IDC410
A course on Image Processing and Machine Learning
(Lecture 16)

Shashikant Dugad,

IISER Mohali

https://medium.com/biased-algorithms/batch-normalization-vs-layer-normalization-c44472883bf2

Batch Normalization (BN) layers

- Batch Normalization is a supervised learning technique that converts interlayer outputs of neural network into a standard format, called normalizing.

- In machine learning, "batch normalization" refers to a technique used to stabilize and accelerate the training process of neural networks by normalizing the inputs to each layer within a mini-batch of data, essentially re-centering and re-scaling the activations to improve learning speed and generalization ability by reducing internal covariate shift.

- In the context of YOLOv3, "batch normalization" refers to a technique applied after convolutional layers that helps stabilize the training process by normalizing the output activations of each layer, essentially ensuring that the data entering subsequent layers has a consistent distribution, leading to faster and more reliable training of the object detection model; it essentially acts as a regularization method to improve the network's performance by mitigating "internal covariate shift" where the distribution of activations in a layer can significantly change due to updates in previous layers.

- This approach leads to faster learning rates since normalization ensures there’s no activation value that’s too high or too low, as well as allowing each layer to learn independently of the others

Batch Normalization (BN)

- BN blends the input values across the mini-batch so that each layer gets a well-mixed input. In BN, each feature across the entire mini-batch is standardized to have a mean of zero and a standard deviation of one. Here's how it does this:
- **Normalization:** It calculates the mean and variance for each feature across the current mini-batch.
- **Scaling and Shifting:** After normalizing the data, it applies a scaling factor (gamma) and a shifting factor (beta). These are learnable parameters, meaning the network can optimize them during training for better performance.
- So, each time data passes through a layer, BN ensures it's appropriately normalized, much like a quality check. The formula for Batch Normalization looks like this:

$$\hat{x}^{\,i} = \frac{x^i - \mu_B}{\sqrt{\sigma_B^2 + \in}}\ \gamma + \beta$$

- x^i: Input value in i^{th} neuron in batch. μ_B and σ_B mean and variance of mini-batch. ε is a small constant added to avoid division by zero condition. β and Y are learnable parameters that scale and shift the normalized values.

Batch Normalization: Advantages and Limitations

Advantages

- ***Faster Convergence:*** With inputs normalized, the network trains faster because each layer receives more stable data.
- ***Regularization Effect:*** By introducing a slight noise due to mini-batch variation, BN has a regularizing effect, reducing the need for other techniques like dropout.
- ***Higher Learning Rates:*** Since BN reduces internal covariate shift, you can afford to use higher learning rates, making the training process even more efficient.

Limitations

- ***Dependent on Mini-Batch Size:*** If your mini-batch size is too small, the statistics (mean and variance) might not be reliable, leading to suboptimal performance.
- ***Not Suitable for Recurrent Networks:*** In scenarios like RNNs, where the input sequence length varies, BN isn't as effective due to the lack of consistent mini-batches.

Layer Normalization (LN)

- Instead of blending across the batch like BN, it focuses on each data point individually. It computes the mean and variance across the features within a layer for each data point.
- It ensures that each sample is treated independently, making it especially effective in models like RNNs or transformers where input sequences can vary in length

$$\hat{x}^{\,i} = \frac{x^i - \mu_{LN}}{\sqrt{\sigma_{LN}^2 + \in}}\ \gamma + \beta$$

- ***Independence from Batch Size***: LN works regardless of how many samples you have in a batch, making it perfect for tasks where batch size is limited or varies.
- ***Better for RNNs and Transformers***: If you're dealing with sequence data or NLP tasks, LN is often more effective than BN.
- ***Effective in Sequence Modelling:*** It stabilizes the learning process in models that deal with long-term dependencies.

