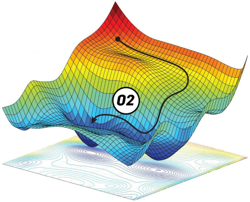
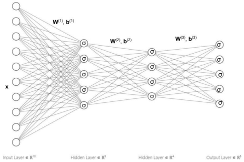
Deep Learning for Computer Vision

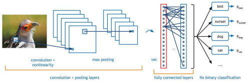
Dr. Konda Reddy Mopuri
Mehta Family School of Data Science and Artificial Intelligence
IIT Guwahati
Aug-Dec 2022

So far in the class..

- Scoring function, loss function, gradient descent
- Artificial Neurons and Multi-Layered Perceptron
- CNN building blocks and a case-study





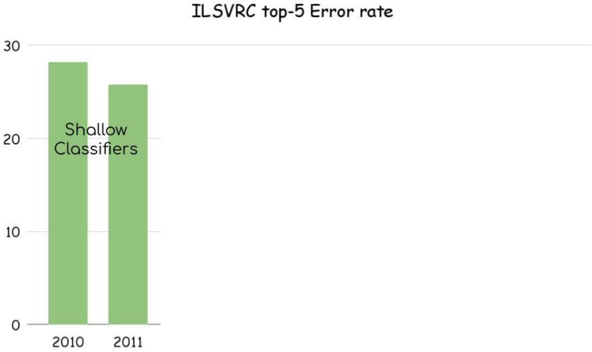


Overview of different CNN architectures

- We will ground the evolution on ILSVRC

Overview of different CNN architectures

- We will ground the evolution on ILSVRC



-

AlexNet (2012)

1. 8-layer CNN: 5 Conv layers, 3 FC layers
2. 227×227 input
3. Max pooling, ReLU nonlinearity, LRN (not used anymore now)

AlexNet (2012)

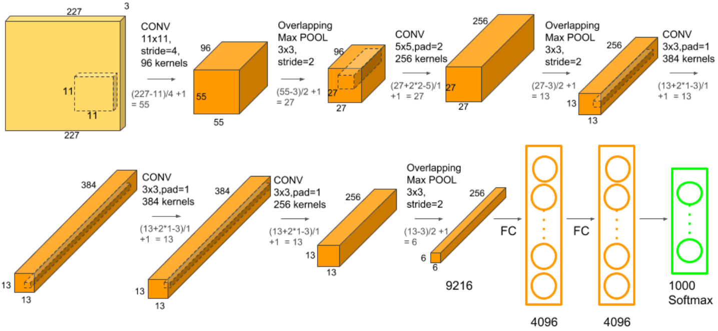


Figure credits: neurohive.io

AlexNet (2012)

1. Implemented on GTX 580 GPUs (2 of them; 3GB of Memory each)

Figure from AlexNet paper by Kryzhevsky et al.

AlexNet (2012)

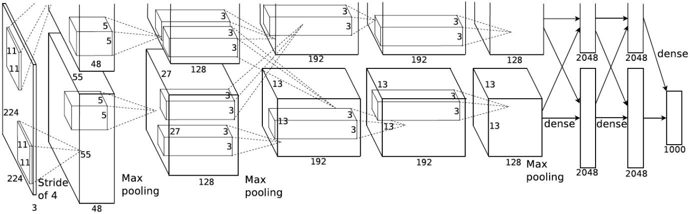
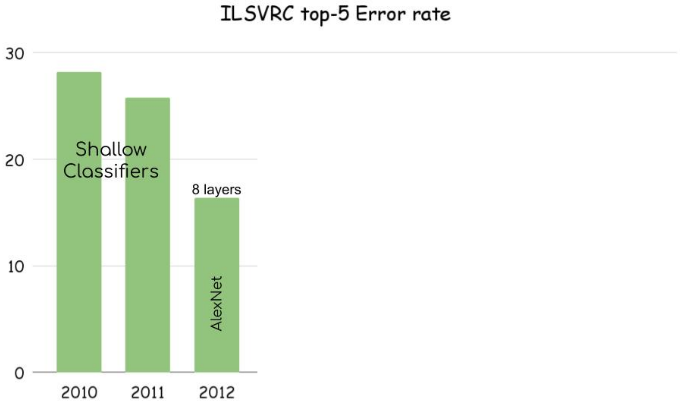
1. Implemented on GTX 580 GPUs (2 of them; 3GB of Memory each)
2. 

