Modern CNN Architectures

ML Instruction Team, Fall 2022

CE Department
Sharif University of Technology

Ali Sharifi-Zarchi
Behrooz Azarkhalili
Koorosh Moslemi

Components of CNNs

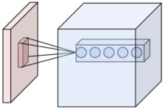


Figure: Convolution Layers

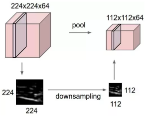


Figure: Pooling Layers

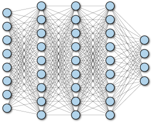


Figure: Dense Layers

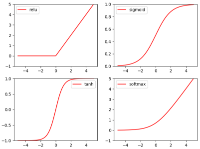


Figure: Activation Functions

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figure: Batch Normalization

Components of CNNs

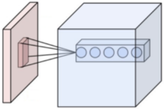


Figure: Convolution Layers

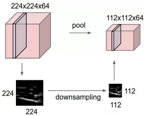


Figure: Pooling Layers

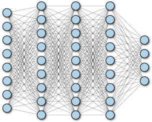


Figure: Dense Layers

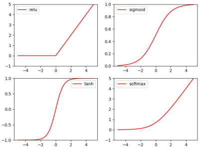


Figure: Activation Functions

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1\ldots m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figure: Batch Normalization

How should we put them together?

LeNet

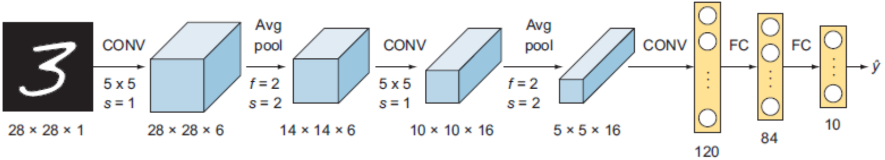


Figure: [1]

- Conv filters were 5x5 , applied at stride 1
- Subsampling (Pooling) layers were 2x2 applied at stride 2

AlexNet

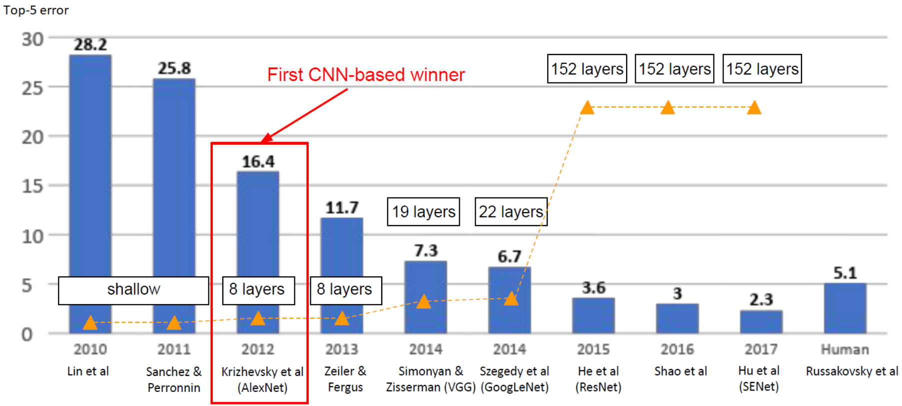


Figure: ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

AlexNet

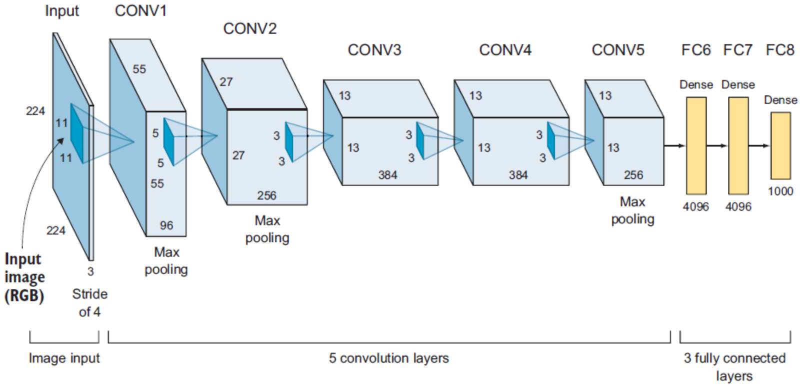


Figure: [1]

- (CONV1): 96 11x11 filters applied at stride 4

AlexNet

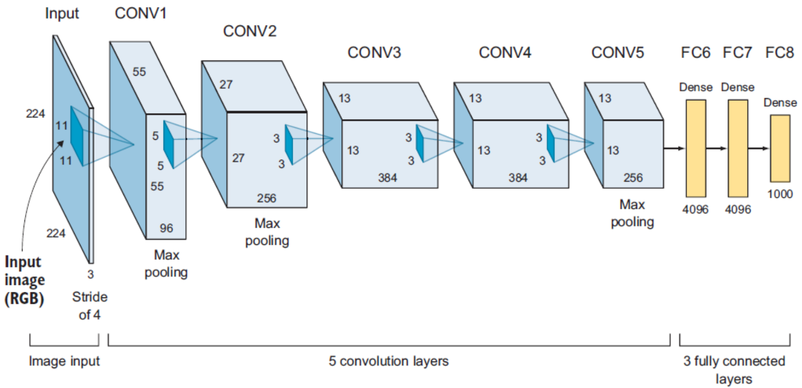


Figure: [1]

- (CONV1): 96 11x11 filters applied at stride 4
- Why is the output volume size 55?