Layer Normalization (LN)

Limitations

- ***Less Effective in CNNs:*** **In convolutional networks, where spatial invariance and batch-based learning are crucial, BN tends to outperform LN.**
- ***Computational Cost:*** **LN can be slightly more computationally intensive in some cases since it normalizes over a larger set of parameters**

Use Case

- **BN is recommended for FFN or CNN where you have large datasets and can afford a decent batch size**
- **LN is recommended for Recurrent Neural Networks (RNNs), Long Short-Term Memory networks (LSTMs), and transformer models used in Natural Language Processing (NLP).**

Residual Blocks

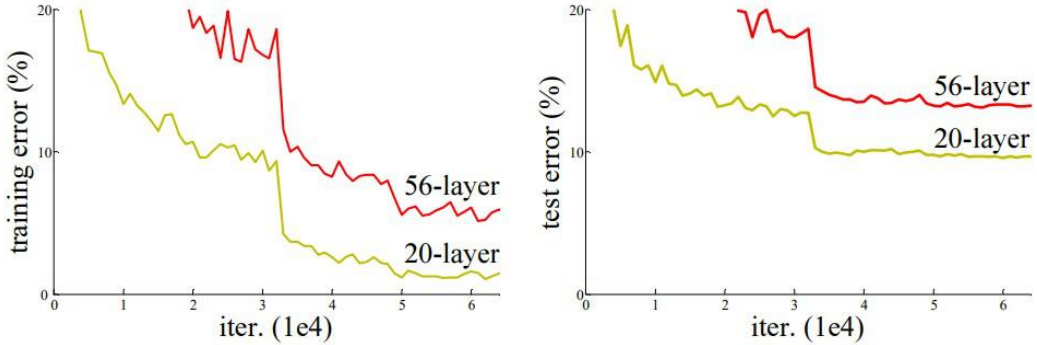
Source:

https://www.geeksforgeeks.org/introduction-to-residual-networks/

https://towardsdatascience.com/what-is-residual-connection-efb07cab0d55

Residual Network (Residual Blocks)

- Traditionally, in feedforward neural networks, data flows through each layer ***sequentially***: The ***output*** of a given layer is the ***input*** for THE ***very*** next layer!
- Increasing the number of convolution layers is a effective way to improve performance of the network
  - With the idea being that, the first layers may detect edges, and the subsequent layers at the end may detect recognizable shapes.
- But if we have ***large*** # of CNN layers **(>~20)** to the network, then its performance suffers and it attains a low accuracy due to the vanishing gradient problem arising from deeper layers.
- A concept of ***Residual Network (Residual Blocks)*** is introduced to solve the problem of the vanishing gradient
- In this network, we use a technique called ***skip connections***. The skip connection connects activations of a layer to a further layers ***(NOT immediate next layer!)*** by skipping some layers in between.

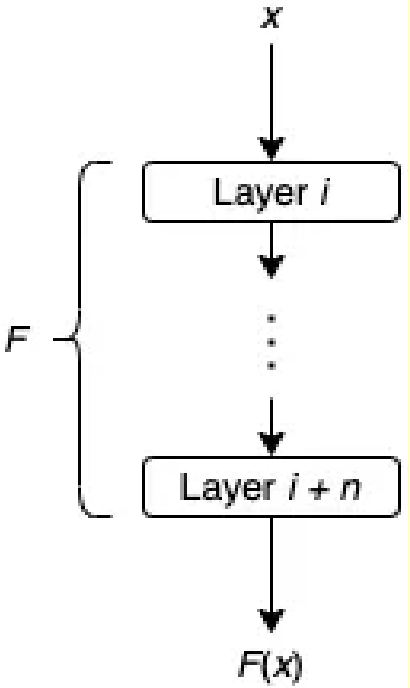


Resnet (Residual Layer)