Figure from AlexNet paper by Kryzhevsky et al.

AlexNet (2012)



ZFNet (2013)

1. A more worked-out AlexNet

ZFNet (2013)

1. A more worked-out AlexNet
2. More trails on the AlexNet architecture that resulted in less error
   - (11×11 stride 4) $\rightarrow$ (7×7 stride 2)
   - Conv 3, 4, and 5 (384, 384, 256) $\rightarrow$ (512, 1024, and 512)

ZFNet (2013)

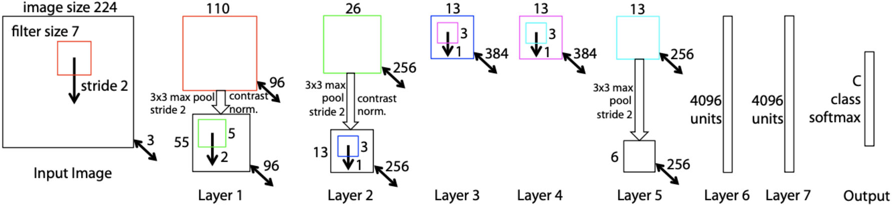
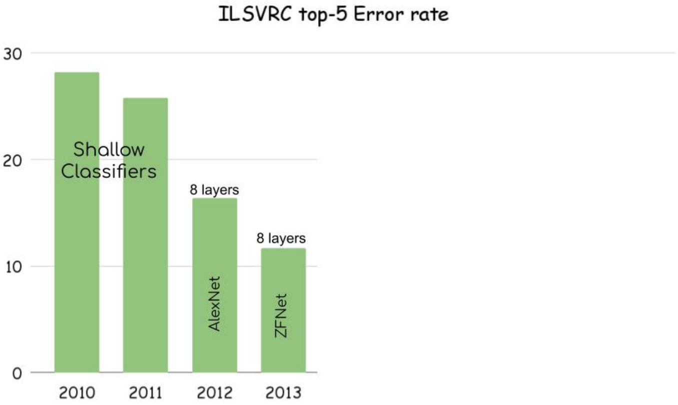


Figure 3. Architecture of our 8 layer convnet model. A 224 by 224 crop of an image (with 3 color planes) is presented as the input. This is convolved with 96 different 1st layer filters (red), each of size 7 by 7, using a stride of 2 in both x and y. The resulting feature maps are then: (i) passed through a rectified linear function (not shown), (ii) pooled (max within 3x3 regions, using stride 2) and (iii) contrast normalized across feature maps to give 96 different 55 by 55 element feature maps. Similar operations are repeated in layers 2,3,4,5. The last two layers are fully connected, taking features from the top convolutional layer as input in vector form ($6 \cdot 6 \cdot 256 = 9216$ dimensions). The final layer is a C-way softmax function, C being the number of classes. All filters and feature maps are square in shape.

Figure from Zeiler and Fergus, ECCV 2014

ZFNet (2013)


ILSVRC top-5 Error rate
30
20
10
0
Shallow Classifiers
8 layers
8 layers
AlexNet
ZFNet
2010
2011
2012
2013

VGG (2014)

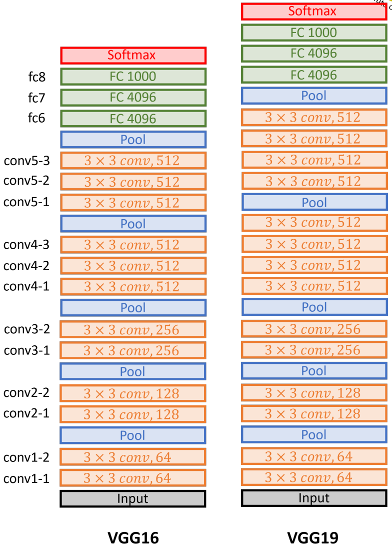
1. First architecture to have a principled design

VGG (2014)

1. First architecture to have a principled design
2. - All conv: 3×3, stride:1, pad:1
   - All max pool: 2×2, stride:2
   - After pooling, double the channels

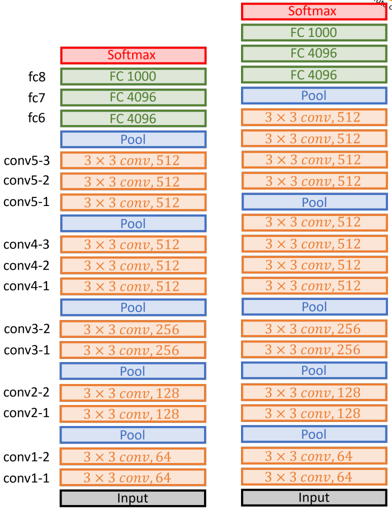
VGG (2014)

