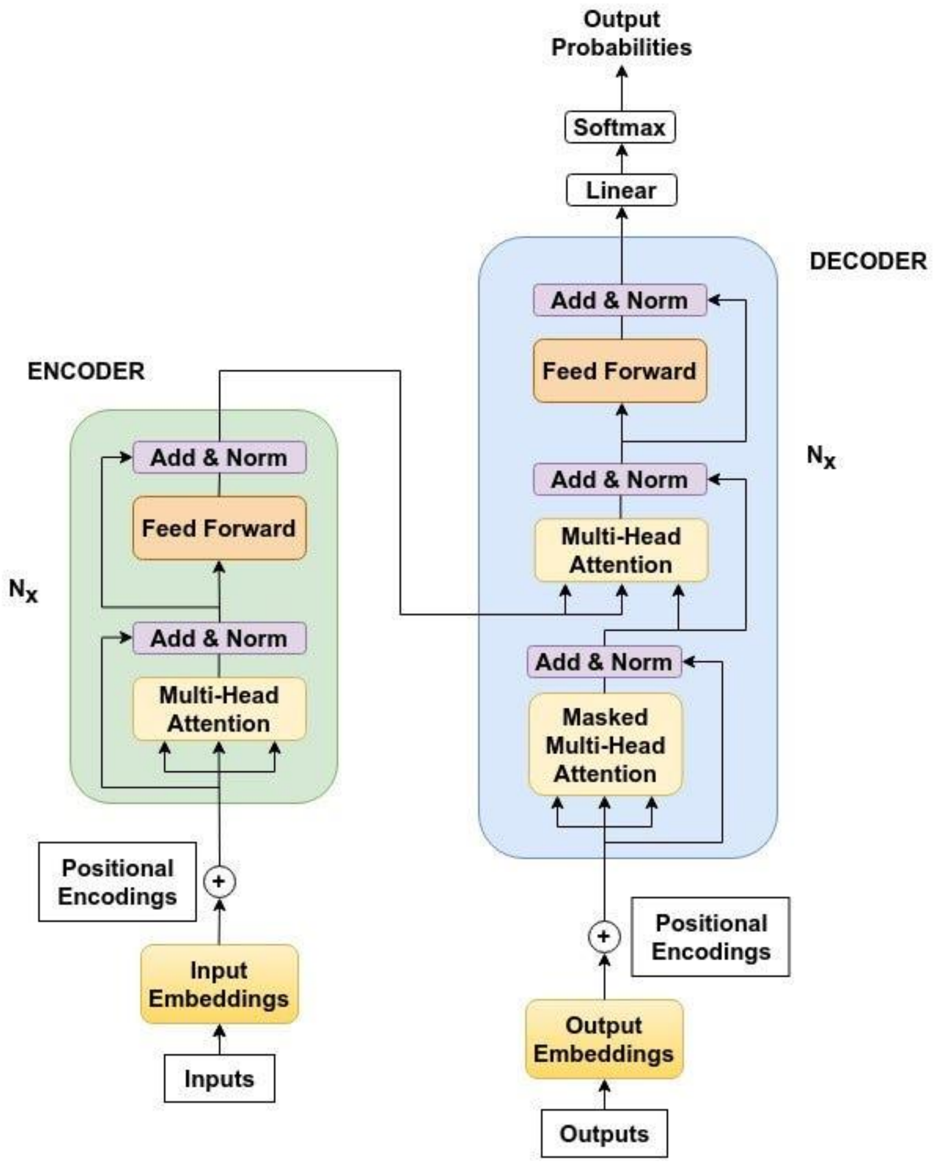

Output
Probabilities
Softmax
Linear
DECODER
Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Positional
Encodings
Output
Embeddings
Outputs
ENCODER
Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head
Attention
Positional
Encodings
Input
Embeddings
Inputs

Residual Network (ResNet)

By
Prof. Khaled Mostafa El Sayed
Faculty of Computers and Information
Cairo University

Many Slides are obtained from
Kaiming He, Xiangyu Zhang, Shaoqing, Jain Sun
Fei-Fei Li & Justin Johnson & Serena Yeung