AlexNet

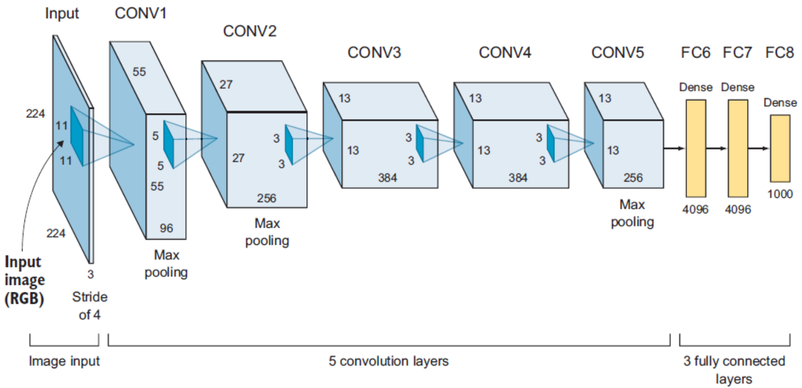


Figure: [1]

- (CONV1): 96 11x11 filters applied at stride 4
- Why is the output volume size 55?
- (227-11)/4+1 = 55

AlexNet

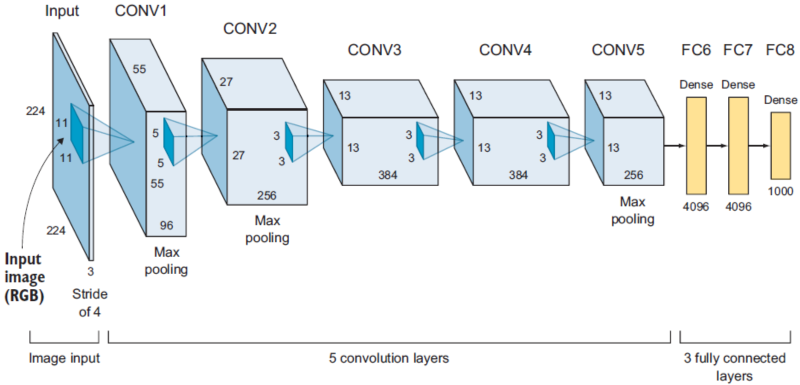


Figure: [1]

- (CONV1): 96 11x11 filters applied at stride 4
- What is the total number of parameters in the first layer?

AlexNet

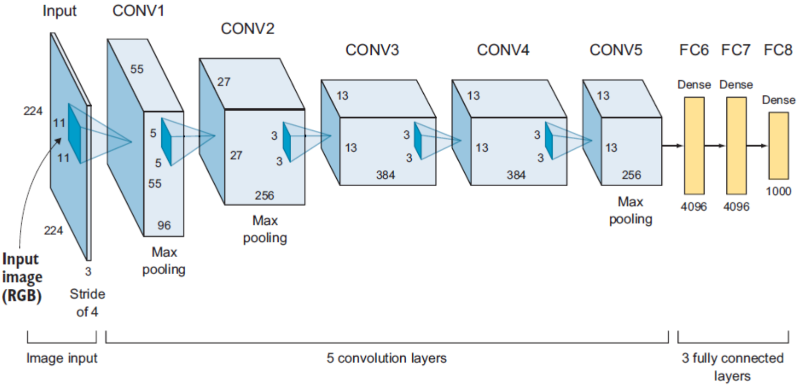


Figure: [1]

- (CONV1): 96 11x11 filters applied at stride 4
- What is the total number of parameters in the first layer?
- (11*11*3 + 1)*96 = 35K

AlexNet

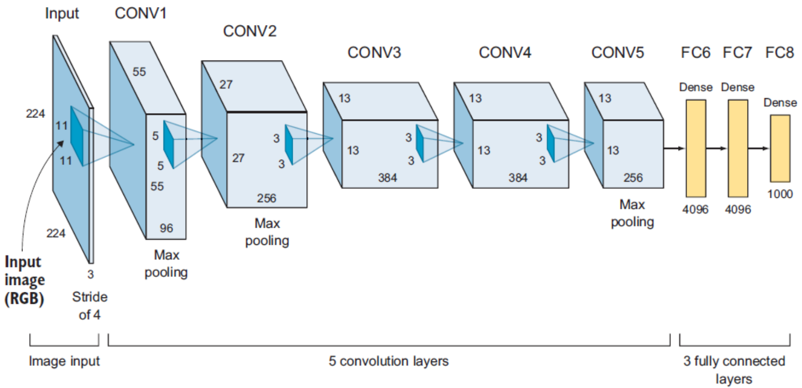


Figure: [1]

- (POOL1): 3×3 filters applied at stride 2

AlexNet

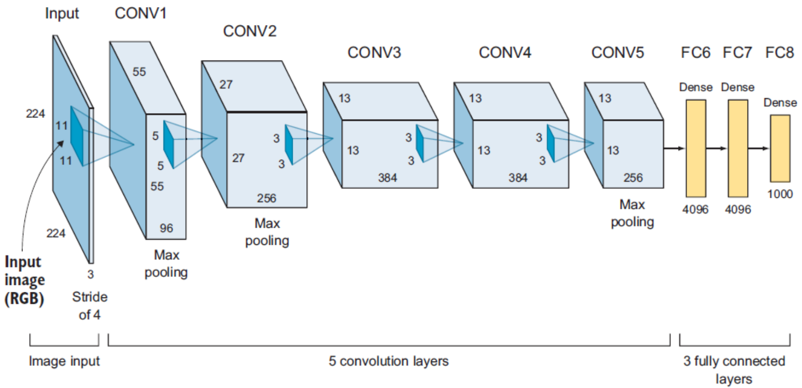


Figure: [1]

- (POOL1): 3×3 filters applied at stride 2
- Why is the output volume size 27?

AlexNet

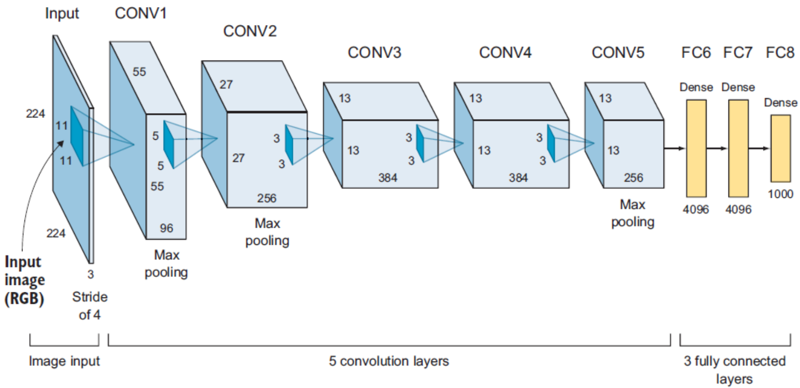


Figure: [1]

- (POOL1): 3×3 filters applied at stride 2
- Why is the output volume size 27?
- (55-3)/2+1 = 27

