_size: 52, 52, 128)
Scale 1: for detecting small objects
Concatenate
(batch_size: 26, 26, 768)
Conv Block
3x(256x1x1+512x3x3)
(batch_size: 26, 26, 512)
Conv 255x1x1
(batch_size: 26, 26, 255)
Detection Result
Conv 256x1x1 +
UpSample
(batch_size: 26, 26, 256)
Scale 2: for detecting medium objects
Conv 255x1x1
(batch_size: 13, 13, 255)
Detection Result
Scale 3: for detecting big objects


416x416x3
Darknet-53 without FC layer
DBL
res1
res2
res8
res8
res4
DBL*5
DBL
conv
y1
13x13x255
上采样
concat
DBL*5
y2
26x26x255
y3
52x52x255
Darknetconv2D_BN_Leaky
DBL
conv
BN
Leaky relu
Res_unit
res unit
add
Resblock_body
resn
zero padding
CSDN @胖墩会武术

Major changes compared with YOLOv2:

YOLOv3 在 YOLOv2 的基礎上，**改良了網路 backbone**、利用**多尺度特徵圖** (feature map) 進行檢測 (three colored regions in the above image)、改用**多個獨立的 Logistic regression 分類器**取代 softmax 來預測類別分類。

New Backbone: Darknet-53

YOLOv3 author Joseph Redmon: Darknet-53 is a **hybrid** approach between **Darknet-19** (YOLOv2 backbone) and **residual network stuff** (originated from ResNet).

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 1. **Darknet-53.**

Why the number is 53? There are 53 convolution layers in Darknet-53.

Note: Darknet-53 has 23 residual blocks. (We will see why the author puts residual blocks in Darknet-53)

No max pooling layer found in Darknet-53 (Darknet-19 still has max pooling layer). To achieve downsampling, convolution layers with 3x3 kernel and stride=2 are used.

Advantages of using convolution layer to replace max pooling layer:

Information of the spatial location of the input can be utilized and conserved in the output feature map.

A network with 53 convolution layer is rather deep. **Deep neural network suffers from the vanishing gradient problem/ exploding gradient problem** (why? Chain rule in backpropagation: if many terms in the gradient are less than 1, gradient vanishing; else, gradient exploding), and therefore hinders updating model weights (this can be resolved by normalization in initialization and intermediate layers).

Also, with the network depth increases, the accuracy gets saturated, resulting in larger errors compared with shallower neural network (problematic).

Why? During forward propagation, if the model is too deep for the problem, the layer in traditional convolution layer is difficult to do identity mapping (i.e. input unchanged), and thus some layers would hold back the performance, i.e. performing useless or even adding detrimental computations to the model input. The bigger problem is we do not know which layers are those holding back in the model, and we have no guidance to prune the model for better performance.

Note: this problem is **not due to model overfitting** as both the training error and test error are larger when the model goes deeper. Fore model overfitting, it should have low training error but high testing error.