1. 5 Conv stages



VGG (2014)

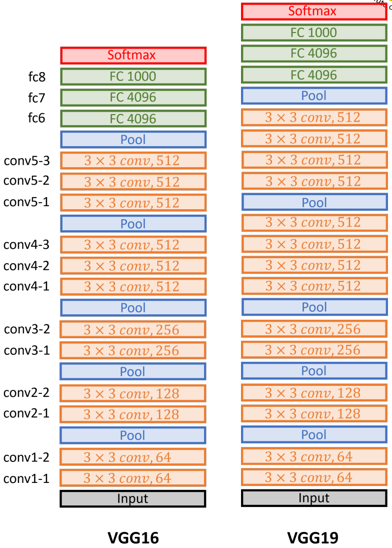
1. 5 Conv stages
2. (initially) Conv-Conv-Pool



VGG16 **VGG19**

VGG (2014)

1. 5 Conv stages
2. (initially) Conv-Conv-Pool
3. (later) Conv-Conv-Conv-Pool (VGG19 has one more Conv)



VGG (2014)

1. Why Only 3×3 Convs?

VGG (2014)

1. Why Only 3×3 Convs?
2. Case-1: Conv$(5 \times 5,\ C \rightarrow C)$

VGG (2014)

1. Why Only 3×3 Convs?
2. Case-1: Conv(5×5, $C \rightarrow C$)
   - Parameters:
     $C \times C \times 5 \times 5 = 25C^2$

VGG (2014)

1. Why Only 3×3 Convs?
2. Case-1: Conv(5×5, $C \rightarrow C$)
   - Parameters:
     $C \times C \times 5 \times 5 = 25C^2$
   - Flops:
     $C \times H \times W \times C \times 5 \times 5 = 25C^2HW$

VGG (2014)

1. Why Only 3×3 Convs?
2. Case-1: Conv(5×5, $C \to C$)
   - Parameters: $C \times C \times 5 \times 5 = 25C^2$
   - Flops: $C \times H \times W \times C \times 5 \times 5 = 25C^2HW$

1. Case-2: Conv(3×3, $C \to C$) and Conv(3×3, $C \to C$)

VGG (2014)

1. Why Only 3×3 Convs?
2. Case-1: Conv(5×5, $C \rightarrow C$)
   - Parameters: $C \times C \times 5 \times 5 = 25C^2$
   - Flops: $C \times H \times W \times C \times 5 \times 5 = 25C^2HW$

1. Case-2: Conv(3×3, $C \rightarrow C$) and Conv(3×3, $C \rightarrow C$)
   - Parameters: $2 \times C \times C \times 3 \times 3 = 18C^2$

VGG (2014)

1. Why Only 3×3 Convs?
2. Case-1: Conv(5×5, $C \rightarrow C$)
   - Parameters: $C \times C \times 5 \times 5 = 25C^2$
   - Flops: $C \times H \times W \times C \times 5 \times 5 = 25C^2HW$

1. Case-2: Conv(3×3, $C \rightarrow C$) and Conv(3×3, $C \rightarrow C$)
   - Parameters: $2 \times C \times C \times 3 \times 3 = 18C^2$
   - Flops: $2 \times C \times H \times W \times C \times 3 \times 3 = 18C^2HW$

VGG (2014)

1. Halving the spatial dimensions (max pooling) and doubling the channels $\rightarrow$ computational cost is unchanged

VGG (2014)

1. Halving the spatial dimensions (max pooling) and doubling the channels $\rightarrow$ computational cost is unchanged
2. Case-1: $C \times 2H \times 2W$, Conv $(3 \times 3,\ C \rightarrow C)$

VGG (2014)

1. Halving the spatial dimensions (max pooling) and doubling the channels $\rightarrow$ computational cost is unchanged
2. Case-1: $C \times 2H \times 2W$, Conv $(3 \times 3,\ C \rightarrow C)$
   - Memory: 4CHW, parameters: $9C^2$, Flops: $36HWC^2$
3. Case-2: $2C \times H \times W$, Conv $(3 \times 3,\ 2C \rightarrow 2C)$
   - Memory: 2CHW, parameters: $36C^2$, Flops: $36HWC^2$

