Deep Learning for Computer Vision

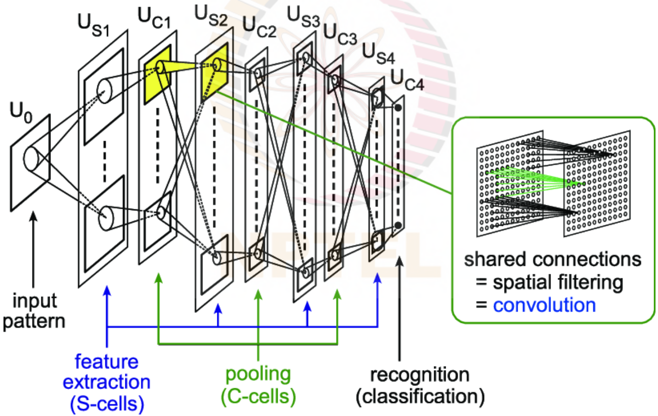
Evolution of CNN Architectures for Image Classification

Vineeth N Balasubramanian

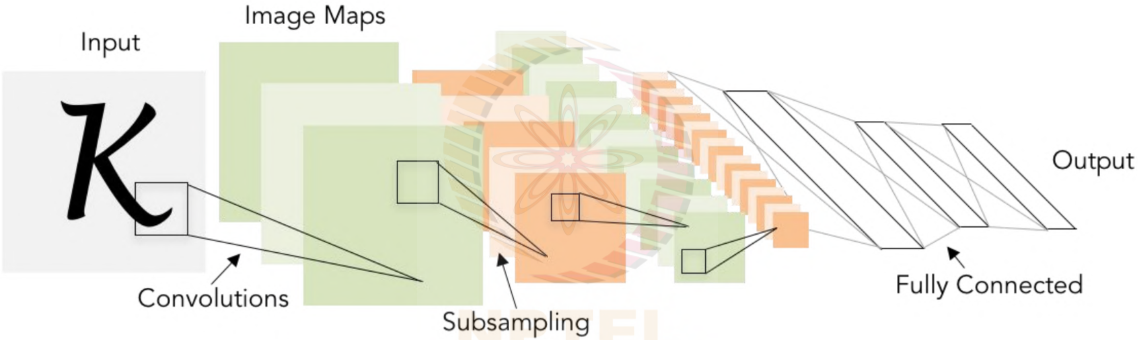
Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

History of CNNs

Neocognitron, 1980



LeNet-5 (1989-1998)

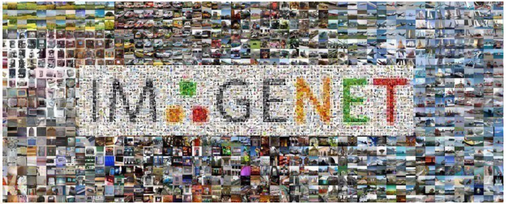


- Conv filters were 5×5, applied at stride 1
- Subsampling (Pooling) layers were 2×2 applied at stride 2
- **Overall Architecture: [CONV-POOL-CONV-POOL-FC-FC]**

Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

ImageNet Classification Challenge

- Image database organized according to WordNet hierarchy (currently only nouns)
- Currently, over five hundred images per node
- Started the ImageNet LSVRC in 2010, for benchmarking of methods for image classification
- Performance measure in Top-1 error and Top-5 error
- http://www.image-net.org/



Loss Functions: Beyond Mean Square Error

- **Cross-Entropy Loss Function:** Most popular for classification
- Given by:

$$
\begin{aligned}
L &= -\frac{1}{C}\sum_{i=1}^{C} y_i \log \hat{y}_i \\
&= -y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i) \text{(binary case)}
\end{aligned}
$$

- When activation function is sigmoid $\left(\sigma(x) = \frac{1}{1+e^{-x}}\right)$, derivative of cross-entropy loss function, $\frac{\partial L}{\partial w_j}$, w.r.t. a weight in last layer, w_j, is:

Loss Functions: Beyond Mean Square Error

- **Cross-Entropy Loss Function:** Most popular for classification
- Given by:

$$L = -\frac{1}{C}\sum_{i=1}^{C} y_i \log \hat{y}_i$$
$$= -y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i) \text{(binary case)}$$

- When activation function is sigmoid $\left(\sigma(x) = \frac{1}{1+e^{-x}}\right)$, derivative of cross-entropy loss function, $\frac{\partial L}{\partial w_j}$, w.r.t. a weight in last layer, w_j, is:

$$\frac{\partial L}{\partial w_j} = -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{1-y}{1-\sigma(z)}\right)\frac{\partial \sigma}{\partial w_j}$$
$$= -\frac{1}{n}\sum_x \left(\frac{y}{\sigma(z)} - \frac{1-y}{1-\sigma(z)}\right)\sigma'(z)x_j$$
$$= \frac{1}{n}\sum_x \frac{\sigma'(z)x_j}{\sigma(z)(1-\sigma(z))}(\sigma(z) - y)$$
$$= \frac{1}{n}\sum_x x_j\,(\sigma(z) - y)$$

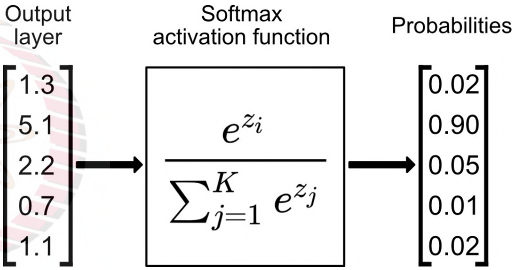
Note the last term in the final expression, very similar to gradient of MSE loss function

Activation Function in Output Layer

Softmax activation function

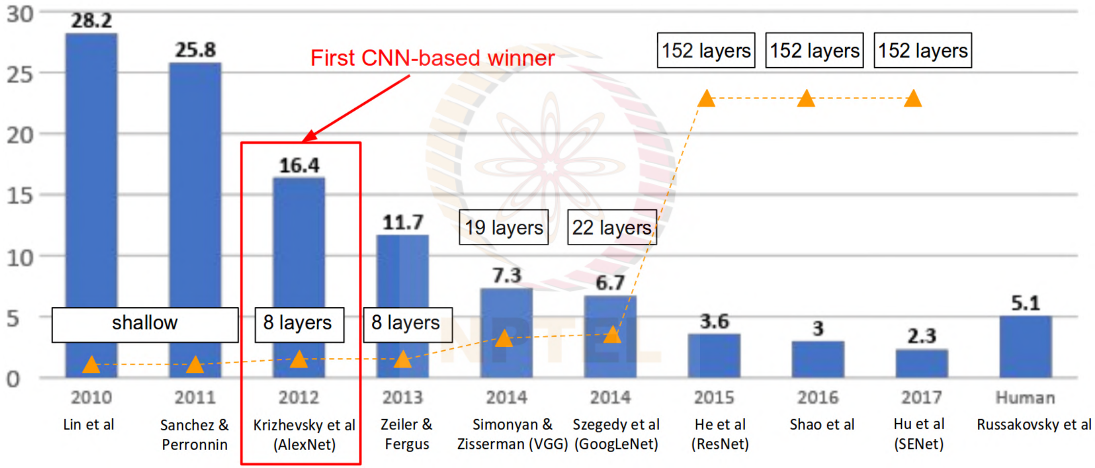
$$a_j = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

Helps convert output layer values (also called **logits**) to probability scores



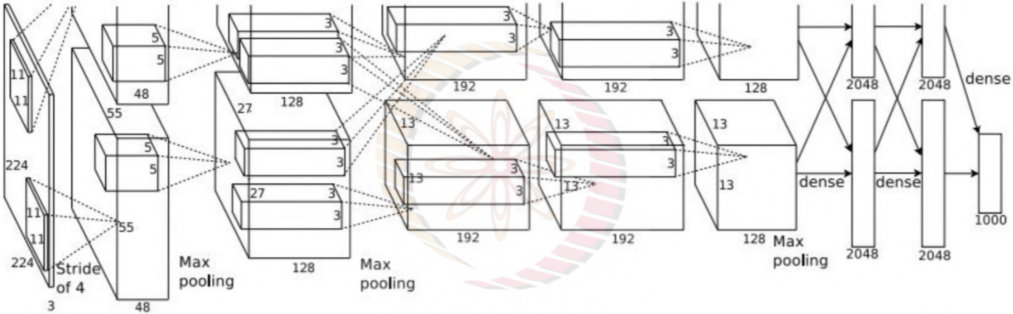
Credit: Dario Redicic, TowardsDataScience blog

Winners of ImageNet Classification Challenge



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

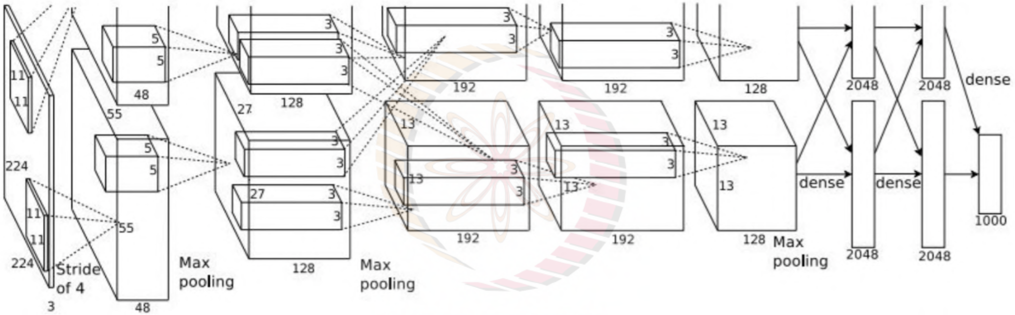
AlexNet[1]



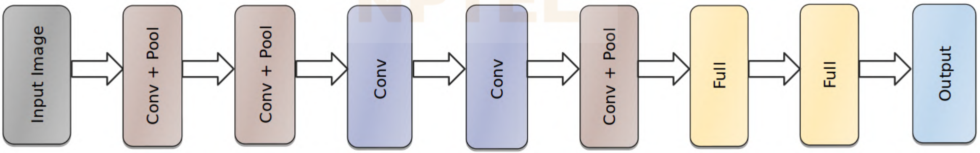
- Winner of ImageNet LSVRC-2012
- Overall architecture design similar to LeNet; but deeper with conv layers stacked on top of each other
- Trained over 1.2M images using SGD with regularization

[1]Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." NIPS 2012.