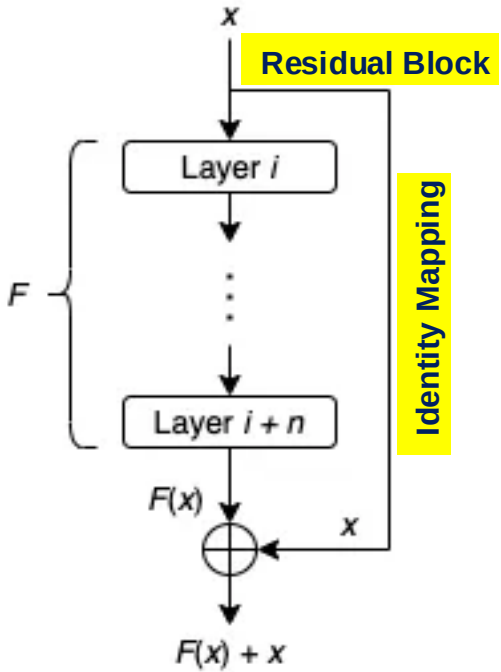
- Consider a sequence of layers, layer ***i*** to layer ***i+n***, and let ***F*** be the function which is resultant output of these layers. Denote the input for layer ***i*** by ***x***
- In ***traditional feedforward*** setting, ***x*** will sequentially go through these layers one by one, and the outcome of layer ***i+n*** is ***F(x)***
- The residual (skip) connection first applies identity mapping to ***x***, then it performs element-wise addition ***F(x)+x***
  - In literature, the whole architecture that takes an ***input x*** and produces output ***F(x)+x*** is usually called a ***residual block or a building block***. Quite often, a residual block will also include an activation function such as ***ReLU*** applied to ***F(x)+x***
- If ***F(x)*** and ***x*** have different dimensions, then we replace the identity mapping ***x*** with a linear transformation (i.e. multiplication by a ***matrix W***), and then perform ***F(x)+Wx***
- See figures in next slide…

Source: https://www.geeksforgeeks.org/introduction-to-residual-networks/

Resnet (Residual Layer)



Traditional FFN WITHOUT Residual Network



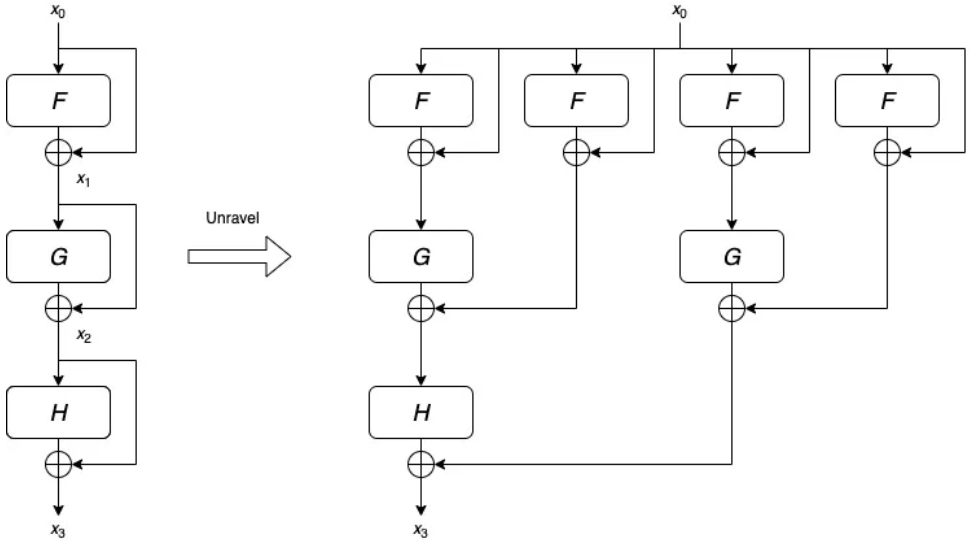
Traditional FFN WITH Residual Network

The training process of a neural network with residual connections is empirically shown to converge much more easily, even if the network has several hundreds layers

Residual block is also called identity connections. Resnets are made by stacking these residual blocks together

Because of the skip, the weights in the convolution layers learn features that are not yet learned by the previous layers

Network with 3 Skip Connections (3 Residual Blocks)