VGG (2014)

1. Huge network (VGG-16) compared to AlexNet

VGG (2014)

1. Huge network (VGG-16) compared to AlexNet
2. Memory: $1.9 \rightarrow 48.6$MB (25X)

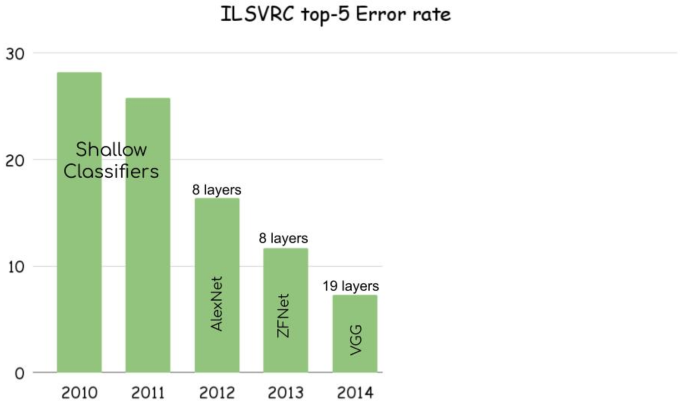
VGG (2014)

1. Huge network (VGG-16) compared to AlexNet
2. Memory: $1.9 \rightarrow 48.6$MB (25X)
3. Parameters: $61 \rightarrow 138$M (2.3X)

VGG (2014)

1. Huge network (VGG-16) compared to AlexNet
2. Memory: $1.9 \rightarrow 48.6$MB (25X)
3. Parameters: $61 \rightarrow 138$M (2.3X)
4. Flops: $0.7 \rightarrow 13.6$G Flop (19.4X)

VGG (2014)



ILSVRC top-5 Error rate
30
20
10
0
Shallow Classifiers
8 layers
8 layers
19 layers
AlexNet
ZFNet
VGG
2010
2011
2012
2013
2014

GoogLeNet (2014)

1. Efficiency was the focus of design

Figure credits:Medium.com and Anas Brital

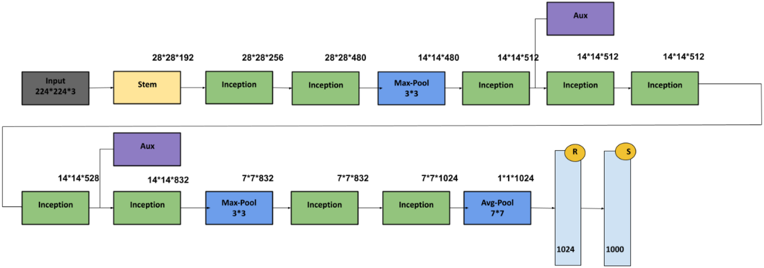
GoogLeNet (2014)

1. Efficiency was the focus of design
2. Reduce the parameters, memory and the compute requirements (towards deployment)

Figure credits:Medium.com and Anas Brital

GoogLeNet (2014)

1. Efficiency was the focus of design
2. Reduce the parameters, memory and the compute requirements (towards deployment)



3.

Figure credits: Medium.com and Anas Brital

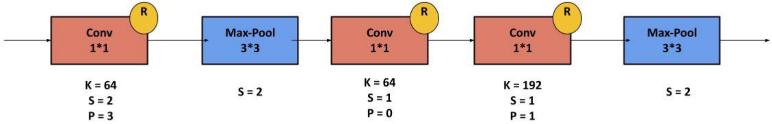
GoogLeNet (2014)

1. Stem architecture at the early stage $\rightarrow$ aggressive down-sampling

Figure credits: Medium.com and Anas Brital

GoogLeNet (2014)

1. Stem architecture at the early stage → aggressive down-sampling

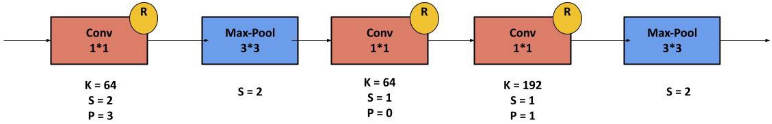


2.