AlexNet

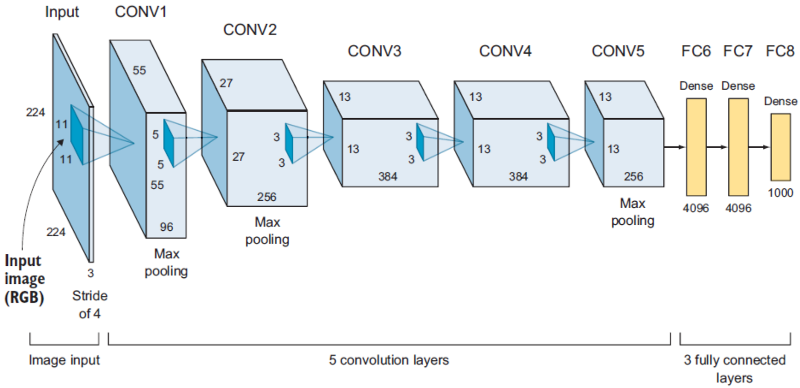


Figure: [1]

- (POOL1): 3×3 filters applied at stride 2
- What is the number of parameters?

AlexNet

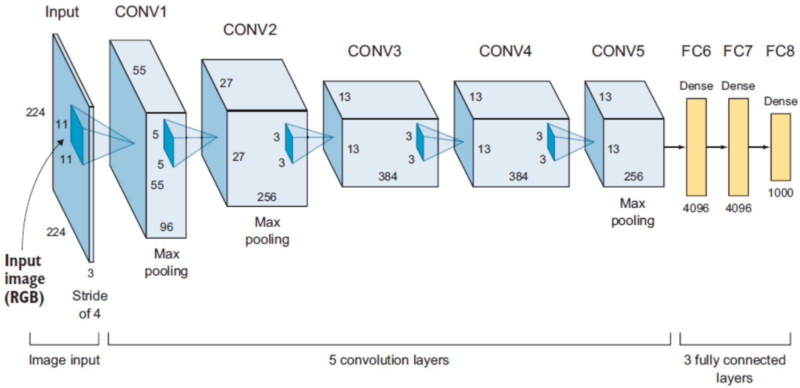


Figure: [1]

- (POOL1): 3×3 filters applied at stride 2
- What is the number of parameters?
- Zero!

How is AlexNet Different from LeNet?

- 62M parameters vs 61K parameters
- ReLU vs Sigmoid (why?)
- Use of Regularization Techniques (Dropout, Weight Decay) (why?)
- Use of Normalization (why?)

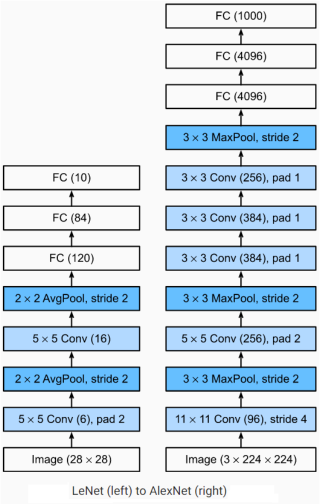


Figure: [3]

VGGNet

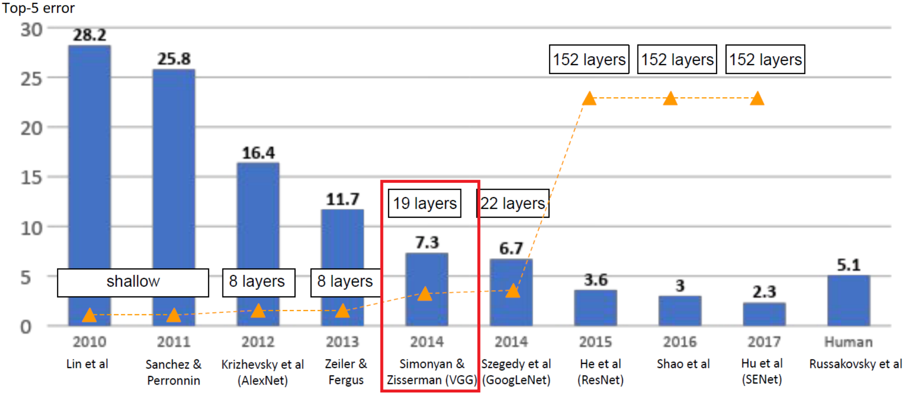


Figure: ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

VGGNet

Do you see any difference?

Figure: [2]

VGGNet

Do you see any difference?

- Smaller filters
- Deeper networks
- Only 3×3 CONV with stride 1, pad 1 and 2×2 MAX POOL with stride 2

Figure: [2]

VGGNet

Why use smaller filters? (3×3 conv)

Figure: [2]

VGGNet

Why use smaller filters? (3×3 conv)

- Stack of three 3×3 conv (stride 1) layers has same effective receptive field as one 7×7 conv layer

Figure: [2]

VGGNet

What is the effective receptive field of three 3×3 conv (stride 1) layers?

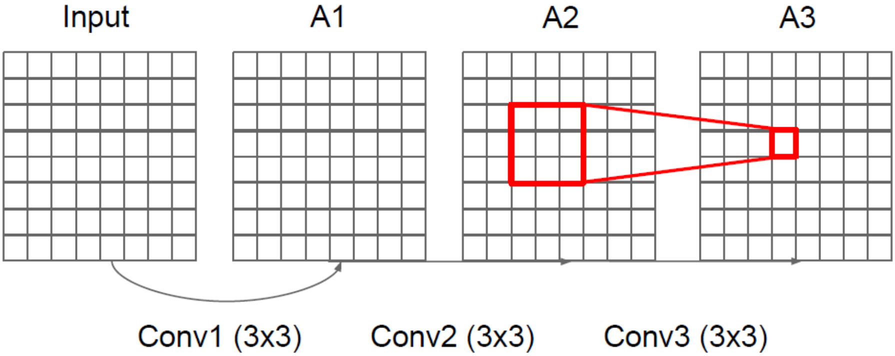


Figure: [2]

VGGNet

What is the effective receptive field of three 3×3 conv (stride 1) layers?