AlexNet



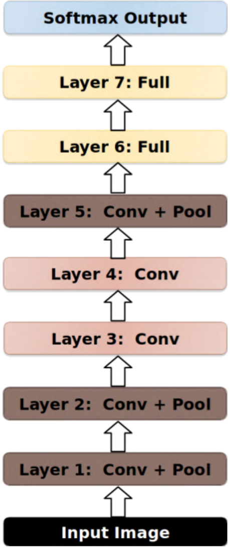
224
224
3
11
11
Stride of 4
55
55
48
48
5
Max pooling
27
27
128
128
3
Max pooling
13
192
192
128
Max pooling
dense
dense
dense
2048
2048
1000



Input Image
Conv + Pool
Conv + Pool
Conv
Conv
Conv + Pool
Full
Full
Output

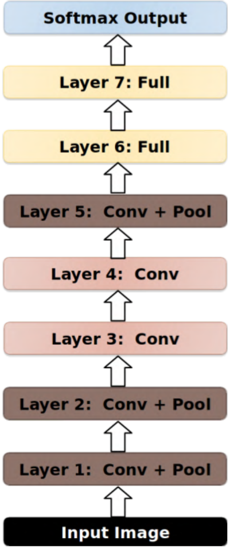
AlexNet

- 8 layers in total (5 convolutional layers, 3 fully connected layers)
- Trained on ImageNet Dataset



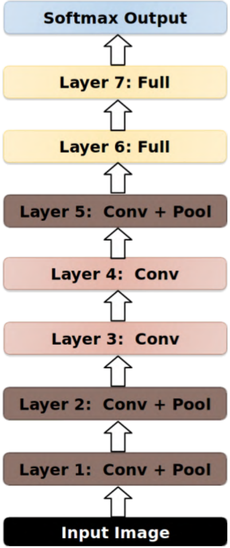
AlexNet

- 8 layers in total (5 convolutional layers, 3 fully connected layers)
- Trained on ImageNet Dataset
- **Response normalization layers** follow the first and second convolutional layers.



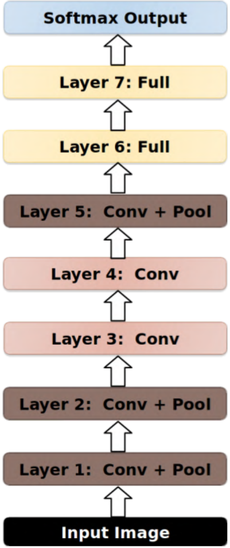
AlexNet

- 8 layers in total (5 convolutional layers, 3 fully connected layers)
- Trained on ImageNet Dataset
- **Response normalization layers** follow the first and second convolutional layers.
- Max-pooling follow first, second and the fifth convolutional layers

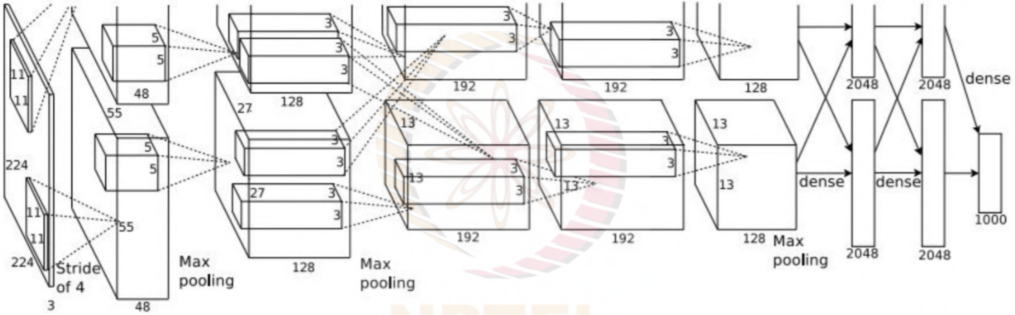


AlexNet

- 8 layers in total (5 convolutional layers, 3 fully connected layers)
- Trained on ImageNet Dataset
- **Response normalization layers** follow the first and second convolutional layers.
- Max-pooling follow first, second and the fifth convolutional layers
- The ReLU non-linearity is applied to the output of every layer



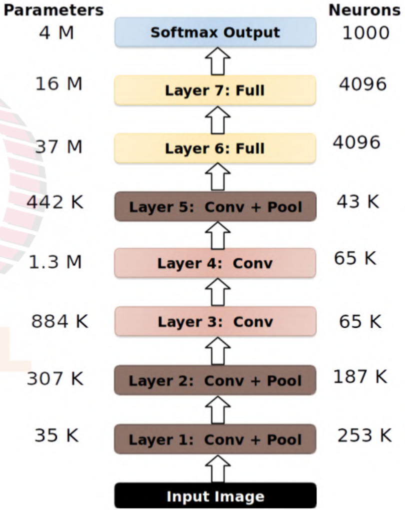
AlexNet



Parameters :	[(11 x 11 x 3) + 1] x 96 = 35 K	[5 x 5 x 48] x 256 = 307 K	[3 x 3 x 256] x 384 = 884 K	663 K	442 K	37 M	16 M	4 M	60 M
Neurons :	253,440	27 x 27 x 256 = 186,624	13 x 13 x 384 = 64,896	13 x 13 x 384 = 64,896	13 x 13 x 256 = 43,264	4096	4096	1000	0.63 M

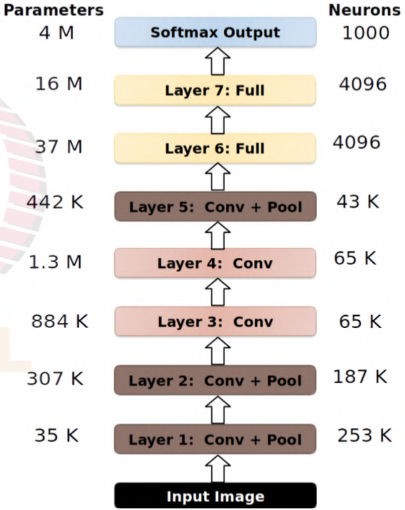
AlexNet

- Convolutional layers cumulatively contain about $90 - 95\%$ of computation, only about 5% of the parameters
- Fully-connected layers contain about 95% of parameters.

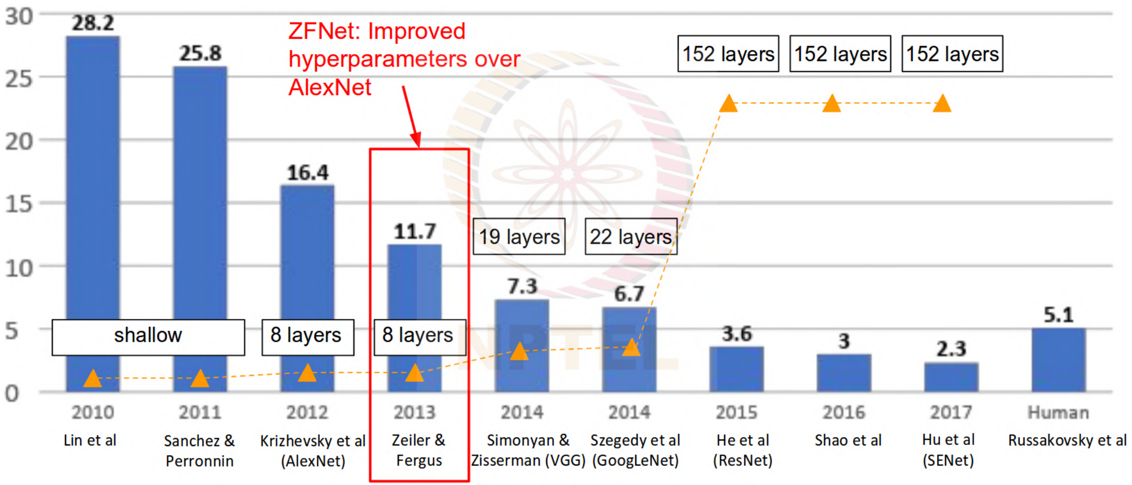


AlexNet

- Convolutional layers cumulatively contain about $90 - 95\%$ of computation, only about 5% of the parameters
- Fully-connected layers contain about 95% of parameters.
- Trained with SGD
  - on **two** NVIDIA GTX 580 3GB GPUs
  - for about a week

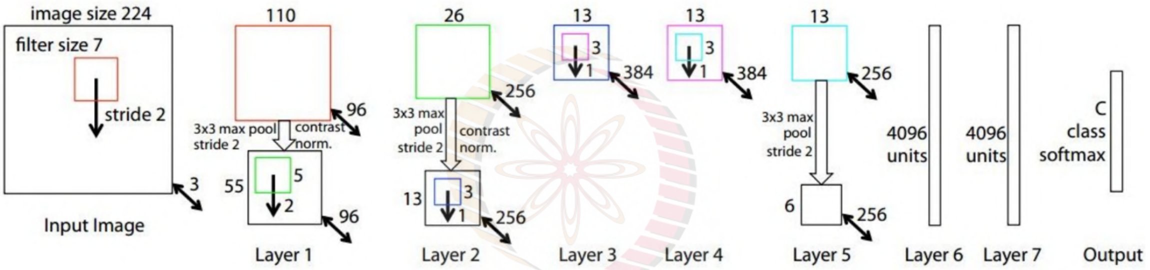


Winners of ImageNet Classification Challenge



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

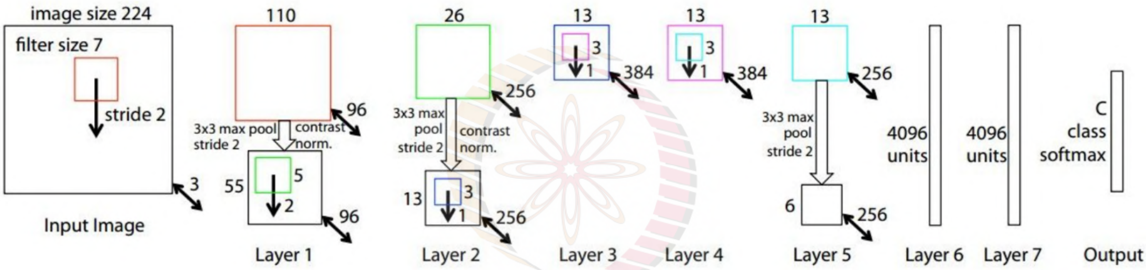
ZFNet (2013)[2]