Figure credits: Medium.com and Anas Brital

GoogLeNet (2014)

1. Stem architecture at the early stage $\rightarrow$ aggressive down-sampling



2.
3. From 224×224 to 28×28
   - GoogLeNet: Compute - 7.5MB, parameters - 124K, and MFlops - 418
   - VGG-16: Compute - 42.9MB (5.7X), parameters - 1.1M (8.9X), and MFlops - 7485 (17.8X)

Figure credits: Medium.com and Anas Brital

GoogLenet (2014)

1. Inception module: unit with parallel branches

Figure credits: Medium.com and Anas Brital

GoogLenet (2014)

1. Inception module: unit with parallel branches
2. Repeated through the architecture

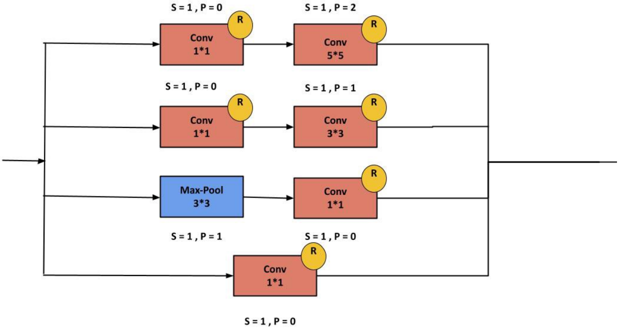


Figure credits: Medium.com and Anas Brital

GoogLenet (2014)

1. Global Average Pooling (GAP) layer

Alexis Cook

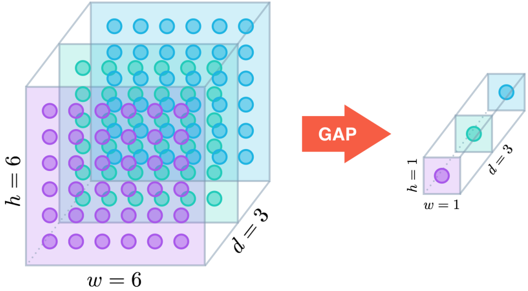
GoogLenet (2014)

1. Global Average Pooling (GAP) layer
2. Flattening results in huge weight matrices $\rightarrow$ GoogLeNet introduces GAP layer

Alexis Cook

GoogLenet (2014)

1. Global Average Pooling (GAP) layer
2. Flattening results in huge weight matrices $\rightarrow$ GoogLeNet introduces GAP layer
3. Collapses the spatial dimensions by computing the average (kernel size = spatial dimensions of the last conv layer)



Alexis Cook

GoogLenet (2014)

1. No more fully connected layers

GoogLenet (2014)

1. No more fully connected layers
2. One linear layer to predict the classification scores (feather light!)

GoogLenet (2014)

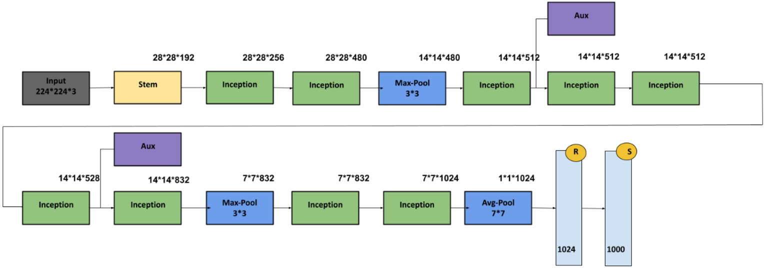
1. Auxiliary classifiers

GoogLenet (2014)

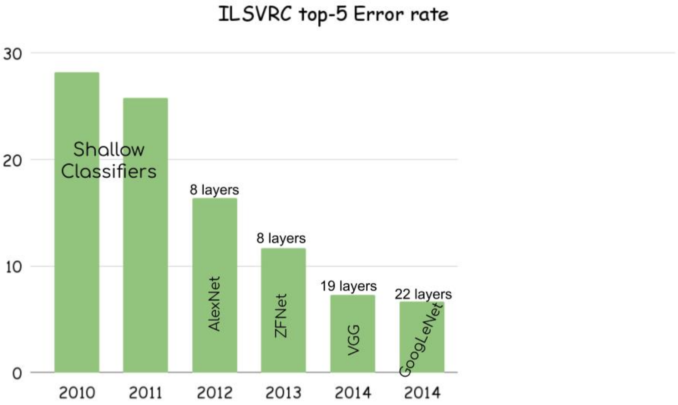
1. Auxiliary classifiers
2. Training using the gradients at the end of the network didn't work well (too deep, gradient propagation was not robust)

GoogLenet (2014)