ResNet Output Performance

ResNet Performance

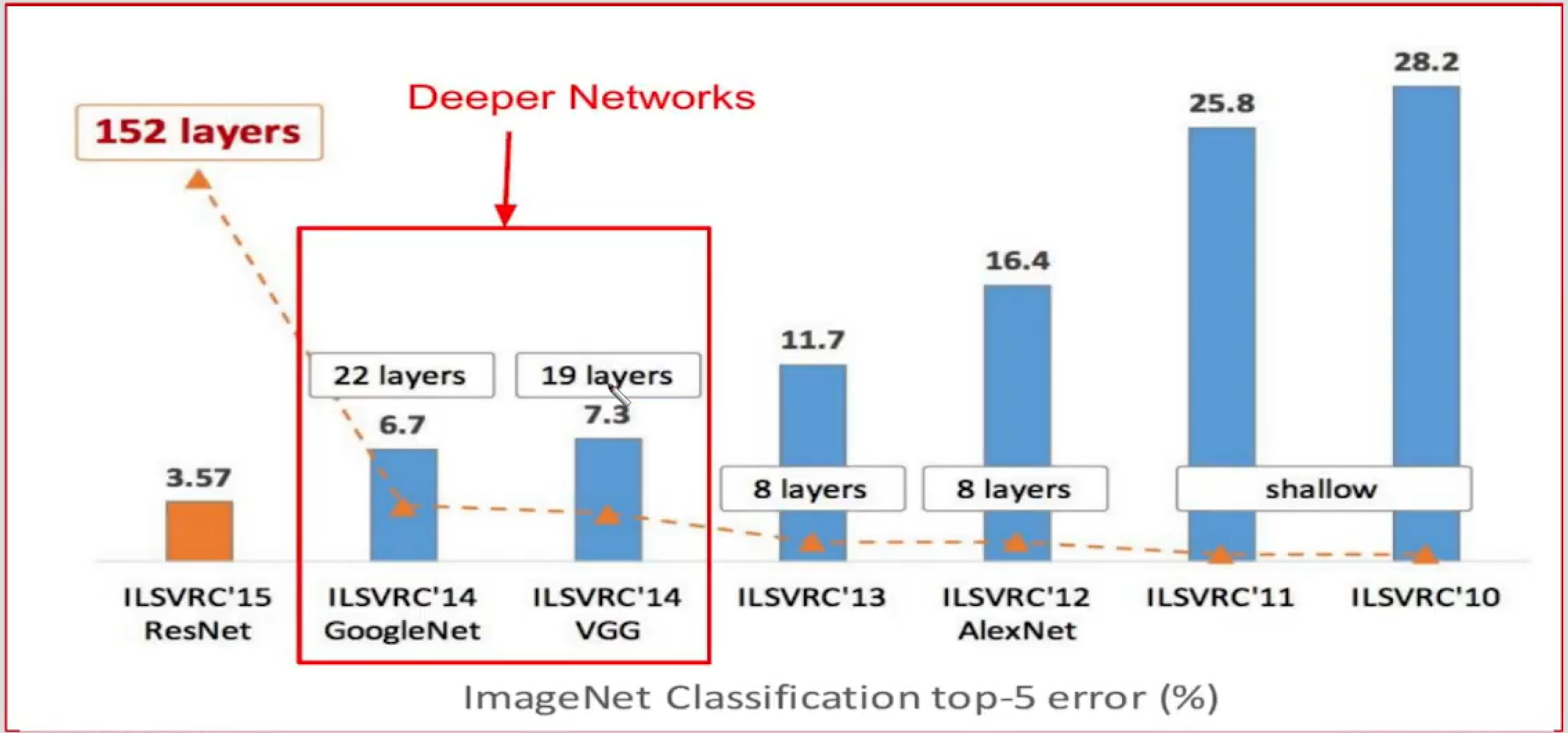
ResNet @ ILSVRC & COCO 2015 Competitions