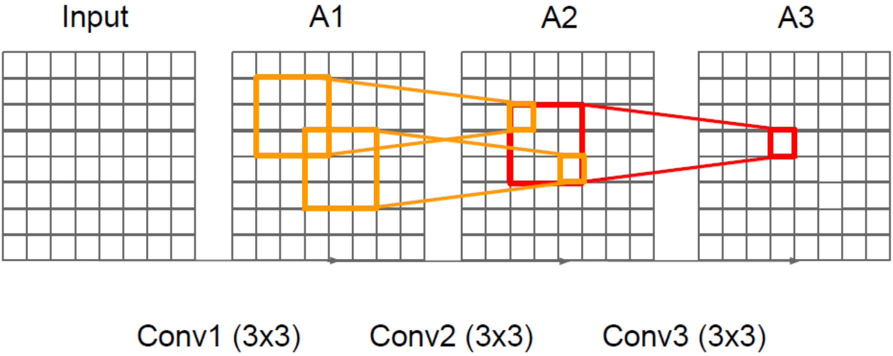


Figure: [2]

VGGNet

What is the effective receptive field of three 3×3 conv (stride 1) layers?

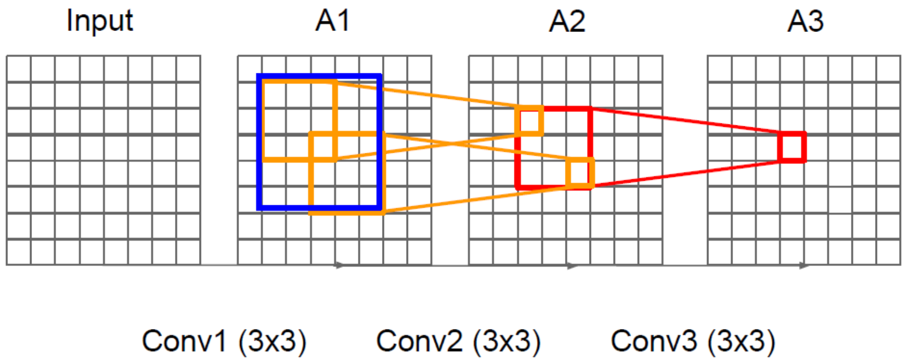


Figure: [2]

VGGNet

What is the effective receptive field of three 3×3 conv (stride 1) layers?

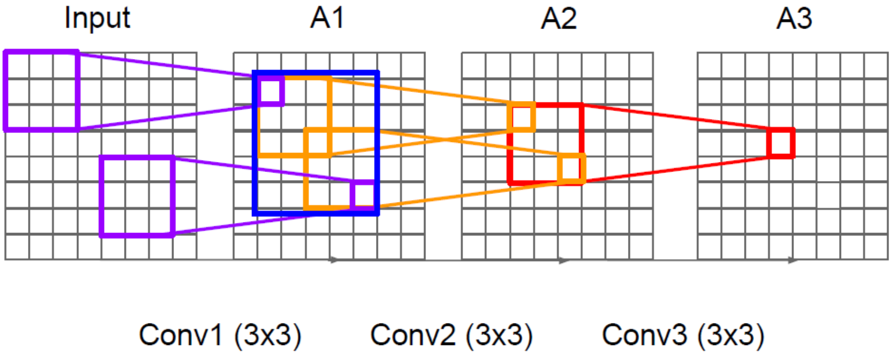


Figure: [2]

VGGNet

What is the effective receptive field of three 3×3 conv (stride 1) layers?

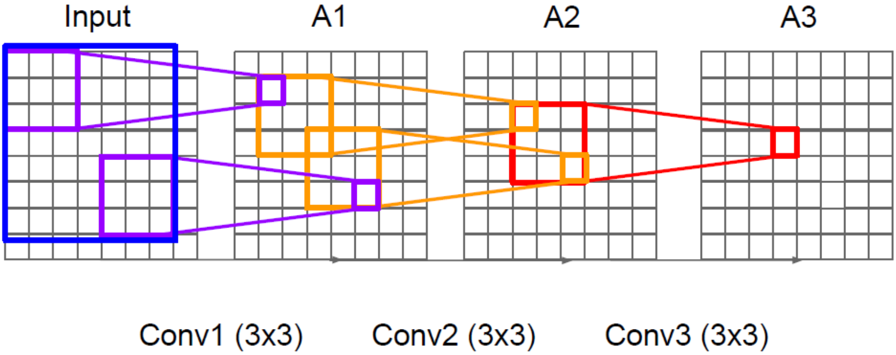


Figure: [2]

VGGNet

Why use smaller filters? (3×3 conv)

- Stack of three 3×3 conv (stride 1) layers has same effective receptive field as one 7×7 conv layer

Figure: [2]

VGGNet

Why use smaller filters? (3×3 conv)

- Stack of three 3×3 conv (stride 1) layers has same effective receptive field as one 7×7 conv layer
- Deeper network means more non-linearities which leads to more capacity

Figure: [2]

VGGNet

Why use smaller filters? (3×3 conv)

- Stack of three 3×3 conv (stride 1) layers has same effective receptive field as one 7×7 conv layer
- Deeper network means more non-linearities which leads to more capacity
- Fewer parameters: $3 \times (3^2C^2)$ vs. 7^2C^2 for C channels per layer

Figure: [2]

Network in Network

Do you see any difference?

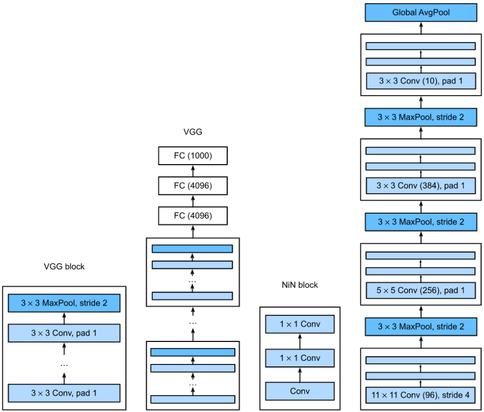


Figure: [3]

Network in Network

Do you see any difference?