1. Auxiliary classifiers
2. Training using the gradients at the end of the network didn't work well (too deep, gradient propagation was not robust)
3. Hack: add auxiliary classifiers at intermediate locations to receive loss/gradients



GoogLeNet (2014)

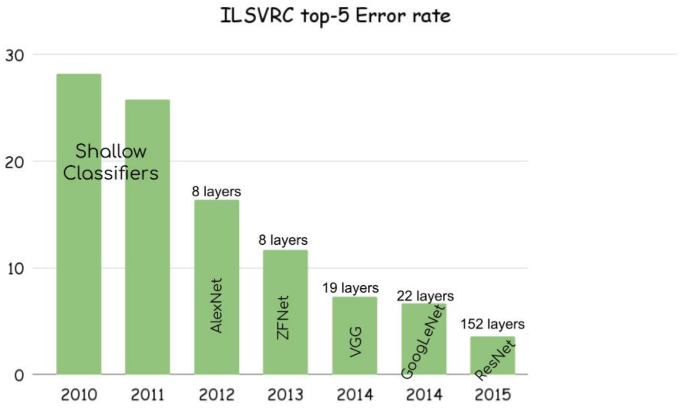


ResNet (2015)

1. Very important time for the DNNs
   - Batch Normalization happened
   - Depth increased by an order $(10 \rightarrow 150+)$
   - ILSVRC error almost halved from that of 2014

ResNet (2015)

1. Very important time for the DNNs
   - Batch Normalization happened
   - Depth increased by an order $(10 \rightarrow 150+)$
   - ILSVRC error almost halved from that of 2014



2.

Training Deeper CNNs

1. When training the “deeper” CNNs, people observed that they were worse than shallow ones

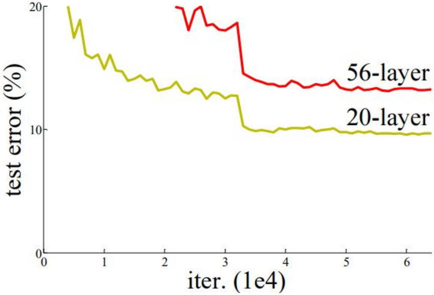
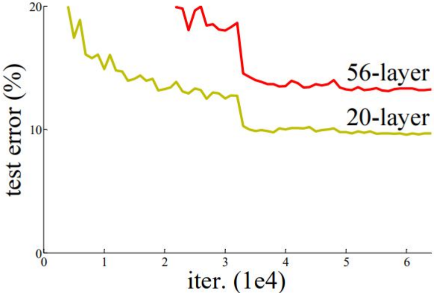


Figure Credits: He et al. 2015

Training Deeper CNNs

1. When training the "deeper" CNNs, people observed that they were worse than shallow ones

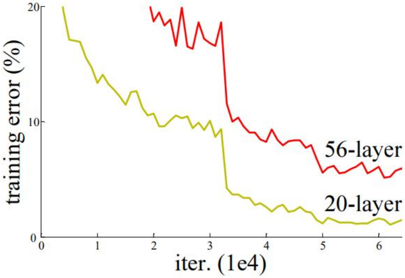


2. Initial suspicion was the 'over-fitting'!

Figure Credits: He et al. 2015

Training Deeper CNNs

1. Initial suspicion was the 'over-fitting'!
2. However, it was due to the under-fitting



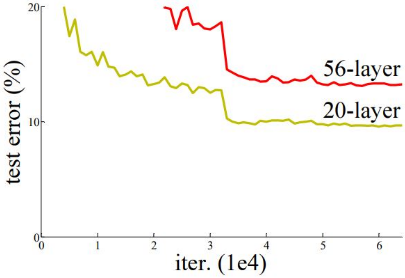


Figure Credits: He et al. 2015

ResNet (2015)

1. Deeper CNNs should easily emulate the shallow ones (extra layers could learn identity function)

ResNet (2015)

1. Deeper CNNs should easily emulate the shallow ones (extra layers could learn identity function)
2. This is not the case $\rightarrow$ some issue in the optimization!