- Similar to AlexNet but:

[2] Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

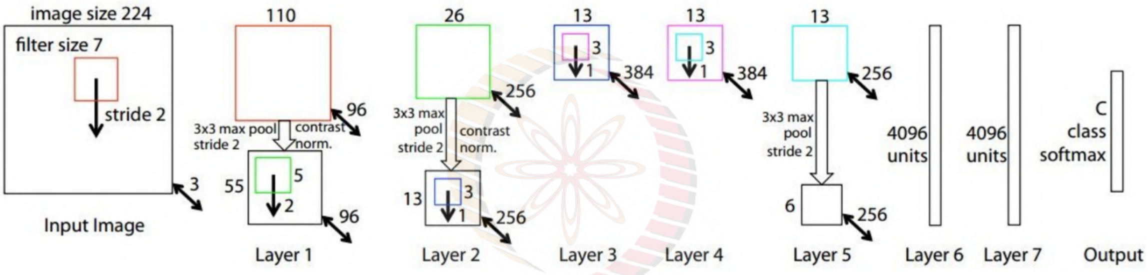
ZFNet (2013)[2]



- Similar to AlexNet but:

[2]Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

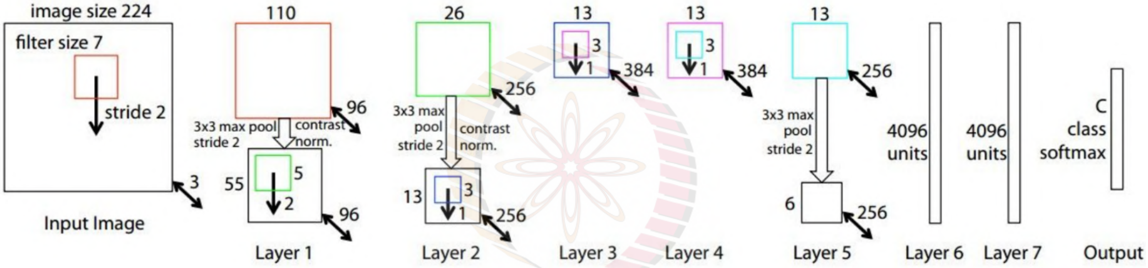
ZFNet (2013)[2]



- Similar to AlexNet but:
  - CONV1: change from (11×11 stride 4) to (7×7 stride 2)

[2]Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

ZFNet (2013)[2]

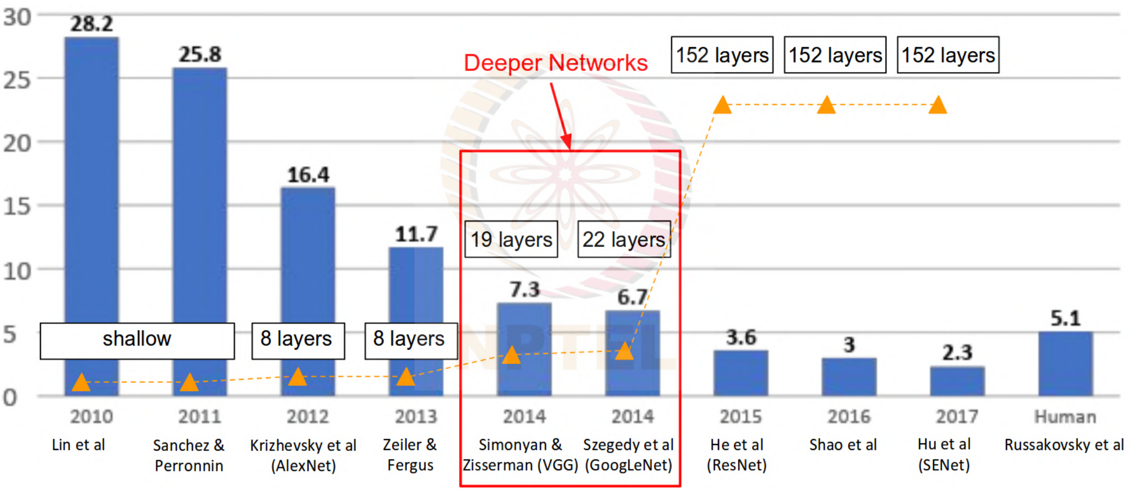


- Similar to AlexNet but:
  - CONV1: change from (11×11 stride 4) to (7×7 stride 2)
  - CONV3,4,5: instead of 384, 384, 256 filters, use 512, 1024, 512
- ImageNet top-5 error: $16.4\% \rightarrow 11.7\%$

Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

[2] Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

Winners of ImageNet Classification Challenge



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

VGGNet

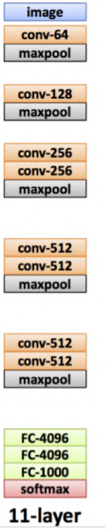


image
conv-64
maxpool
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
11-layer

VGGNet

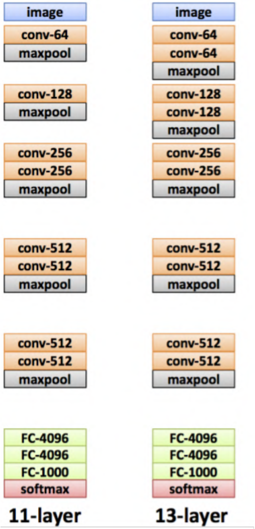


image
conv-64
maxpool
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
11-layer
image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
13-layer

VGGNet

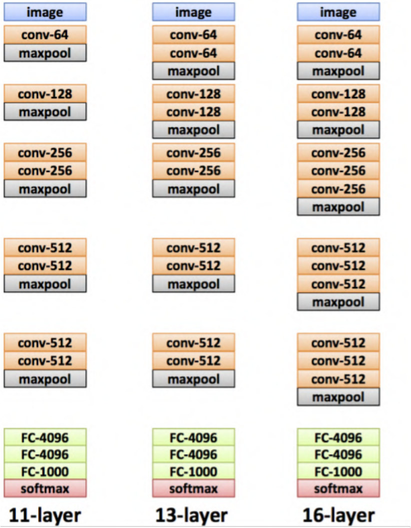


image
conv-64
maxpool
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
11-layer
image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
13-layer
image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
conv-256
maxpool
conv-512
conv-512
conv-512
maxpool
conv-512
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
16-layer

VGGNet

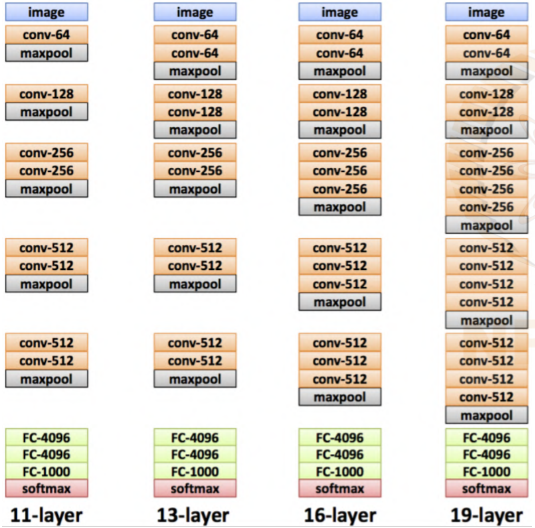
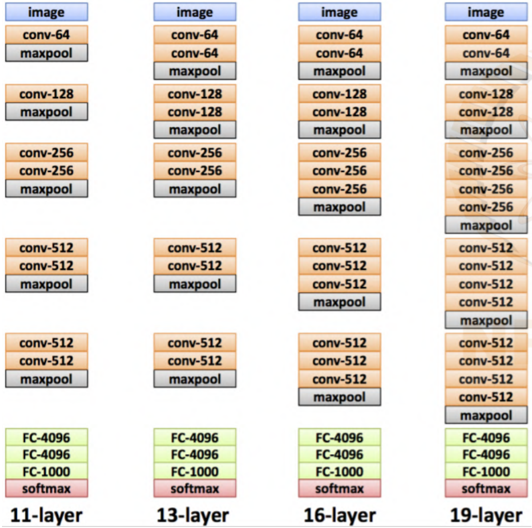


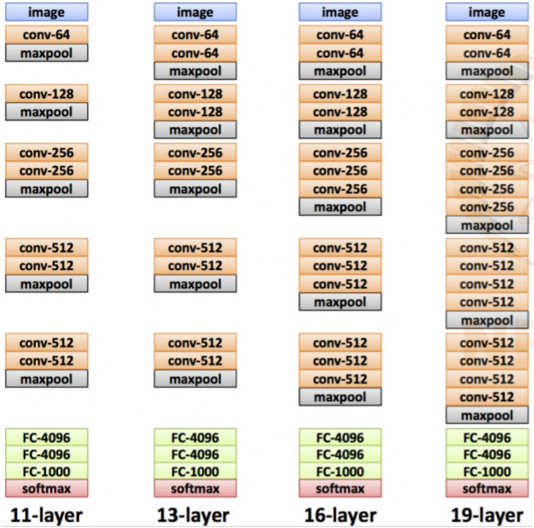
image
conv-64
maxpool
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
11-layer
image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
maxpool
conv-512
conv-512
maxpool
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
13-layer
image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
conv-256
maxpool
conv-512
conv-512
conv-512
maxpool
conv-512
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
16-layer
image
conv-64
conv-64
maxpool
conv-128
conv-128
maxpool
conv-256
conv-256
conv-256
conv-256
maxpool
conv-512
conv-512
conv-512
conv-512
maxpool
conv-512
conv-512
conv-512
conv-512
maxpool
FC-4096
FC-4096
FC-1000
softmax
19-layer