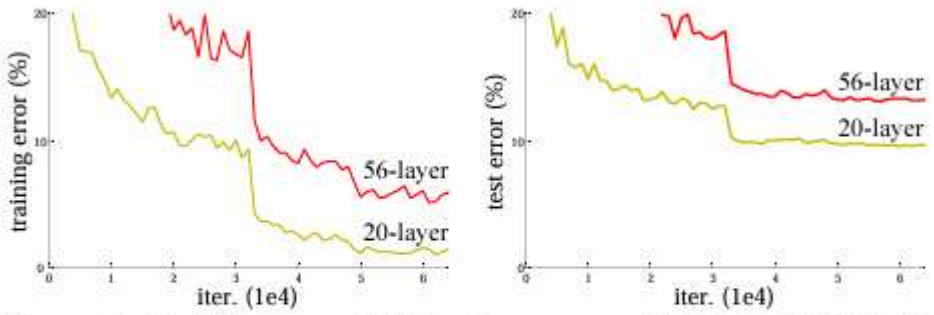


Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

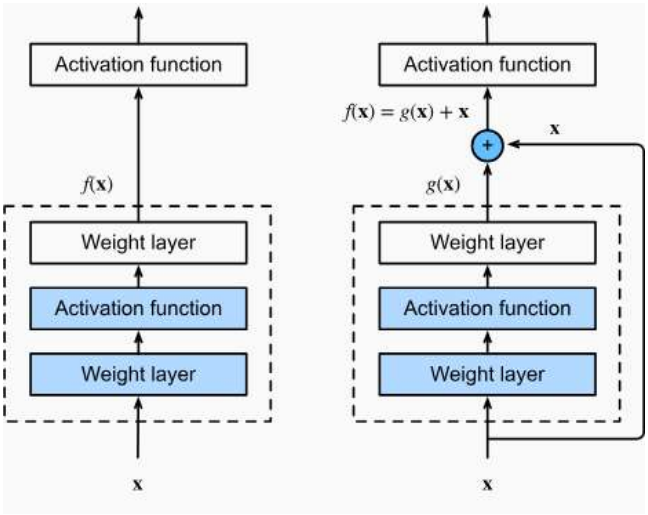
Retrieved from Deep Residual Learning for Image Recognition paper

(**SUPER IMPORTANT**) Breakthrough: Kaiming He proposed residual block architecture in 2015 to tackle the above two problems.

The idea is simple: if the network is too deep to the task, i.e. a shallower network can achieve better, some of the layers in the network can just be **identity mapping**, **skipping some layers** and pass the input to the next layer. Theoretically the deep neural network would behave similar as the shallower counterpart in this case.

How to achieve the **identity mapping** as stated above?

Skip connections. The skip connection connects activations of a layer to further layers by bypassing some layers in between (the right arrow in the figure).



$\boldsymbol{x}$: input, $\boldsymbol{f(x)}$: desired underlying mapping we want the model to learn

Dotted region in the left block: directly learn $\boldsymbol{f(x)}$; (traditional conv block)

Dotted region in the right block: learn the **residual** mapping $\boldsymbol{g(x) = f(x) - x}$ (residual block)

If the model training is saturated, some residual blocks can learn $\boldsymbol{g(x) = 0}$. Then in residual block the output $\boldsymbol{f(x) = x}$, giving the same input from this layer to the next layer, achieving identity mapping.

These are how the residual blocks in Darknet-53 address the degradation problem and the vanishing gradient problem.

由於加深網路層數，Darknet-53 比 Darknet-19 慢的許多，但 Darknet-53 處理速度每秒 78 張圖，還是比同精度的 ResNet 快很多，YOLOv3 依然保持了高性能 (tradeoff between FPS and precision)

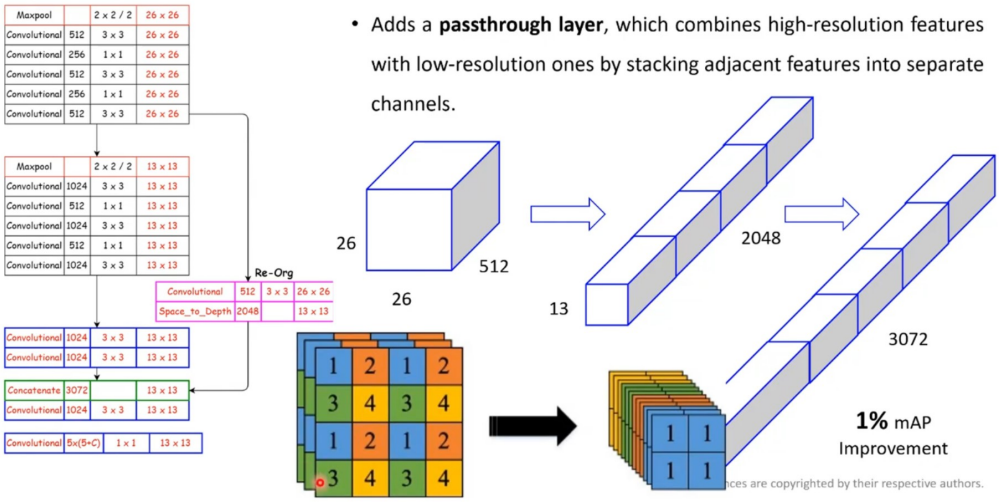
Darknet-53: slightly worse top-1 precision than ResNet-152 but doubles the FPS.