ResNet (2015)

1. Deeper CNNs should easily emulate the shallow ones (extra layers could learn identity function)
2. This is not the case $\rightarrow$ some issue in the optimization!
3. Work on the architecture so that learning identity function gets easier with additional layers

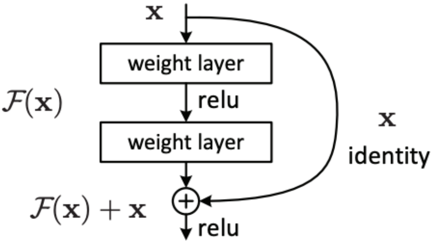
ResNet (2015)

1. Work on the architecture so that learning identity function gets easier with additional layers

Yuanrui Dong

ResNet (2015)

1. Work on the architecture so that learning identity function gets easier with additional layers
2. ResBlock (residual block)



Yuanrui Dong

ResNet (2015)

1. ResBlocks help the gradient backpropagation

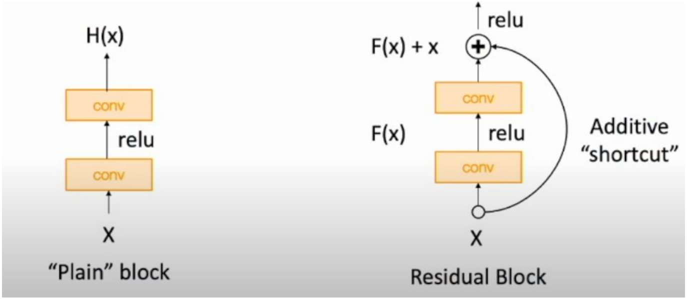


Figure Credits: Dr. Justin Johnson, U Michigan

ResNet (2015)

1. ResNet is a stack of Resblocks

Figure credits: Dr. Justin Johnson, U Michigan

ResNet (2015)

1. ResNet is a stack of Resblocks
2. Inspire from VGG and GoogLeNet

Figure credits: Dr. Justin Johnson, U Michigan

ResNet (2015)

1. ResNet is a stack of Resblocks
2. Inspire from VGG and GoogLeNet
3. Simple and regular design like VGG: each resblock has two 3×3 Conv

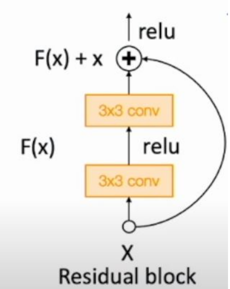


Figure credits: Dr. Justin Johnson, U Michigan

ResNet (2015)

1. Network has stages: first block of each stage halves the resolution and doubles the channels

ResNet (2015)

1. Network has stages: first block of each stage halves the resolution and doubles the channels
2. Aggressive stem in the beginning (downsamples by 4X before the start of the resblocks)

ResNet (2015)

1. Network has stages: first block of each stage halves the resolution and doubles the channels
2. Aggressive stem in the beginning (downsamples by 4X before the start of the resblocks)
3. Eliminates the FC layers via GAP

ResNet (2015)

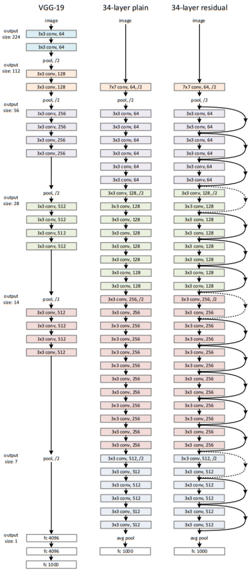


Figure credits: K. he et al., ResNets 92015)

ResNet (2015)

1. ResNet-18
   - Stem: 1 Conv
   - Stage-1 (C=64): 2 resblocks (4 Conv)
   - Stage-2 (C=128): 2 resblocks (4 Conv)
   - Stage-3 (C=256): 2 resblocks (4 Conv)
   - Stage-4 (C=512): 2 resblocks (4 Conv)
   - Linear
   - Top-5 error: 10.92 and GFlop: 1.8

ResNet (2015)

1. ResNet-34
   - Stem: 1 Conv
   - Stage-1 (C=64): 3 resblocks (6 Conv)
   - Stage-2 (C=128): 4 resblocks (8 Conv)
   - Stage-3 (C=256): 6 resblocks (12 Conv)
   - Stage-4 (C=512): 3 resblocks (6 Conv)
   - Linear
   - Top-5 error: 8.58 and GFlop: 3.6 (VGG: 9.6 and 13.6 respectively)

ResNet (2015)

1. Bottlneck Residual block

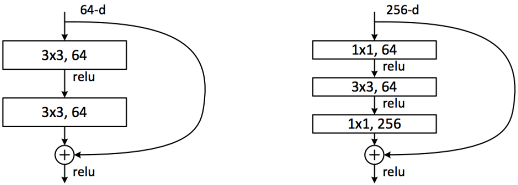


Figure Credits: Nushaine Ferdinand

ResNet (2015)