$x_3 = H(x_2) + x_2$

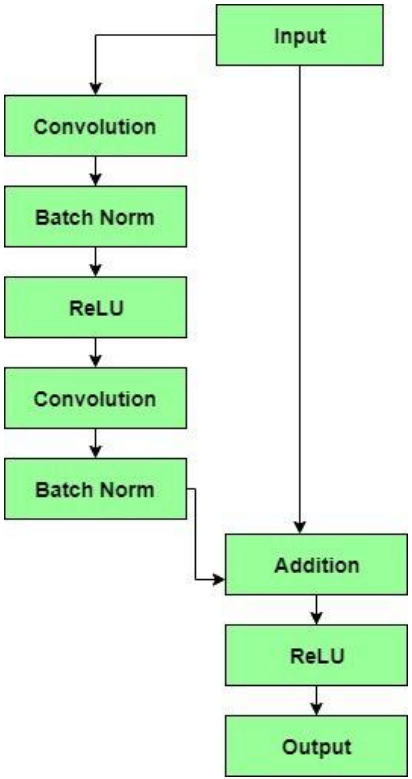
$= H(G(x_1) + x_1) + G(x_1) + x_1$

$= H(G(F(x_0) + x_0) + F(x_0) + x_0) + G(F(x_0) + x_0) + F(x_0) + x_0$

It is to be noted that, there are $2^3 = 8$ terms of the input x_0 that contribute to the output x_3. Hence, we can also view this network as a collection of 8 paths, of lengths 0, 1, 2 and 3, as illustrated in the left figure

Residual connection does not resolve the vanishing gradient problems, rather, it avoids those problems by having shallow networks in the ***ensembles***

Skip Connection: Typical implementation



Feature Pyramid Network (FPN)

Slides made using

1. **Lectures given by Kapil Sachdeva on YouTube**

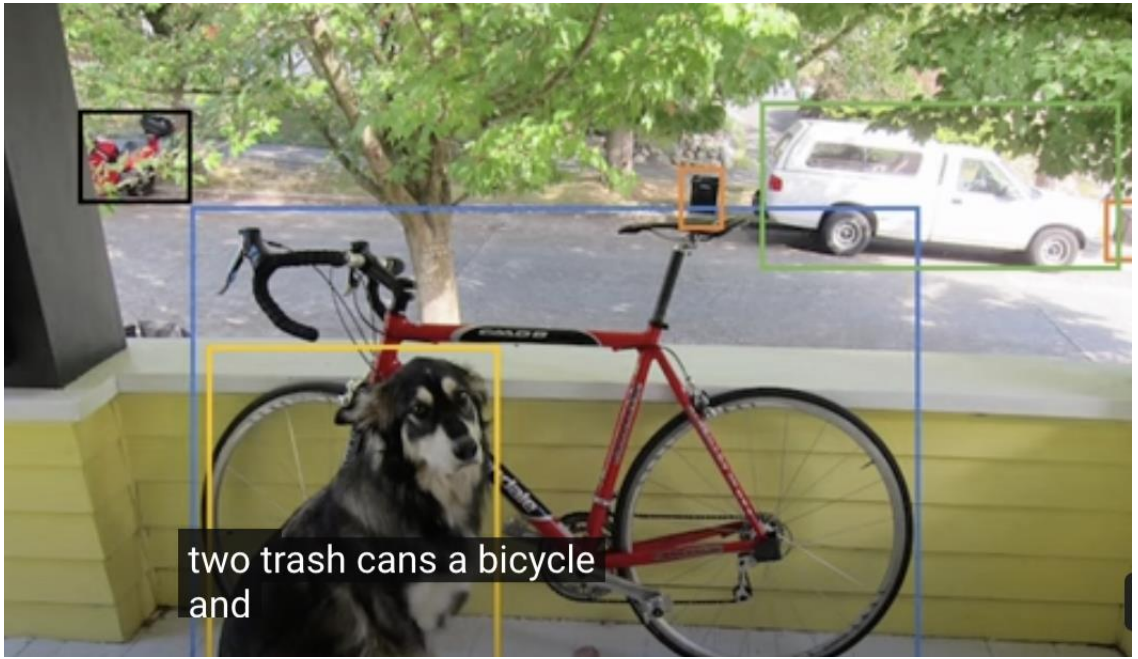
2. **https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c**

Feature Pyramid Network (FPN)

What is to be determined?