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

Multi-scale Detection

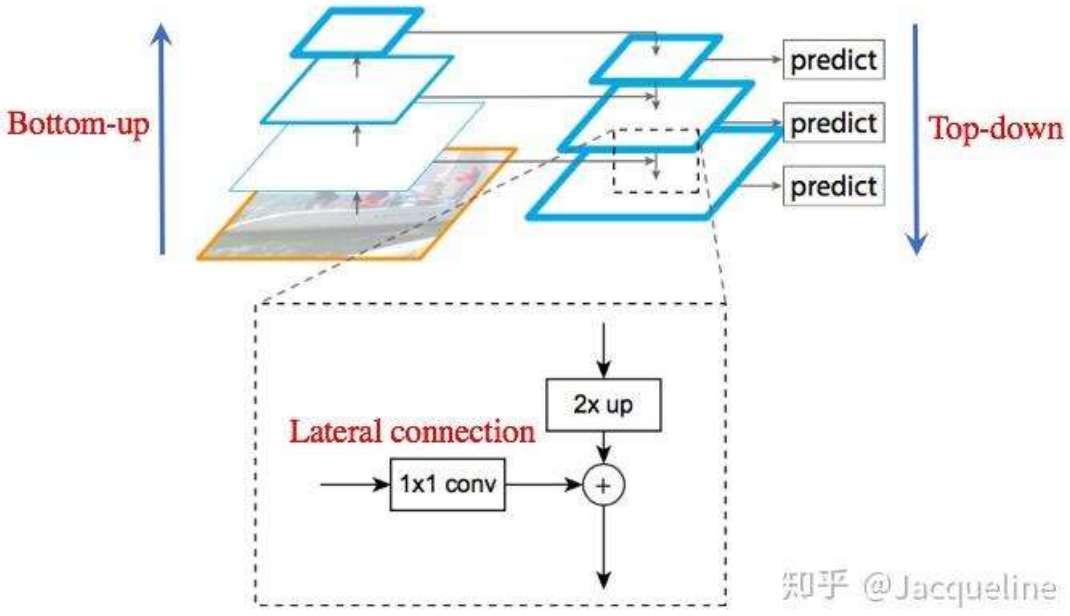
Problem of YOLOv2: information of tiny objects is lost after multiple max pooling and downsampling.

回顾 YOLOv2 的 Fine-Grained Features，它实际类似 Feature Pyramid Network (FPN)，不同之处是 FPN 融合了每一层的特征，而 YOLO v2 只是增加了一层 Passthrough Layer 直通层。’



YOLOv3 uses a similar approach from Feature Pyramid Network (FPN) to do multi-scale detection, especially helpful in tiny object detection.

Framework of FPN:



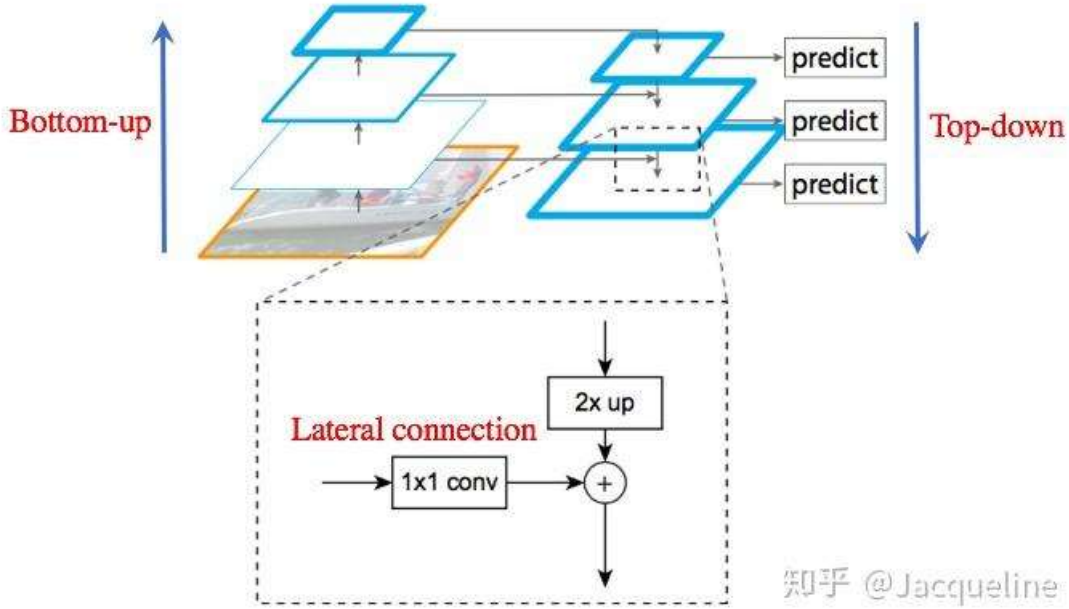
FPN 架構基本上就是**將深層的 feature map 再逐層拉回高解析度，並與先前抽出同解析度的 feature map 結合。**最後，**在多個 feature map 同時進行 dense prediction**。

What are the advantages to extract features from multiple layers? (Why not only using the last feature map?)