- 1×1 Convolution
- Global Average Pooling(GAP) layer instead of FC layers

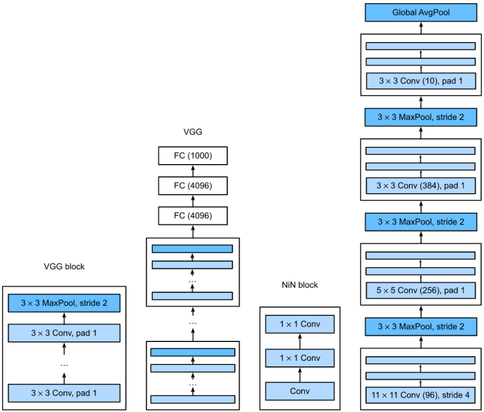


Figure: [3]

1×1 Convolution

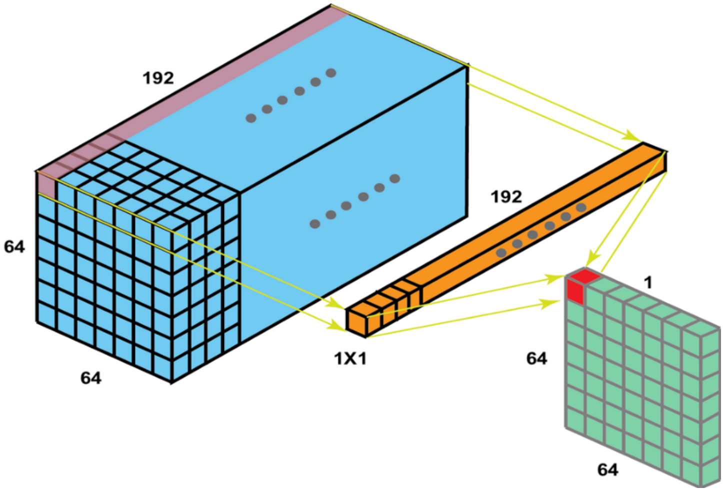


Figure: 1×1 Convolution

1×1 Conv Use Case

- Assume we want to transform a $32 \times 32 \times 200$ tensor to a $32 \times 32 \times 32$ one using 32 5×5 filters. Thus, we need $(32 \times 32 \times 200) \times (5 \times 5 \times 32) \approx 163M$ operations!

1×1 Conv Use Case

- Assume we want to transform a $32 \times 32 \times 200$ tensor to a $32 \times 32 \times 32$ one using 32 5×5 filters. Thus, we need $(32 \times 32 \times 200) \times (5 \times 5 \times 32) \approx 163M$ operations!
- Instead we can use 1×1 convolution:

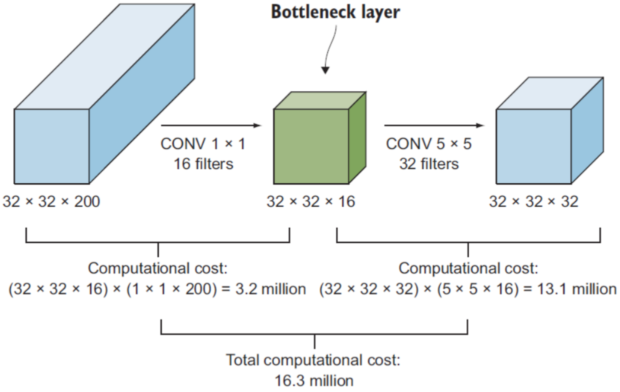


Figure: [1]

Global Average Pooling

Similar to max pooling layers, GAP layers are used to reduce the spatial dimensions of a three-dimensional tensor. However, GAP layers perform a more extreme type of dimensionality reduction, where a tensor with dimensions $h \times w \times d$ is reduced in size to have dimensions $1 \times 1 \times d$. GAP layers reduce each $h \times w$ feature map to a single number by simply taking the average of all hw values.

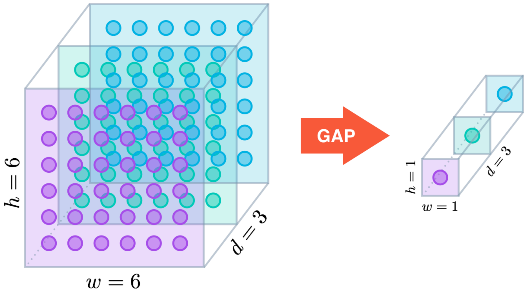


Figure: GAP

Why GAP?

- GAP is used to replace the traditional fully connected layers in CNN.
- There is no parameter to optimize in the GAP thus overfitting is avoided at this layer
- GAP sums out the spatial information, thus it is more robust to spatial translations of the input.

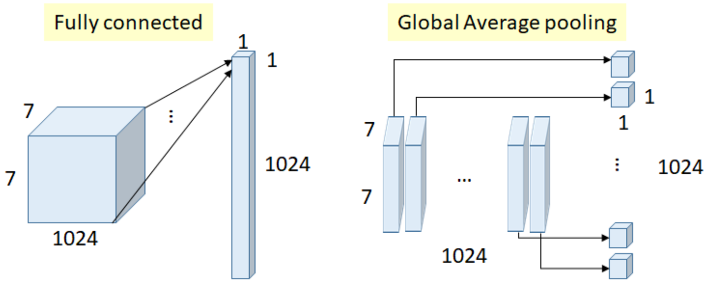


Figure: FC Layer vs GAP Layer

GoogLeNet

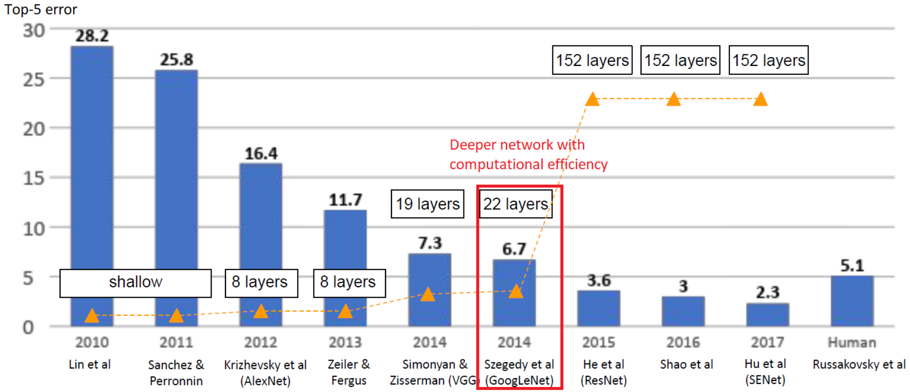


Figure: ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

GoogLeNet

- Only 5 million parameters! (12x less than AlexNet and 27x less than VGG-16)
- Efficient "Inception" module
- No longer multiple expensive FC layers