VGGNet



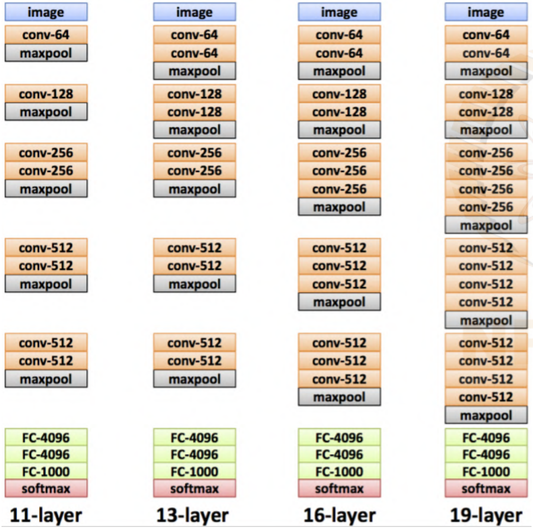
- Runner-up in ILSVRC 2014

VGGNet



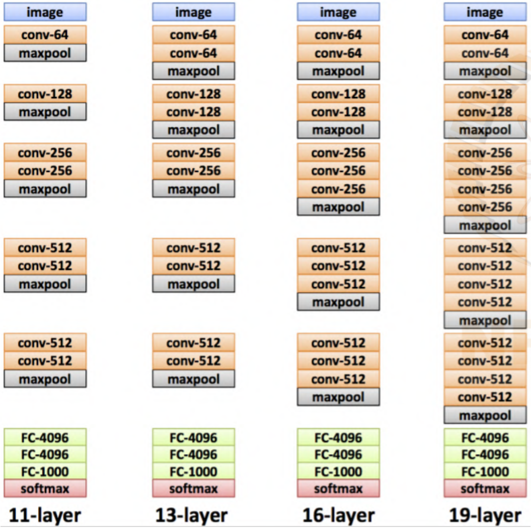
- Runner-up in ILSVRC 2014
- More layers lead to more nonlinearities

VGGNet



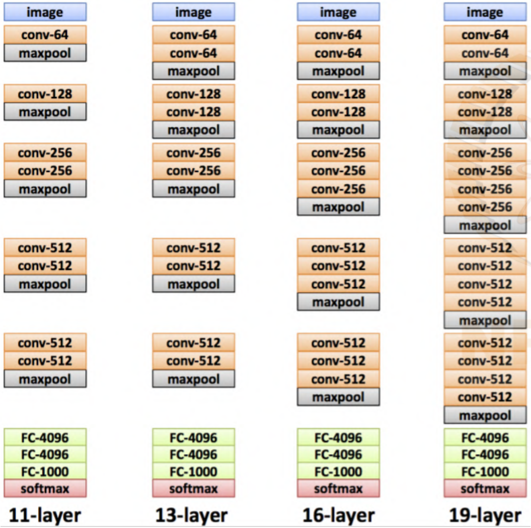
- Runner-up in ILSVRC 2014
- More layers lead to more nonlinearities
- Key contribution: Depth of the network is a critical component for good performance

VGGNet



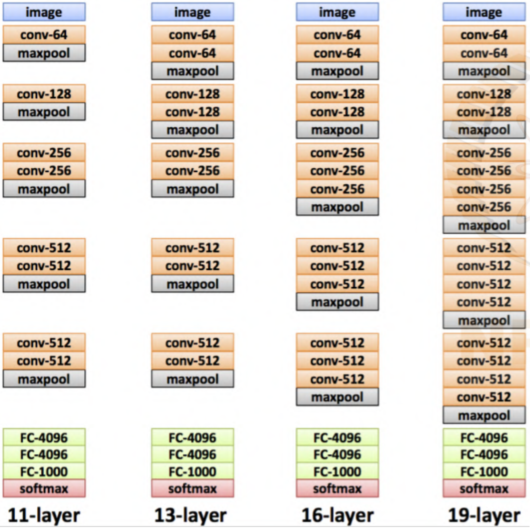
- Runner-up in ILSVRC 2014
- More layers lead to more nonlinearities
- Key contribution: Depth of the network is a critical component for good performance
- Homogeneous Architecture: From beginning to end:

VGGNet



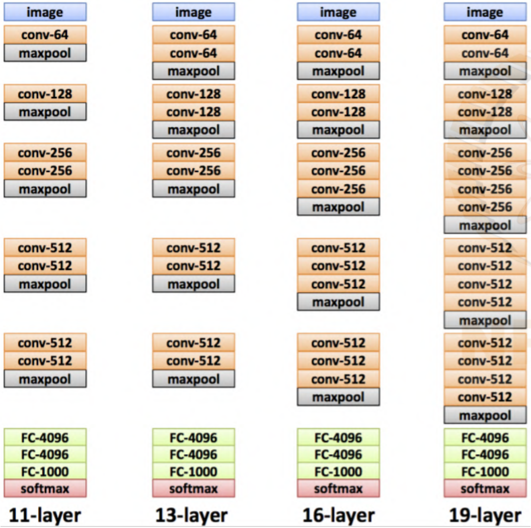
- Runner-up in ILSVRC 2014
- More layers lead to more nonlinearities
- Key contribution: Depth of the network is a critical component for good performance
- Homogeneous Architecture: From beginning to end:
  - 3×3 CONV stride 1 pad 1

VGGNet



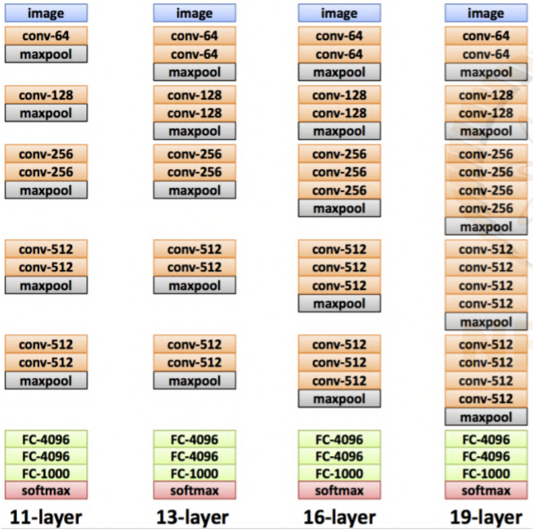
- Runner-up in ILSVRC 2014
- More layers lead to more nonlinearities
- Key contribution: Depth of the network is a critical component for good performance
- Homogeneous Architecture: From beginning to end:
  - 3×3 CONV stride 1 pad 1
  - 2×2 MAX POOL stride 2

VGGNet



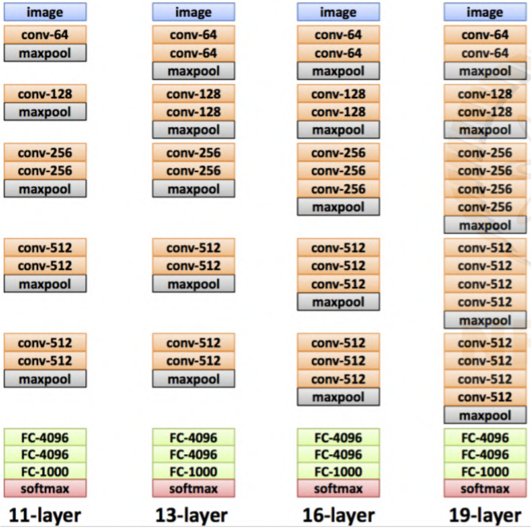
- Runner-up in ILSVRC 2014
- More layers lead to more nonlinearities
- Key contribution: Depth of the network is a critical component for good performance
- Homogeneous Architecture: From beginning to end:
  - 3×3 CONV stride 1 pad 1
  - 2×2 MAX POOL stride 2
- Smaller receptive fields:

VGGNet



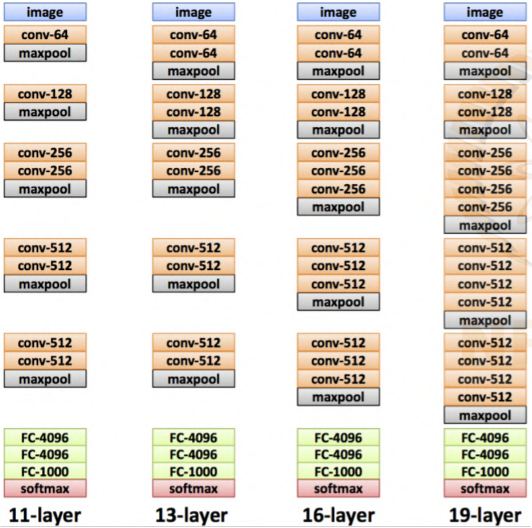
- Runner-up in ILSVRC 2014
- More layers lead to more nonlinearities
- Key contribution: Depth of the network is a critical component for good performance
- Homogeneous Architecture: From beginning to end:
  - 3×3 CONV stride 1 pad 1
  - 2×2 MAX POOL stride 2
- Smaller receptive fields:
  - less parameters; faster

VGGNet



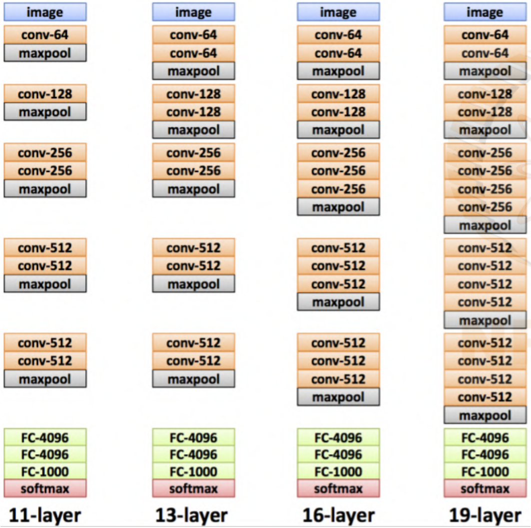
- Runner-up in ILSVRC 2014
- More layers lead to more nonlinearities
- Key contribution: Depth of the network is a critical component for good performance
- Homogeneous Architecture: From beginning to end:
  - 3 x 3 CONV stride 1 pad 1
  - 2 x 2 MAX POOL stride 2
- Smaller receptive fields:
  - less parameters; faster
  - two 3 x 3 conv has same receptive field as a single 5 x 5 conv; three 3 x 3 conv has same receptive field as a single 7 x 7 conv