This allows the model to gather different information of objects from multiple layers.

- Low level feature (high resolution) 對於**細節保留較多資訊** (capture tiny objects)。
- High level feature (low resolution) 對於**語意 (semantic) 有更好的理解** (capture large objects)。
- 在位置敏感的任務中，如 detection、segmentation 等，兩者資訊不可或缺，需要妥善的結合。

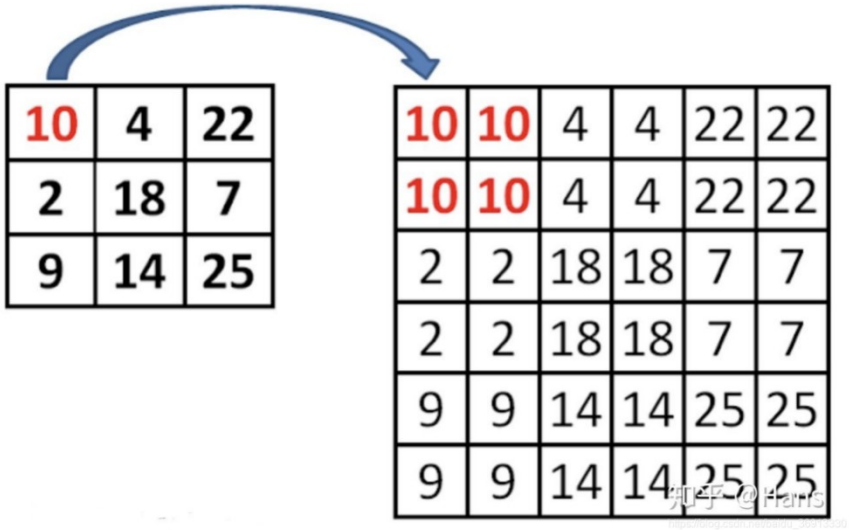
FPN composes of a **bottom-up** and a **top-down** pathway. The **bottom-up** pathway is the **usual convolutional network** for feature extraction. As we go up, the model involves **downsampling the feature map**. With more high-level structures detected, the **semantic value** for each layer increases.



The **top-down** process involves **upsampling the feature map** obtained at higher levels and then propagating the feature map downward. As high-level features contain rich semantic information, through top-down propagation, this semantic information is transferred to the lower-level features, allowing them to also carry abundant semantic details.

Method used for upsampling in FPN and YOLOv3: nearest neighbor interpolation

e.g. 3x3 feature map upsample to 6x6 using nearest neighbor interpolation:



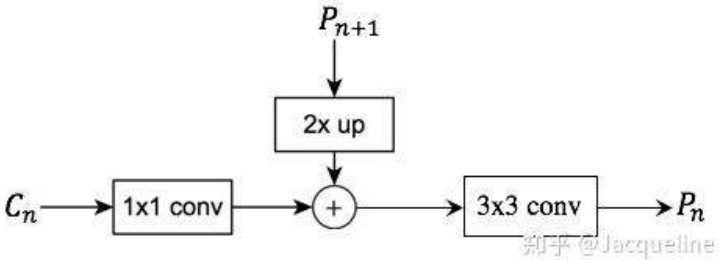
Other types of interpolation: linear, spline.

Nearest neighbor interpolation has low computation cost, but the output feature map is discontinuities and has abrupt changes, often producing blocky or jagged outputs.