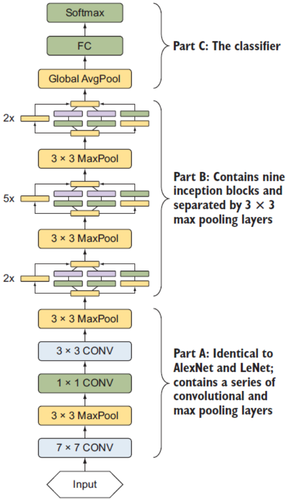


Figure: [1]

GoogLeNet

Inception modules instead of classical CNNs for feature extraction

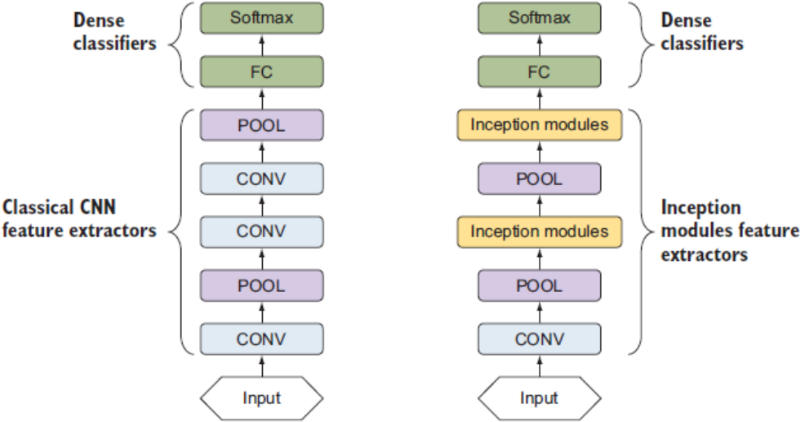


Figure: [1]

Naive Inception Module

- Apply parallel filter operations on the input from previous layer
- Multiple receptive field sizes for convolution (1×1, 3×3, 5×5)
- Pooling operation (3×3)
- Concatenate all filter outputs together channel-wise

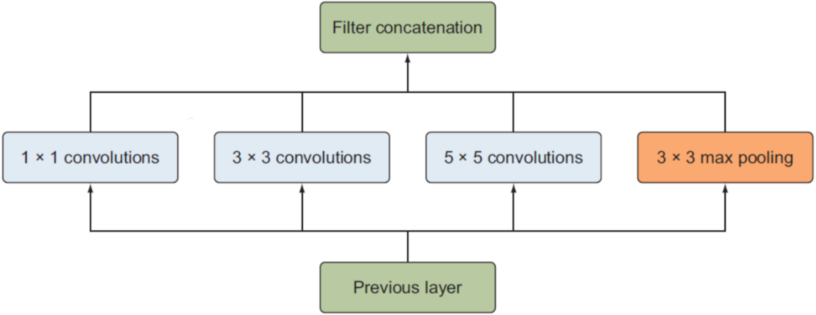


Figure: [1]

Naive Inception Module

- What are the output sizes of all different filter operations?

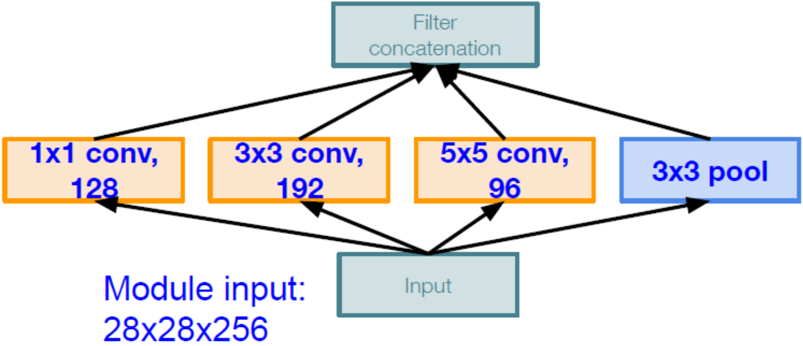


Figure: [2]

Naive Inception Module

- What are the output sizes of all different filter operations?

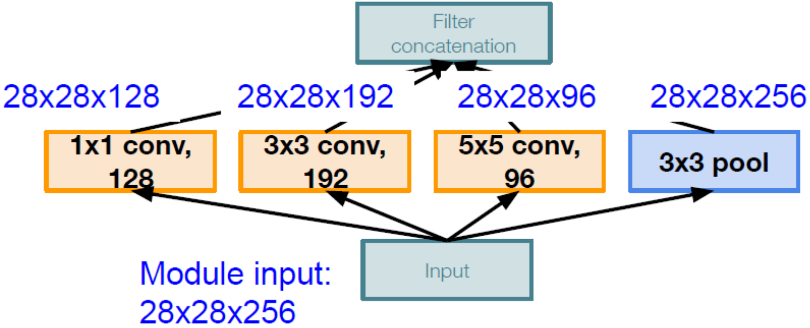


Figure: [2]

Naive Inception Module

- What is output size after filter concatenation?

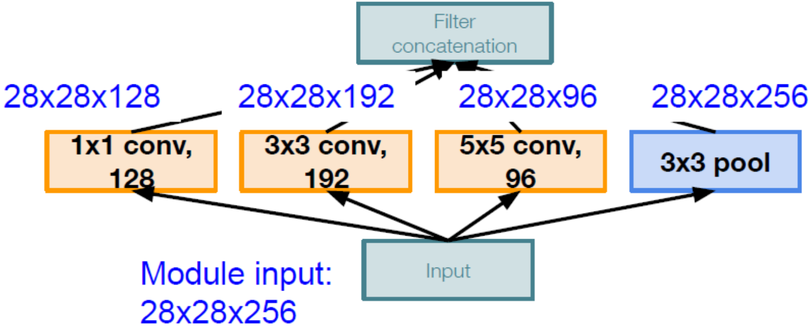


Figure: [2]

Naive Inception Module

- What is output size after filter concatenation?