VGGNet



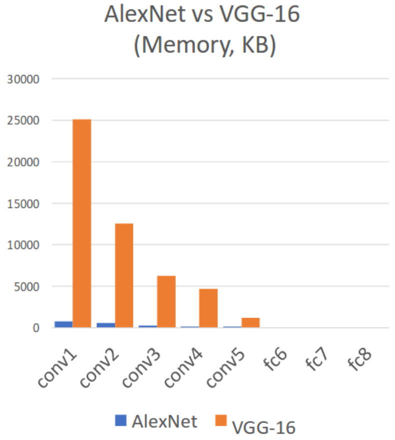
- Runner-up in ILSVRC 2014
- More layers lead to more nonlinearities
- Key contribution: Depth of the network is a critical component for good performance
- Homogeneous Architecture: From beginning to end:
  - 3×3 CONV stride 1 pad 1
  - 2×2 MAX POOL stride 2
- Smaller receptive fields:
  - less parameters; faster
  - two 3×3 conv has same receptive field as a single 5×5 conv; three 3×3 conv has same receptive field as a single 7×7 conv
  - Fewer parameters: $3 \times 3^2C^2$ (vs) 7^2C^2

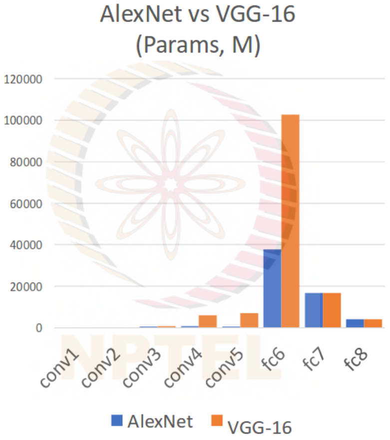
VGGNet



- VGG19 only slightly better than VGG16
- Used ensembles of networks for best results

VGGNet





AlexNet total: 1.9 MB

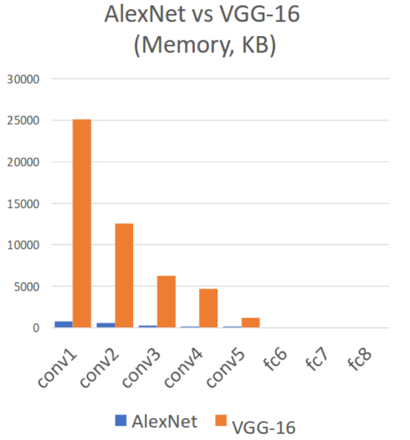
VGG-16 total: 48.6 MB (25x)

AlexNet total: 61M

VGG-16 total: 138M (2.3x)

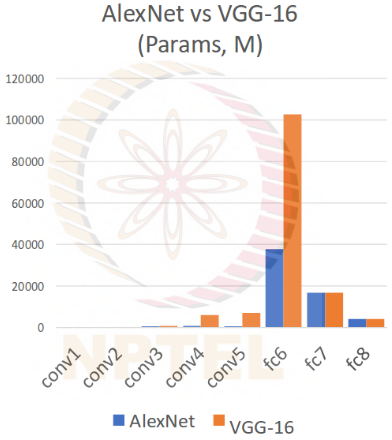
Credit: Justin Johnson, Univ of Michigan

VGGNet



AlexNet total: 1.9 MB

VGG-16 total: 48.6 MB (25x)



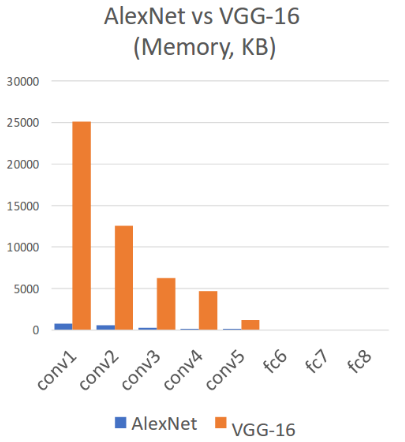
AlexNet total: 61M

VGG-16 total: 138M (2.3x)

- Uses a lot more memory and parameters

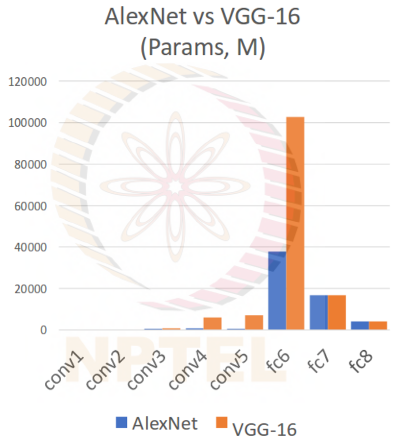
Credit: Justin Johnson, Univ of Michigan

VGGNet



AlexNet total: 1.9 MB

VGG-16 total: 48.6 MB (25x)



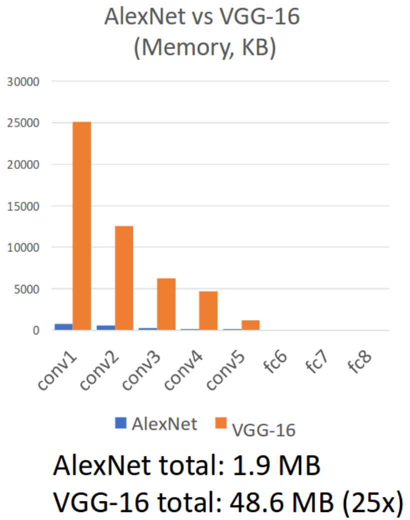
AlexNet total: 61M

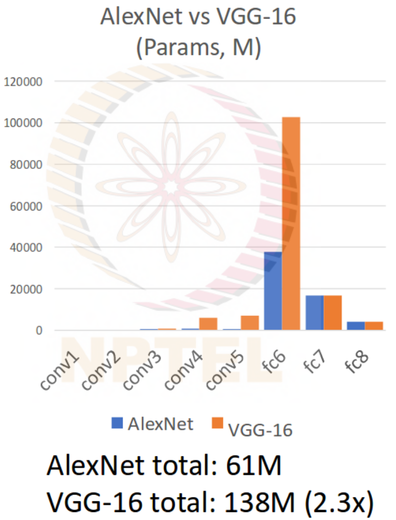
VGG-16 total: 138M (2.3x)

Credit: Justin Johnson, Univ of Michigan

- Uses a lot more memory and parameters
- Most of these parameters are in the first fully connected layer

VGGNet



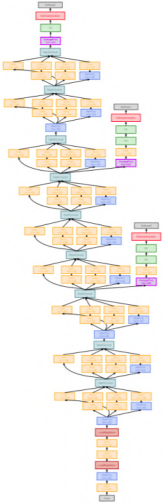


Credit: Justin Johnson, Univ of Michigan

- Uses a lot more memory and parameters
- Most of these parameters are in the first fully connected layer
- Most of the memory is used in early CONV layer

GoogleNet

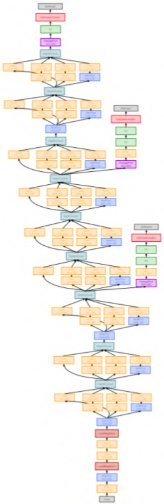
- **Deeper networks with focus on efficiency**: reduce parameter count, memory usage, and computation



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford; Justin Johnson, Univ of Michigan

GoogleNet

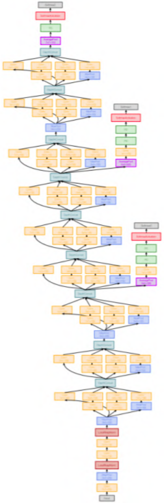
- **Deeper networks with focus on efficiency**: reduce parameter count, memory usage, and computation
- 22 layers



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford; Justin Johnson, Univ of Michigan

GoogleNet

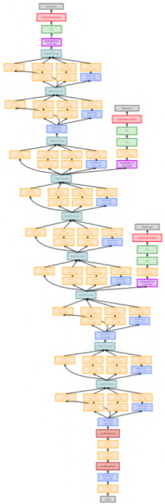
- **Deeper networks with focus on efficiency**: reduce parameter count, memory usage, and computation
- 22 layers
- No FC layers



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford; Justin Johnson, Univ of Michigan

GoogleNet

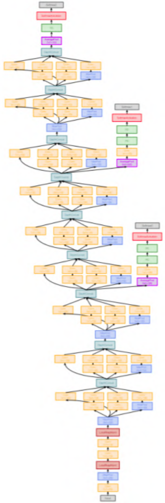
- **Deeper networks with focus on efficiency**: reduce parameter count, memory usage, and computation
- 22 layers
- No FC layers
- Efficient "**Inception**" module



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford; Justin Johnson, Univ of Michigan

GoogleNet

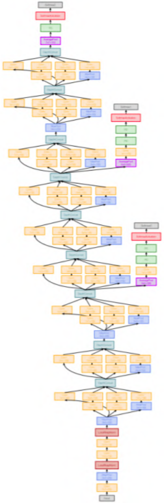
- **Deeper networks with focus on efficiency**: reduce parameter count, memory usage, and computation
- 22 layers
- No FC layers
- Efficient "**Inception**" module
- Only 5 million parameters! (12x less than AlexNet)



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford; Justin Johnson, Univ of Michigan

GoogleNet

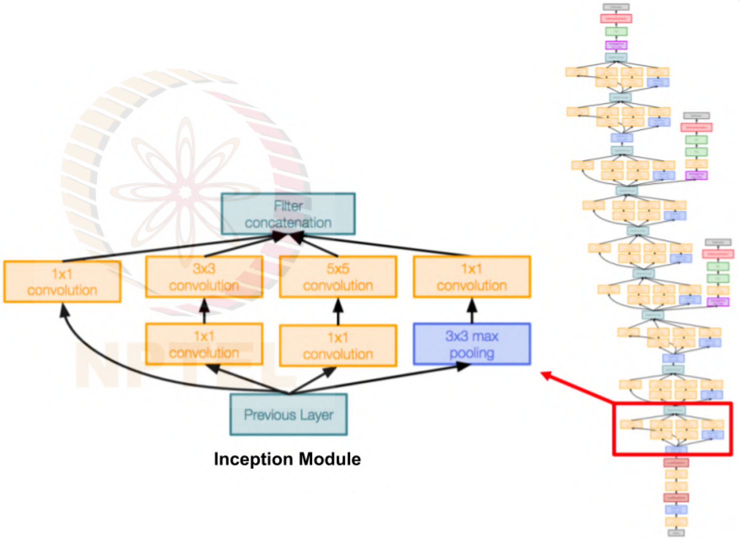
- **Deeper networks with focus on efficiency**: reduce parameter count, memory usage, and computation
- 22 layers
- No FC layers
- Efficient "**Inception**" module
- Only 5 million parameters! (12x less than AlexNet)
- **ILSVRC'14 classification winner** (6.7% top-5 error)