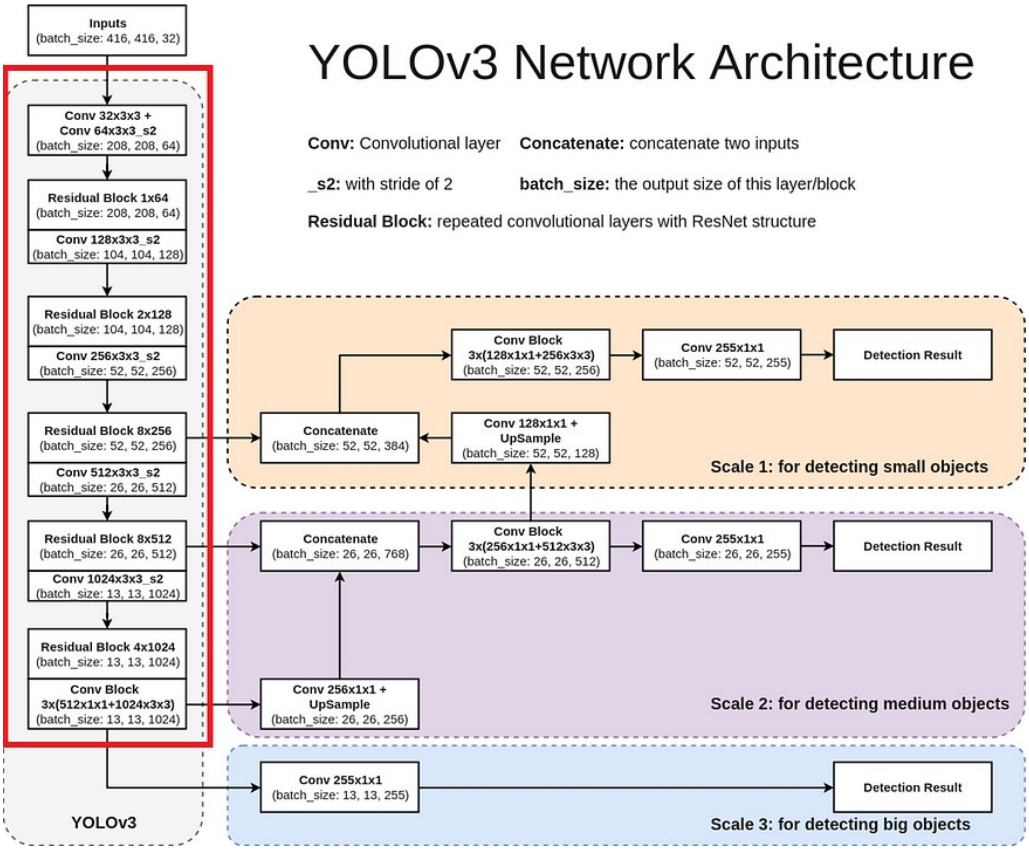
Having bottom-up and top-down alone are not enough. We need a way to combine the high-level features from top-down and the low-level features from intermediate feature maps in bottom-up.

Lateral connection:

1. Do 1x1 convolution of the feature map from each stage C_n to reduce the number of output channels.
2. Do element-wise addition (Note: YOLOv3 uses concatenation) to the upsampled feature map from the previous stage to injects the strong semantic context from the top layers into the fresh spatial details present in the lower layers.
3. Do 3x3 convolution on the feature map from step 2 to obtain the output feature map of this stage to reduce the aliasing effect introduced by the simple upsampling process and extract features.



How is the idea of FPN included in YOLOv3?