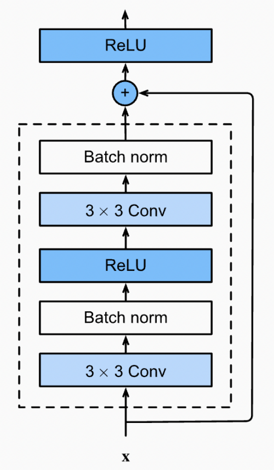
1. Resnet-34 becomes ResNet-50 if we replace the plain resblocks withe bottleneck ones

ResNet (2015)

1. Resnet-34 becomes ResNet-50 if we replace the plain resblocks withe bottleneck ones
2. More blocks at each stage result in ResNet-101 and Resnet-152 architectures

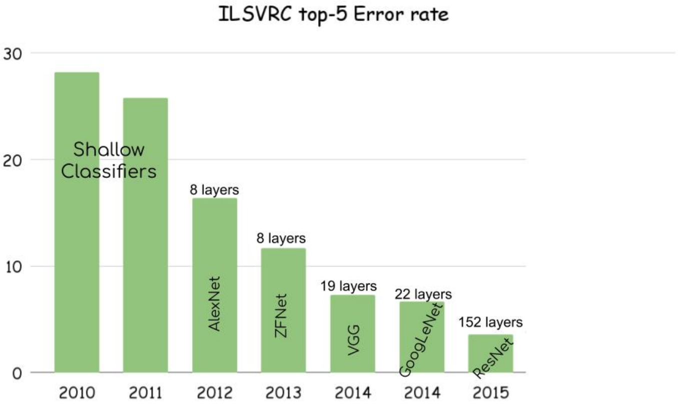
Resnet (2015)

1. Resblocks have Batch Normalization layers



Yashovardhan Shinde and Analyticsvidhya

ResNet (2015)



ILSVRC top-5 Error rate
30
20
10
0
Shallow Classifiers
8 layers
8 layers
19 layers
22 layers
152 layers
AlexNet
ZFNet
VGG
GoogLeNet
ResNet
2010
2011
2012
2013
2014
2014
2015

Post 2015

1. 2016 Winners (Trimps Soushen): Multi-scale Ensemble models of Inception, ResNets, WRN, etc.

Post 2015

1. 2016 Winners (Trimps Soushen): Multi-scale Ensemble models of Inception, ResNets, WRN, etc.
2. Improving ResNets: multiple parallel pathways of bottlenecks (ResNeXt), Squeeze and Excitation Nets (SENet)
3. Densenets, Tiny Networks (MobileNets, ShuffleNets), etc.

CNN Architectures: Summary

1. Initial families of architectures (AlexNet, ZFNet, VGG) $\rightarrow$ Bigger the better!

CNN Architectures: Summary

1. Initial families of architectures (AlexNet, ZFNet, VGG) $\rightarrow$ Bigger the better!
2. GoogLeNet emphasized on efficiency

CNN Architectures: Summary

1. Initial families of architectures (AlexNet, ZFNet, VGG) $\rightarrow$ Bigger the better!
2. GoogLeNet emphasized on efficiency
3. ResNet enabled extreme depth

CNN Architectures: Summary

1. Initial families of architectures (AlexNet, ZFNet, VGG) $\rightarrow$ Bigger the better!
2. GoogLeNet emphasized on efficiency
3. ResNet enabled extreme depth
4. Focus back on efficiency: improving accuracy w/o growing the complexity

CNN Architectures: Summary

1. Initial families of architectures (AlexNet, ZFNet, VGG) $\rightarrow$ Bigger the better!
2. GoogLeNet emphasized on efficiency
3. ResNet enabled extreme depth
4. Focus back on efficiency: improving accuracy w/o growing the complexity
5. Deploy-able models: MobileNet, ShuffleNet, etc.

CNN Architectures: Summary

1. Initial families of architectures (AlexNet, ZFNet, VGG) $\rightarrow$ Bigger the better!
2. GoogLeNet emphasized on efficiency
3. ResNet enabled extreme depth
4. Focus back on efficiency: improving accuracy w/o growing the complexity
5. Deploy-able models: MobileNet, ShuffleNet, etc.
6. Neural Architecture Search (NAS)