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford; Justin Johnson, Univ of Michigan

GoogleNet

- **Inception module:** Local unit with parallel branches
- Local structure repeated many times throughout the network

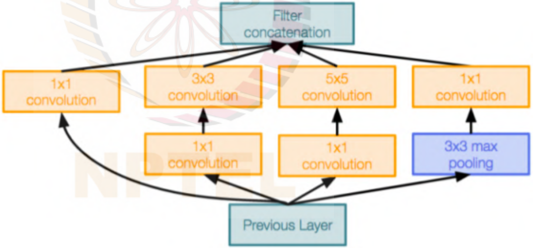


Inception Module

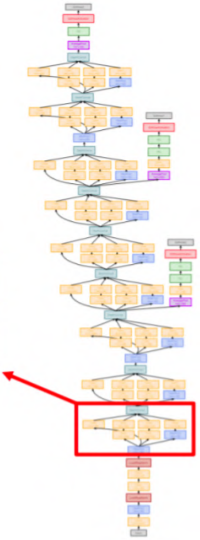
Credit: *Justin Johnson, Univ of Michigan*

GoogleNet

- **Inception module:** Local unit with parallel branches
- Local structure repeated many times throughout the network



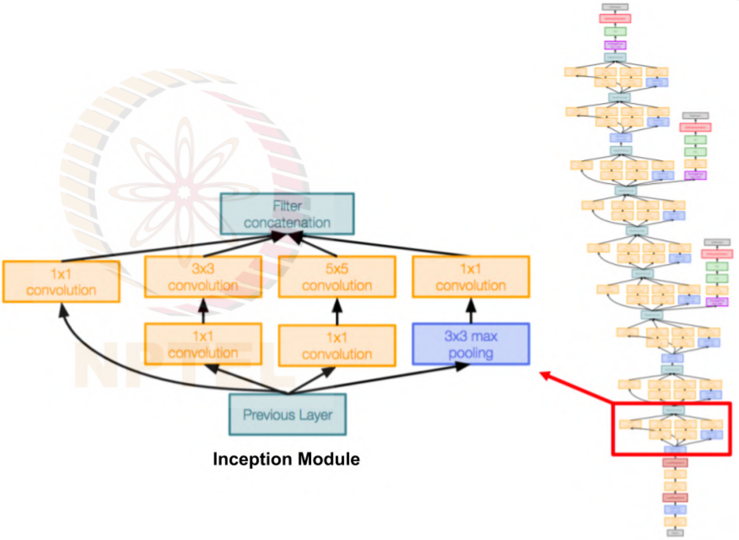
Inception Module



Credit: Justin Johnson, Univ of Michigan

GoogleNet

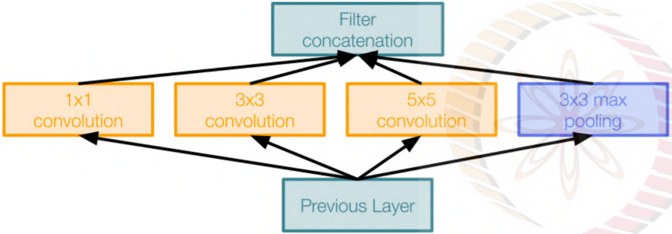
- **Inception module:** Local unit with parallel branches
- Local structure repeated many times throughout the network



Inception Module

Credit: Justin Johnson, Univ of Michigan

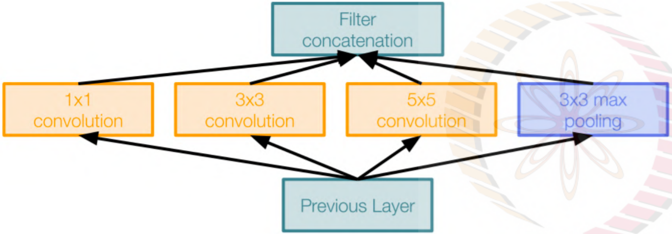
GoogleNet



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 max pooling)
- Concatenate all filter outputs together depth-wise

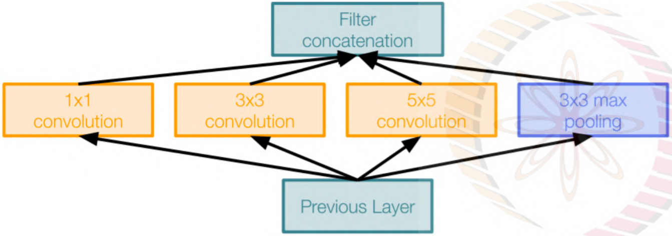
GoogleNet



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 max pooling)
- Concatenate all filter outputs together depth-wise
- **What's the problem with this?**

GoogleNet



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 max pooling)
- Concatenate all filter outputs together depth-wise
- **What's the problem with this?** Computationally very expensive

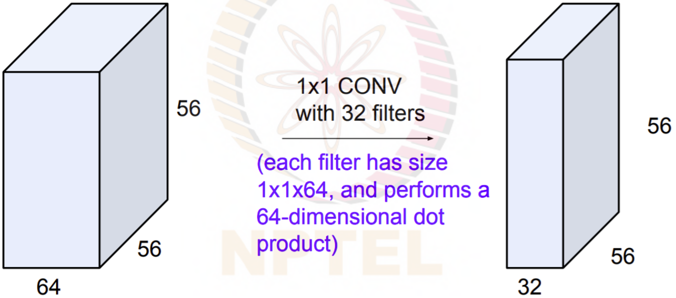
Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

GoogleNet

Solution: Use 1×1 "Bottleneck" layers to reduce channel dimension before expensive conv layers

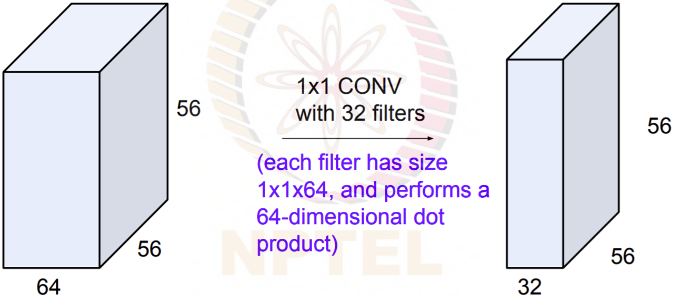
GoogleNet

Solution: Use 1×1 "Bottleneck" layers to reduce channel dimension before expensive conv layers



GoogleNet

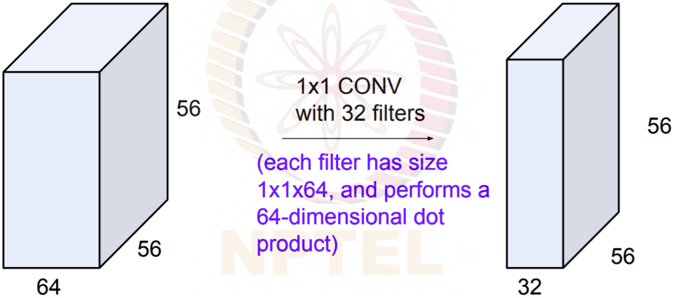
Solution: Use 1×1 "Bottleneck" layers to reduce channel dimension before expensive conv layers



- Preserves spatial dimensions, reduces depth!

GoogleNet

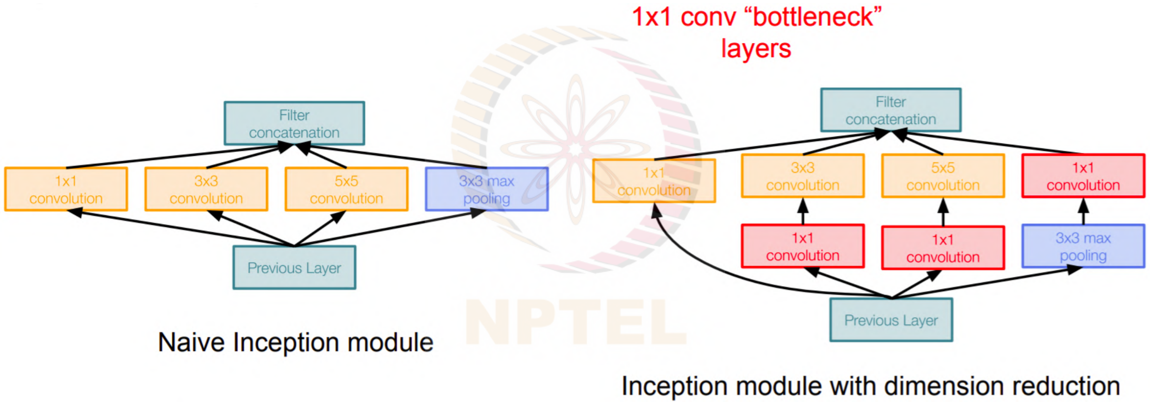
Solution: Use 1×1 "Bottleneck" layers to reduce channel dimension before expensive conv layers



- Preserves spatial dimensions, reduces depth!
- Projects depth to lower dimension (combination of feature maps)

Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

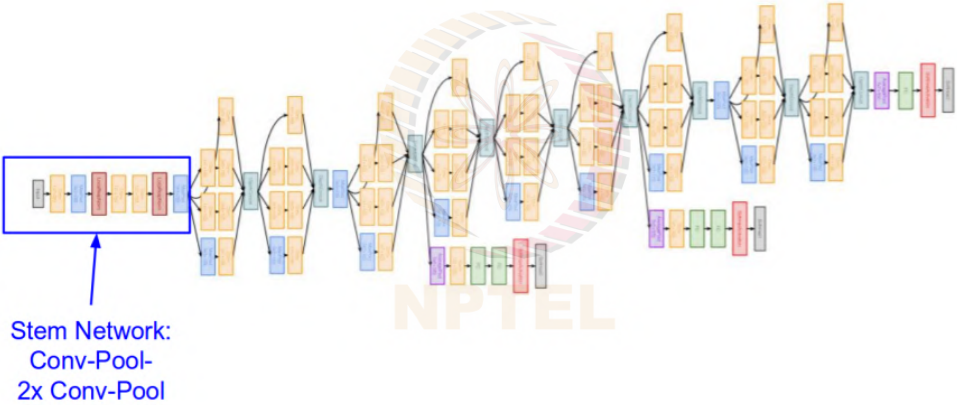
GoogleNet



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

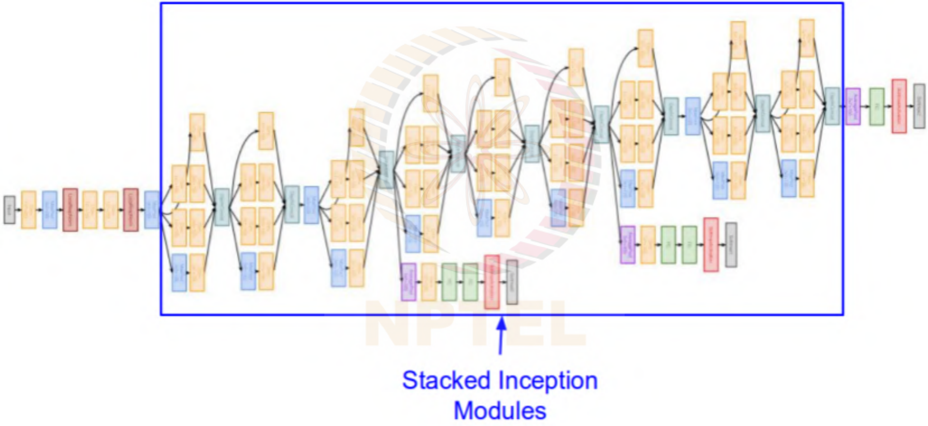
GoogleNet

Full Architecture:

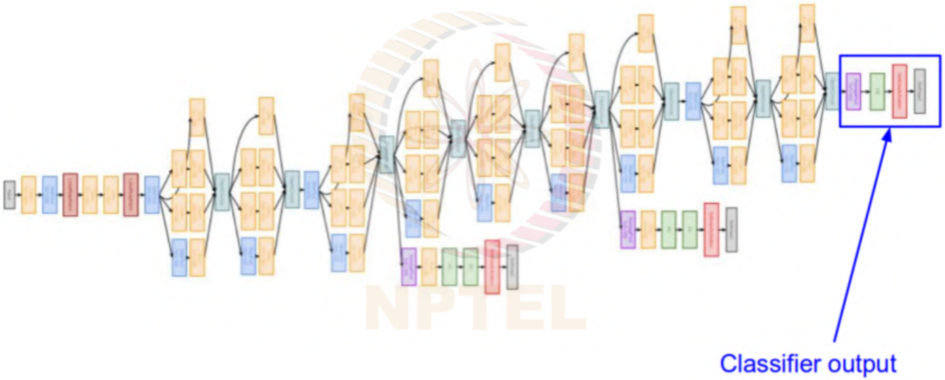


GoogleNet

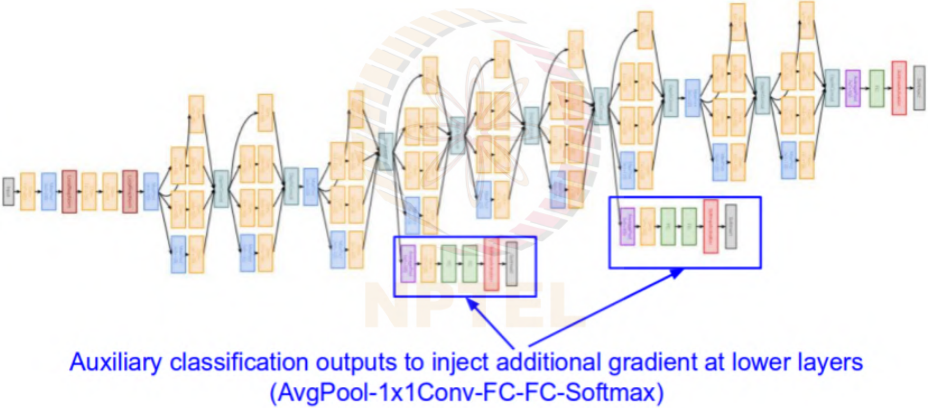
Full Architecture:



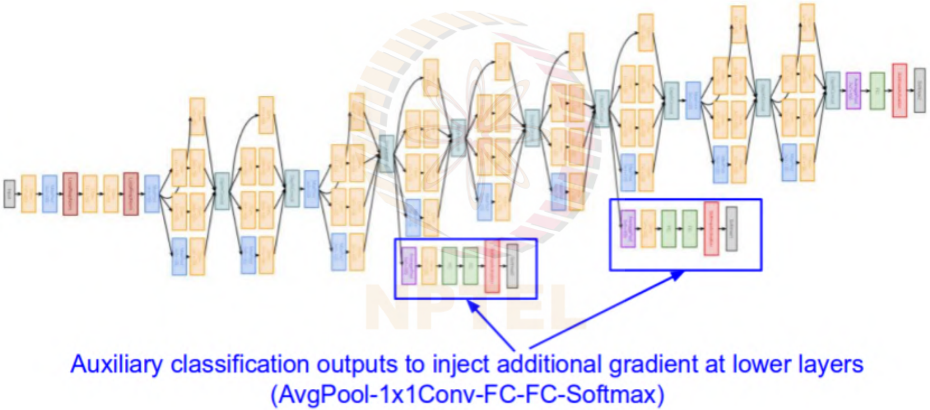
Full Architecture:



Full Architecture:



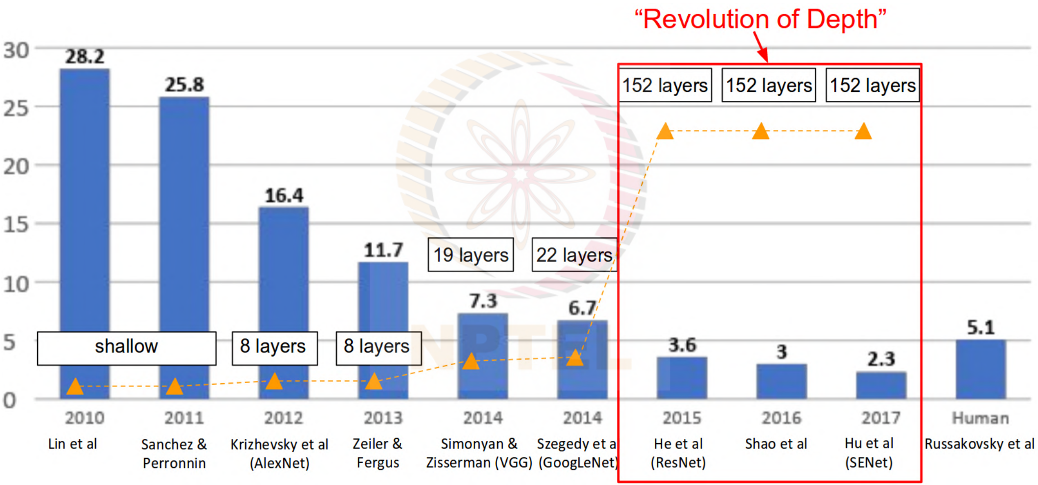
Full Architecture:



22 total layers (parallel layers count as 1 layer. Auxiliary output layers not counted)

Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

Deeper the Merrier



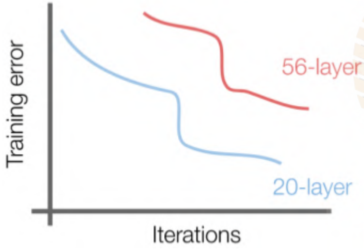
Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

How deep can we go?

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?

How deep can we go?

What happens when we continue stacking deeper layers on a "plain" convolutional neural network?



How deep can we go?

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?

How deep can we go?

What happens when we continue stacking deeper layers on a "plain" convolutional neural network?

Deeper model does worse than shallow model!

How deep can we go?

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?

Deeper model does worse than shallow model! **Why?**

How deep can we go?

What happens when we continue stacking deeper layers on a “plain” convolutional neural network?

Deeper model does worse than shallow model! **Why?**

The initial guess is that the deep model is **overfitting** since it is much bigger than the shallow model

How deep can we go?

What happens when we continue stacking deeper layers on a "plain" convolutional neural network?

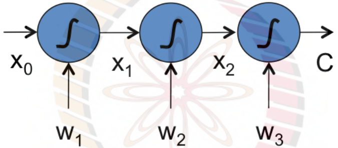
Deeper model does worse than shallow model! **Why?**

The deep model is actually **underfitting** since it also performs worse than the shallow model on the training set

Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford; Justin Johnson, Univ of Michigan

How deep can we go? Vanishing/Exploding Gradient

Consider a simple network:



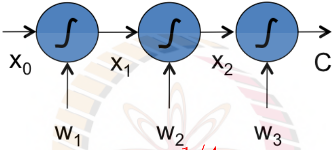
$$\frac{\partial L}{\partial x_0} = \underbrace{\boxed{\sigma'(w_3^T x_2)}}_{} \times w_3 \times \boxed{\sigma'(w_2^T x_1)} \times w_2 \times \boxed{\sigma'(w_1^T x_0)} \times w_1$$

$< 1/4$ $\quad$ $< 1/4$ $\quad$ $< 1/4$

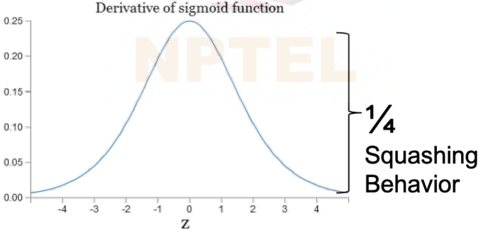
Why?