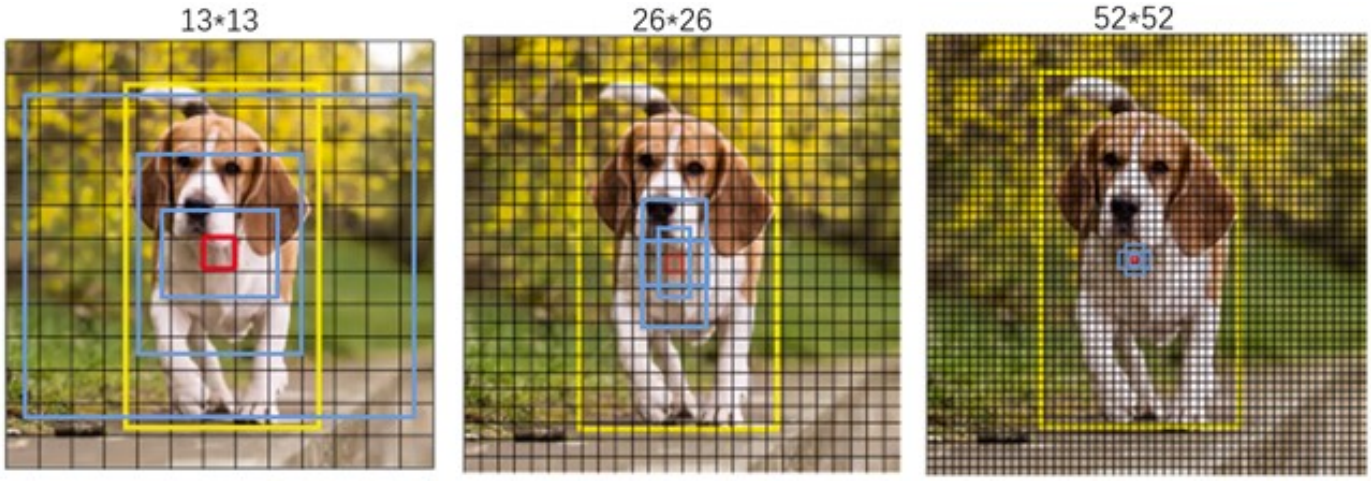


Bottom-up: Darknet-53 (Left part), Input size: 416x416x3

Top-down: Three regions on the right part.

Lateral connection: Concatenate block in three regions

從 Darknet53 其中三層輸出來做 detection，大小分別是 **52x52 (Scale 1: downsampled 8 times)**、**26x26 (Scale 2: downsampled 16 times)**、**13x13 (Scale 3: downsampled 32 times)**。所以每丟入一張圖片到網路中，detection model 將會針對這三種尺寸算出三個 loss (still Regression Loss + Objectness Loss + Classification Loss)。



https://medium.com/ching-i/yolo%E6%BC%94%E9%80%B2-2-85ee99d114a1

每個尺度的 feature map 會預測出 3 個 Anchor prior，而 Anchor prior 的大小則採用 K-means 進行聚類分析 (跟 YOLOv2 的作法一樣)。

- 13x13 feature map (有最大的感受野) 用於偵測大物體，所以用較大的 Anchor prior (116x90), (156x198), (373x326). Output size: 13x13x**255**
- 26x26 feature map (中等的感受野) 用於偵測中等大小的物體，所以用中等的 Anchor prior (30x61), (62x45), (59x119). Output size: 26x26x**255**
- 52x52 feature map (較小的感受野) 用於檢測小物體，所以用較小的 Anchor prior (10x13), (16x30), (33x23). Output size: 52x52x**255**

Total number of anchor boxes in YOLOv2 for 1 image = 13x13x5 = 845

Total number of anchor boxes in YOLOv3 for 1 image = 13x13x3 + 26x26x3 + 52x52x3 = 10647 > > 845.

Why **255** output channels?

Each grid has **3** bounding boxes. Each box has **4** box coordinates (t_x, t_y, t_w, t_h)(the definition of box coordinates follows from YOLOv2), **1** objectness score , and **80** classification scores. Total = **3 x (4 + 1 + 80) = 255**

YOLOv2: 20 classes; YOLOv3: 80 classes, why does the number of classes increase?

Different datasets are used. YOLOv2: VOC2007/2012 (20 classes); YOLOv3: COCO2014/2017 (80 classes)

Bounding Box Prediction

YOLOv3 使用 Logistic regression 來預測每個 bounding box 的 objectness score，以 bounding box 與 ground truth 的 IOU 為判定標準，對每個 ground truth 只分配一個最好的 bounding box。利用這個方式，在做 Detect 之前可以減少不必要的 Anchor 並減少計算量。

- **正例**: 將 IOU 最高的 bounding box ， objectness score 設為 1 (unlike YOLOv2)
- **忽略樣例**: 其他不是最高 IOU 的 bounding box 並且 IOU 大於閾值 (threshold，預測為 0.5) ，則忽略這些 bounding box，不計算 loss
- **負例**: 若 bounding box 沒有與任一 ground truth 對應，則減少其 objectness score

❓ 為什麼 YOLOv3 要將 objectness score 設為 1?

💬 因為 objectness score 是指該 bounding box 是否預測出一個物體的信心程度，是一個二分類。並且在學習小像素物體時，有很大程度會影響 IOU，若像 YOLOv1 使用 bounding box 與 ground truth 的 IOU 作為 objectness score，那 objectness score 始終很小，無法有效學習，導致檢測 Recall 不高

❓ 為什麼要有忽略樣例?