- **Type of Class (l)**
- **Size of Object (L, W)**
- **Position of Object (X_0, Y_0)**
- **Objectiveness (Probability)**

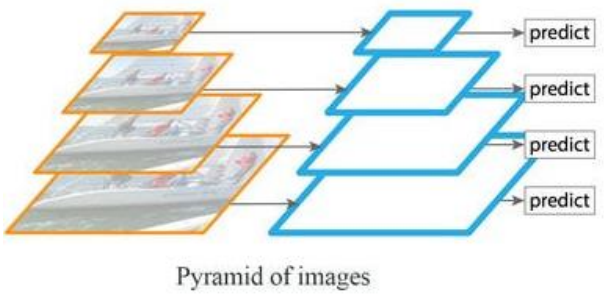
- **Full Description of an identified object requires 5 integers and 1 Real number**



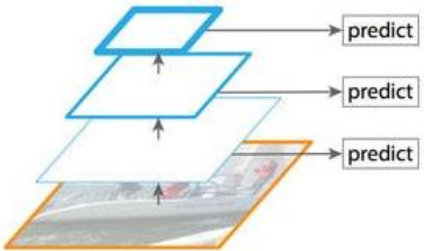
Feature Pyramid Network (FPN)

- Detecting objects at different scales is challenging
  - Same object may look smaller or bigger depending on its distance from the camera.
  - Object of one type may be of smaller size as compared to other type of object
- Pyramid of the image at different scale may be an option to detect objects at different scales. However it is computational taxing and not so accurate
- Pyramid of feature maps may be more appropriate
  - Feature maps closer to the image layer composed of low-level structures with better spatial resolution. They may not be effective for accurate object detection.
  - Feature map at the top may have higher semantic content require for accurate object detection but coarser spatial resolution

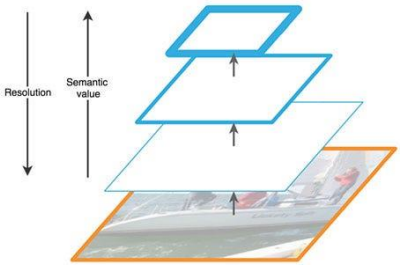
https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c



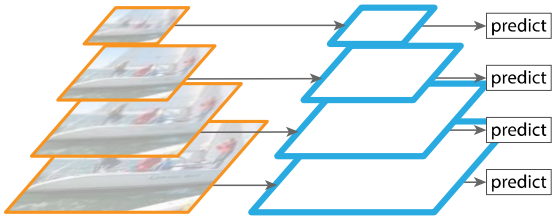
Pyramid of images



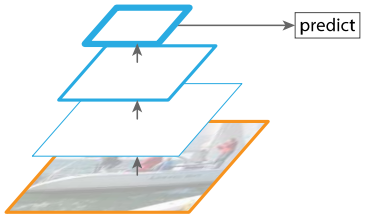
Pyramid of feature maps



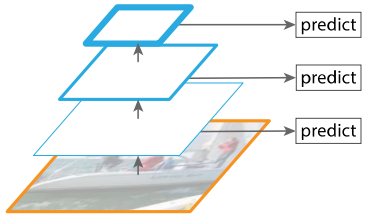
Feature Pyramid Network (FPN)



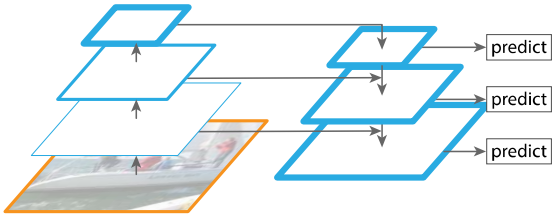
(a) Featurized image pyramid



(b) Single feature map



(c) Pyramidal feature hierarchy



(d) Feature Pyramid Network

(a) Using an image pyramid to build a feature pyramid. Features are computed on each of the image scales independently, which is slow. (b) Recent detection systems have opted to use only single scale features for faster detection. (c) An alternative is to reuse the pyramidal feature hierarchy computed by a ConvNet as if it were a featurized image pyramid. (d) Our proposed Feature Pyramid Network (FPN) is fast like (b) and (c), but more accurate. In this figure, feature maps are indicate by blue outlines and thicker outlines denote semantically stronger features

arXiv:1612.03144v2 [cs.CV] 19 Apr 2017

Feature Pyramid Network (FPN)