How deep can we go? Vanishing/Exploding Gradient

Consider a simple network:

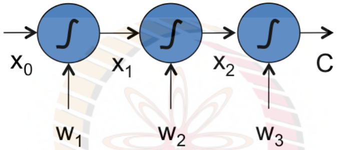


$$\frac{\partial L}{\partial x_0} = \underbrace{\sigma'(w_3^T x_2)}_{<1/4} \times w_3 \times \underbrace{\sigma'(w_2^T x_1)}_{<1/4} \times w_2 \times \underbrace{\sigma'(w_1^T x_0)}_{<1/4} \times w_1$$



How deep can we go? Vanishing/Exploding Gradient

Consider a simple network:



$$\frac{\partial L}{\partial x_0} = \underbrace{\boxed{\sigma'(w_3^T x_2)}}_{<1/4} \times w_3 \times \underbrace{\boxed{\sigma'(w_2^T x_1)}}_{<1/4} \times w_2 \times \underbrace{\boxed{\sigma'(w_1^T x_0)}}_{<1/4} \times w_1$$

- **Vanishing gradients:** Deeper the network, gradients vanish quickly, thereby slowing the rate of change in initial layers
- **Exploding gradients:** Happen when the individual layer gradients are much higher than 1, for instance - can be overcome by **gradient clipping**

ResNet

The deeper model should be able to perform at least as well as the shallower model;how?

ResNet

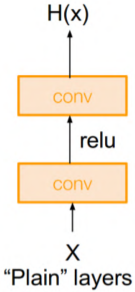
The deeper model should be able to perform at least as well as the shallower model;how?

Solution: Change the network with identity connections between layers:

ResNet

The deeper model should be able to perform at least as well as the shallower model;how?

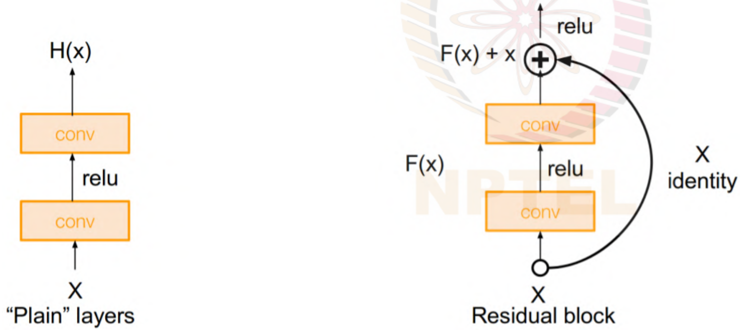
Solution: Change the network with identity connections between layers:



ResNet

The deeper model should be able to perform at least as well as the shallower model;how?

Solution: Change the network with identity connections between layers:

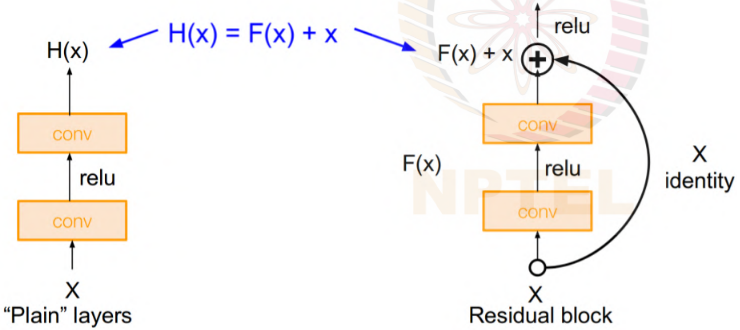


Use network layers to fit residual F(x) = H(x) - x instead of H(x) directly

Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

The deeper model should be able to perform at least as well as the shallower model;how?

Solution: Change the network with identity connections between layers:

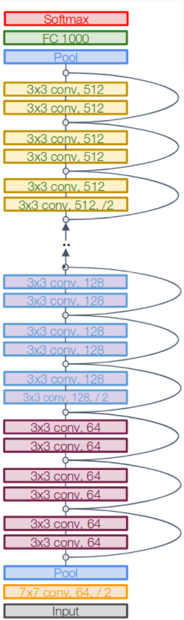


Use network layers to fit residual F(x) = H(x) - x instead of H(x) directly

Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

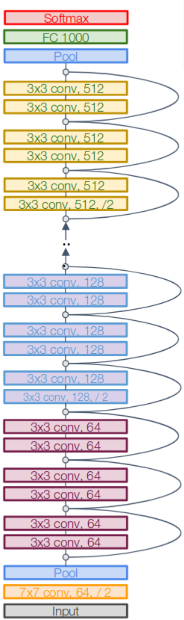
ResNet

- A residual network is a stack of many residual blocks
- Each residual block has two 3×3 conv layers

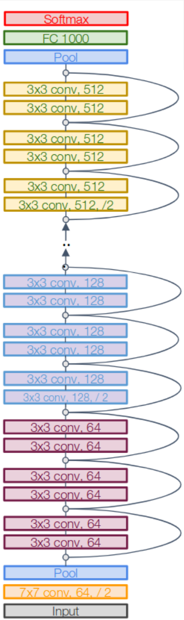


ResNet

- A residual network is a stack of many residual blocks
- Each residual block has two 3×3 conv layers
- Periodically, double number of filters and downsample spatially using stride 2 (/2 in each dimension)



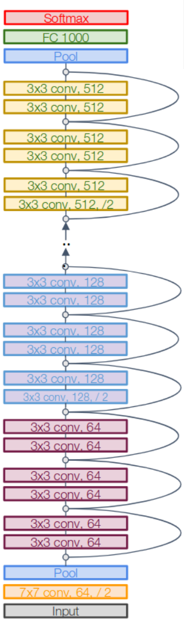
ResNet



- A residual network is a stack of many residual blocks
- Each residual block has two 3×3 conv layers
- Periodically, double number of filters and downsample spatially using stride 2 (/2 in each dimension)
- Use **global average pooling** and a single linear layer at the end (FC 1000 to output classes)

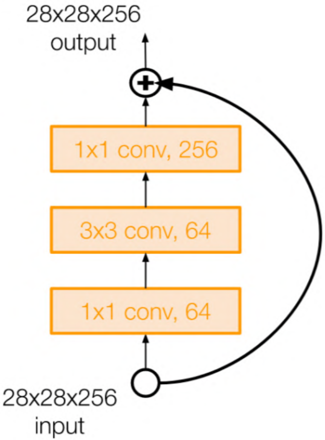
ResNet

- A residual network is a stack of many residual blocks
- Each residual block has two 3×3 conv layers
- Periodically, double number of filters and downsample spatially using stride 2 ($/2$ in each dimension)
- Use **global average pooling** and a single linear layer at the end (FC 1000 to output classes)
- Total depths of 34, 50, 101, or 152 layers for ImageNet dataset



ResNet

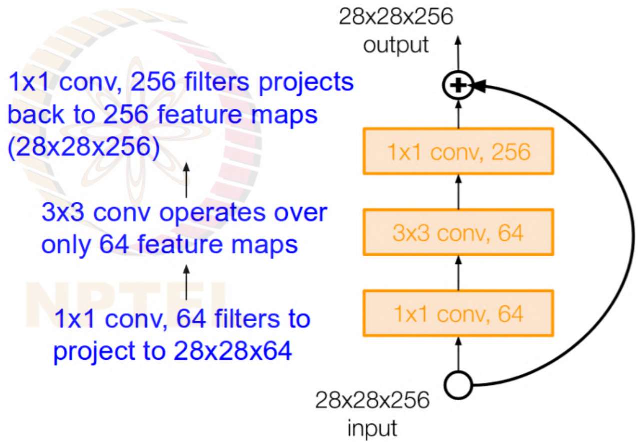
For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

ResNet

For deeper networks (ResNet-50+), use "bottleneck" layer to improve efficiency (similar to GoogLeNet)



Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

ResNet

- Able to train very deep networks
- Deeper networks performs better than shallow networks now (as expected); residual blocks help avoid vanishing gradient
- 1st place in all ILSVRC and COCO 2015 competitions
- We will discuss detection, localization, segmentation and the COCO dataset a bit later

ResNet

- Able to train very deep networks
- Deeper networks performs better than shallow networks now (as expected); residual blocks help avoid vanishing gradient
- 1st place in all ILSVRC and COCO 2015 competitions
- We will discuss detection, localization, segmentation and the COCO dataset a bit later

ResNet

- Able to train very deep networks
- Deeper networks performs better than shallow networks now (as expected); residual blocks help avoid vanishing gradient
- 1st place in all ILSVRC and COCO 2015 competitions
- We will discuss detection, localization, segmentation and the COCO dataset a bit later

ResNet

- Able to train very deep networks
- Deeper networks performs better than shallow networks now (as expected); residual blocks help avoid vanishing gradient
- 1st place in all ILSVRC and COCO 2015 competitions
- We will discuss detection, localization, segmentation and the COCO dataset a bit later

ResNet

- Able to train very deep networks
- Deeper networks performs better than shallow networks now (as expected); residual blocks help avoid vanishing gradient
- 1st place in all ILSVRC and COCO 2015 competitions
- We will discuss detection, localization, segmentation and the COCO dataset a bit later

ResNet @ ILSVRC & COCO 2015 Competitions

1st place in all five major challenges

- ImageNet Classification: "Ultra-deep" 152-layer nets
- ImageNet Detection: 16% better than the 2nd best
- ImageNet Localization: 27% better than the 2nd best
- COCO Detection: 11% better than the 2nd best
- COCO Segmentation: 12% better than the 2nd best

Credit: Fei-Fei Li, Justin Johnson and Serena Yeung, CS231n course, Stanford, Spring 2019

Homework

Readings

- Tutorial: Illustrated: 10 CNN Architectures
- (Optional) For more details, skim through the following papers:
  - ImageNet Classification with Deep Convolutional Neural Networks
  - Very Deep Convolutional Networks for Large-Scale Image Recognition
  - Going Deeper with Convolutions
  - Deep Residual Learning for Image Recognition

Exercise

- Show that minimizing negative log likelihood in a neural network with a softmax activation function in the last layer is equivalent to minimizing cross-entropy error function (*Hint:* Read Chapter 3 of Nielsen's online book on basics of NNs)

References

- Yann LeCun et al. "Gradient-based learning applied to document recognition". In: 1998.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS*. 2012.
- Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *CoRR* abs/1409.1556 (2015).
- Christian Szegedy et al. "Going deeper with convolutions". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 1–9.
- Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.
- *Johnson, Justin, EECS 498-007 / 598-005 - Deep Learning for Computer Vision (Fall 2019)*. URL: https://web.eecs.umich.edu/~justincj/teaching/eecs498/ (visited on 06/29/2020).
- *Li, Fei-Fei; Johnson, Justin; Serena, Yeung; CS 231n - Convolutional Neural Networks for Visual Recognition (Spring 2019)*. URL: http://cs231n.stanford.edu/2019/ (visited on 06/29/2020).