💬 由於 YOLOv3 使用了多尺度的 feature map 進行檢測，而不同尺度的 feature map 之間會有重合檢測的部分。例如檢測一個物體，在訓練時他被分配到的檢測框是第一個 feature map 的第三個 bounding box，IOU 達 0.98，此時恰好第二個 feature map 的第一個 bounding box 與該 ground truth 的 IOU 達 0.95，也檢測到了該 ground truth，如果此時給其 objectness score 強行打 0 的標籤，網絡學習效果會不理想

Class Prediction

YOLO 之前都是使用 softmax 去分類每個 bounding box，**但 softmax 只適用於單標籤多分類 (基於類別是互斥的假設)**，而預測目標裡可能有重疊的標籤 (屬於多個類並且類別間有相互關係)，比如 Person 和 Women

因此 YOLOv3 改採用多個獨立的 **Logistic regression 分類器 (可以對多標籤進行多分類預測)** 替代，使用 binary cross-entropy 作為 loss function，並且發現準確率不會下降

$$loss = -\sum_{i=1}^{n} y_i \log(\hat{y_i}) + (1 - y_i) \log(1 - \hat{y_i}),$$

$$y_i \; is \; true \; value, \hat{y_i} \; is \; predict \; value$$

Loss function

✧ bndBox 中心點座標x, y 與寬高

- 一樣沿用 YOLOv2 的 offset 做計算：
  - ◆ (b_{xi}, b_{yi}) 為 ground truth的 bndBox中心點座標，$(\widehat{b_{xi}}, \widehat{b_{yi}})$ 為預測 bndBox的中心點座標
  - ◆ (b_{wi}, b_{hi}) 為 ground truth的 bndBox寬高，$(\widehat{b_{wi}}, \widehat{b_{hi}})$ 為預測 bndBox的寬高
- 1_{ij}^{obj} 為第 i 個網格中第 j 個 bndBox是否為正例，若是則輸出 1；否則為 0
- $\lambda_{coord} = 5$，使模型著重於物體定位的能力

✧ bndBox物件 confidence

- 1_{ij}^{noobj} 為第 i 個網格中第 j 個 bndBox是否為負例，若是則輸出 1；否則為 0
- 若是忽略樣例(不是正例及負例)，則皆輸出0 (不產生任何 loss)
- 有物體的係數為1，無物體的係數 $\lambda_{nocoord} = 0.5$，使得模型專注於有物體的辨識，並降低找到無物體的情形

YOLOv3 在 bndBox 座標與寬高的loss計算多乘了 $(2 - b_{wi} \times b_{hi})$ 係數並且在confidence score 跟 class 的 loss function 使用 binary cross-entropy

loss function =

$$\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}(2-b_{wi}\times b_{hi})[(b_{xi}-\widehat{b_{xi}})^2+(b_{yi}-\widehat{b_{yi}})^2]$$

bndBox的中心點座標計算loss

$$+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}(2-b_{wi}\times b_{hi})[(b_{wi}-\widehat{b_{wi}})^2+(b_{hi}-\widehat{b_{hi}})^2]$$

bndBox的寬高計算loss

$$-\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}c_{ij}\log(\widehat{c_{ij}})+(1-c_{ij})\log(1-\widehat{c_{ij}})$$

bndBox 有物件的confidence計算loss

$$-\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{noobj}c_{ij}\log(\widehat{c_{ij}})+(1-c_{ij})\log(1-\widehat{c_{ij}})$$

bndBox 無物件的confidence計算loss

$$-\sum_{i=0}^{S^2}\sum_{j=0}^{B}1_{ij}^{obj}\sum_{ceclasses}[p_{ij}(c)\log(\widehat{p_{ij}}(c))+(1-p_{ij}(c))\log(1-\widehat{p_{ij}}(c))]$$

bndBox物件類別計算loss

We can see that the loss function still consists of the three losses in YOLOv1 and YOLOv2 (regression loss (1^{st} and 2^{nd} terms) + objectness loss (3^{rd} and 4^{th} terms) + classification loss (5^{th} term)), but **it uses BCE loss for objectness loss and classification loss**.

Note: YOLOv3 is the last YOLO from Joseph Redmon, who is the main author of YOLOv1 and YOLOv2.

References:

YOLO 演進 — 2

https://medium.com/ching-i/yolo%E6%BC%94%E9%80%B2-2-85ee99d114a1

Yolov3 詳細解說

https://b10515007.medium.com/yolov3-%E8%A9%B3%E7%B4%B0%E8%A7%A3%E8%AA%AA-22e3da36260a