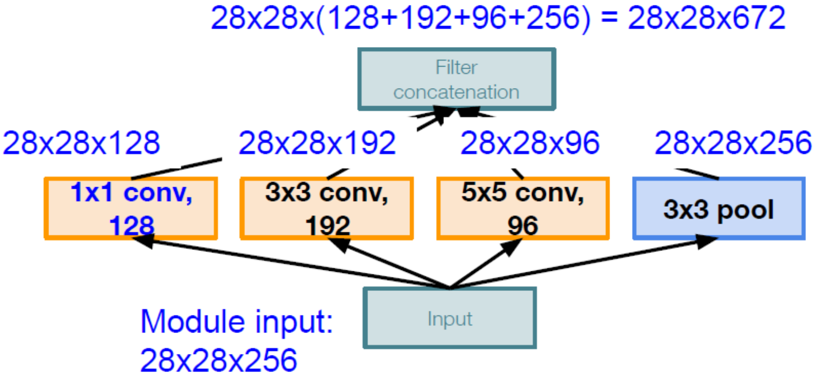


Figure: [2]

Naive Inception Module

- What is the problem with this?

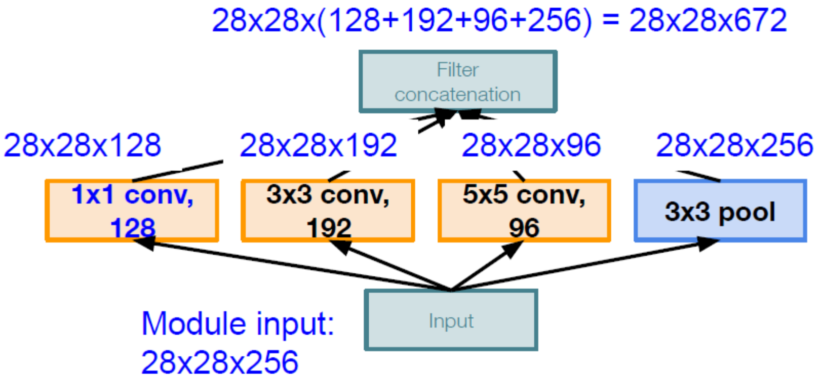


Figure: [2]

Naive Inception Module

- What is the problem with this?
- Computational complexity!
  - Conv Ops:
  - $[1 \times 1 \text{ conv}, 128]$ $28 \times 28 \times 128 \times 1 \times 1 \times 256$
  - $[3 \times 3 \text{ conv}, 192]$ $28 \times 28 \times 192 \times 3 \times 3 \times 256$
  - $[5 \times 5 \text{ conv}, 96]$ $28 \times 28 \times 96 \times 5 \times 5 \times 256$
  - Total: 854M ops
- Pooling layer preserves feature depth, which means total depth after concatenation can only grow at every layer!

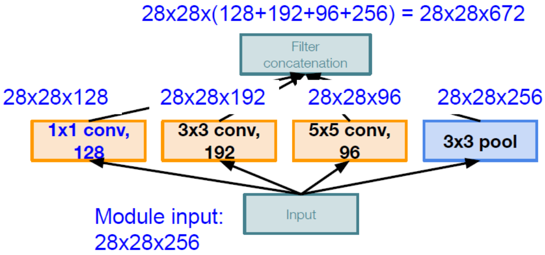


Figure: [2]

Inception Module

- Any Solution?

Inception Module

- Any Solution?
- "bottleneck" layers that use 1×1 convolutions to reduce feature channel size

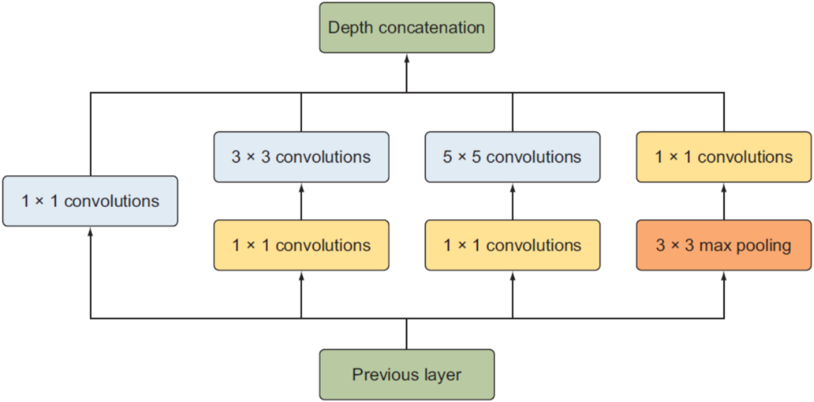


Figure: [1]

Dimension Reduction in Inception Module

- Using same parallel layers as naive example, and adding "1×1 conv, 64 filter" bottlenecks
  - $[1 \times 1 \text{ conv}, 64]\ 28 \times 28 \times 64 \times 1 \times 1 \times 256$
  - $[1 \times 1 \text{ conv}, 64]\ 28 \times 28 \times 64 \times 1 \times 1 \times 256$
  - $[1 \times 1 \text{ conv}, 128]\ 28 \times 28 \times 128 \times 1 \times 1 \times 256$
  - $[3 \times 3 \text{ conv}, 192]\ 28 \times 28 \times 192 \times 3 \times 3 \times 64$
  - $[5 \times 5 \text{ conv}, 96]\ 28 \times 28 \times 96 \times 5 \times 5 \times 64$
  - $[1 \times 1 \text{ conv}, 64]\ 28 \times 28 \times 64 \times 1 \times 1 \times 256$
  - Total: 358M ops
- Compared to 854M ops for naive version Bottleneck can also reduce depth after pooling layer

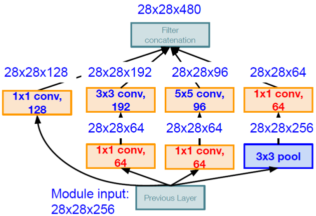


Figure: [2]

ResNet

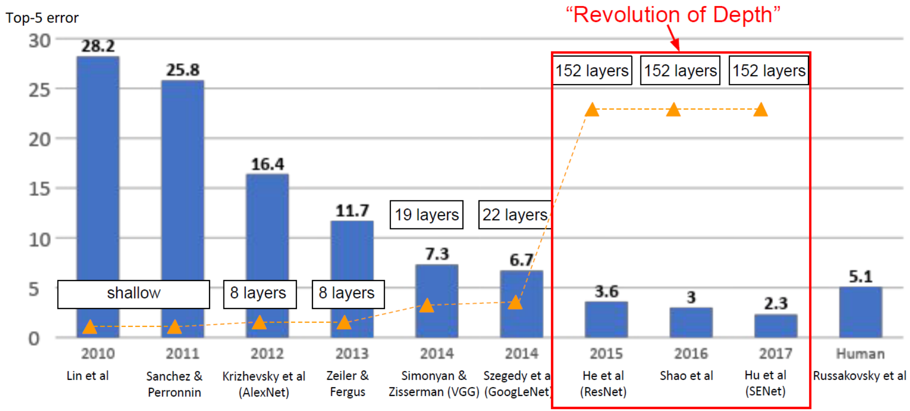


Figure: ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners

ResNet

- A very deep network using residual connections
- What happens when we continue stacking deeper layers on a "plain" convolutional

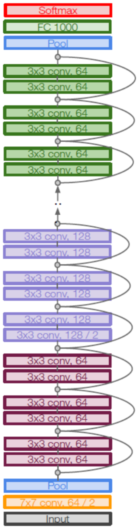


Figure: [2]

ResNet

- A very deep network using residual connections
- What happens when we continue stacking deeper layers on a "plain" convolutional

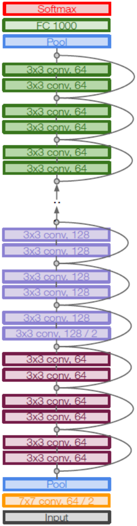
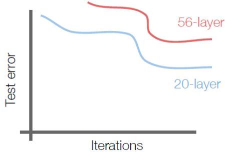
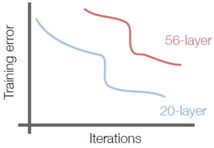


Figure: [2]

ResNet

- A very deep network using residual connections
- What happens when we continue stacking deeper layers on a "plain" convolutional





- The deeper model performs worse, but it's not caused by overfitting!

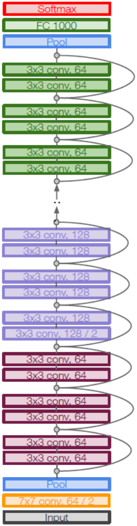


Figure: [2]

ResNet

- Fact: Deep models have more representation power (more parameters) than shallower models.
- Hypothesis: the problem is an optimization problem, deeper models are harder to optimize

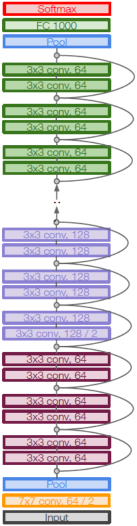


Figure: [2]

ResNet

- Fact: Deep models have more representation power (more parameters) than shallower models.
- Hypothesis: the problem is an optimization problem, deeper models are harder to optimize
- What should the deeper model learn to be at least as good as the shallower model?

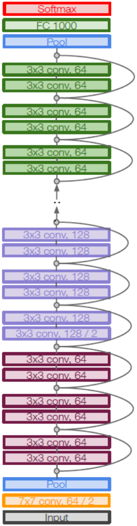


Figure: [2]

Skip Connection

Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping

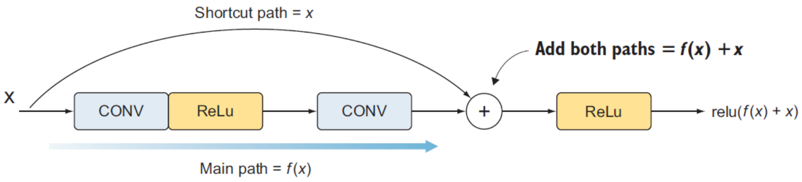


Figure: [1]

Full ResNet Architecture

- Stack residual blocks
- Every residual block has two 3×3 conv layers
- Periodically, double size of filters and downsample spatially using stride 2 (/2 in each dimension)
- Additional conv layer at the beginning (stem)
- No FC layers besides FC 1000 to output classes
- Global average pooling layer after last conv layer
- Batch Normalization after every CONV layer
- No dropout used

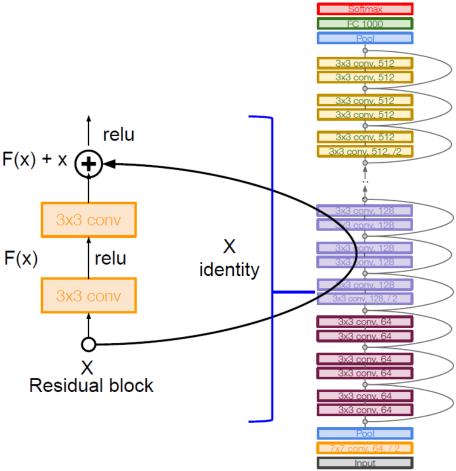


Figure: [2]

ResNeXt

- Increases width of residual block through multiple parallel pathways (similar to Inception module)
- Using g pathways for computational efficiency (why?)
- What is the purpose of the last 1×1 CONV layer?

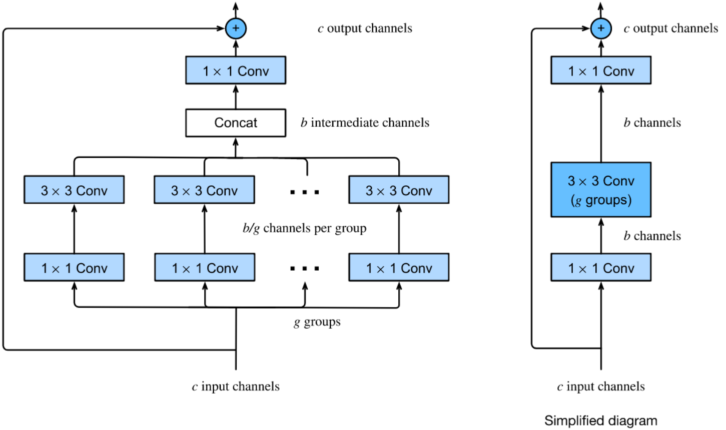


Figure: [3]

Final Notes

Thank You!

Any Question?

Refrences

Mohamed Elgendy.
Deep learning for vision systems.
Manning Publications, 2020.

Ruohan Gao Fei-Fei Li, Jiajun Wu.
Cnn architectures.
CS231n: Deep Learning for Computer Vision, 2022.

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola.
Dive into deep learning.
arXiv preprint arXiv:2106.11342, 2021.