1st places in all five main tracks

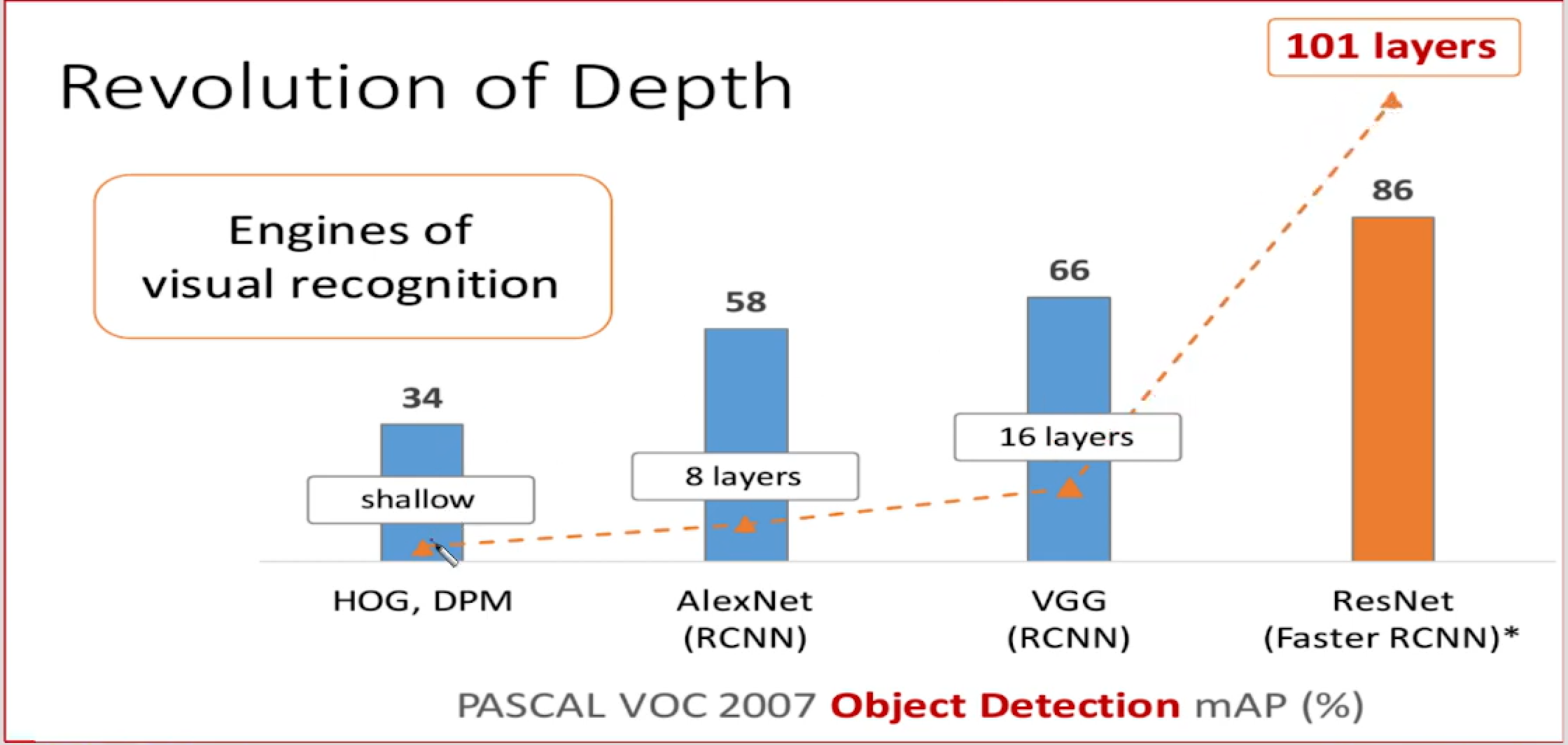
- ImageNet Classification: "*Ultra-deep*" 152-layer nets
- ImageNet Detection: 16% better than 2nd
- ImageNet Localization: 27% better than 2nd
- COCO Detection: 11% better than 2nd
- COCO Segmentation: 12% better than 2nd

ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

ResNet Performance



Revolution of Depth



Khaledms@fci-cu.edu.eg

Revolution of Depth

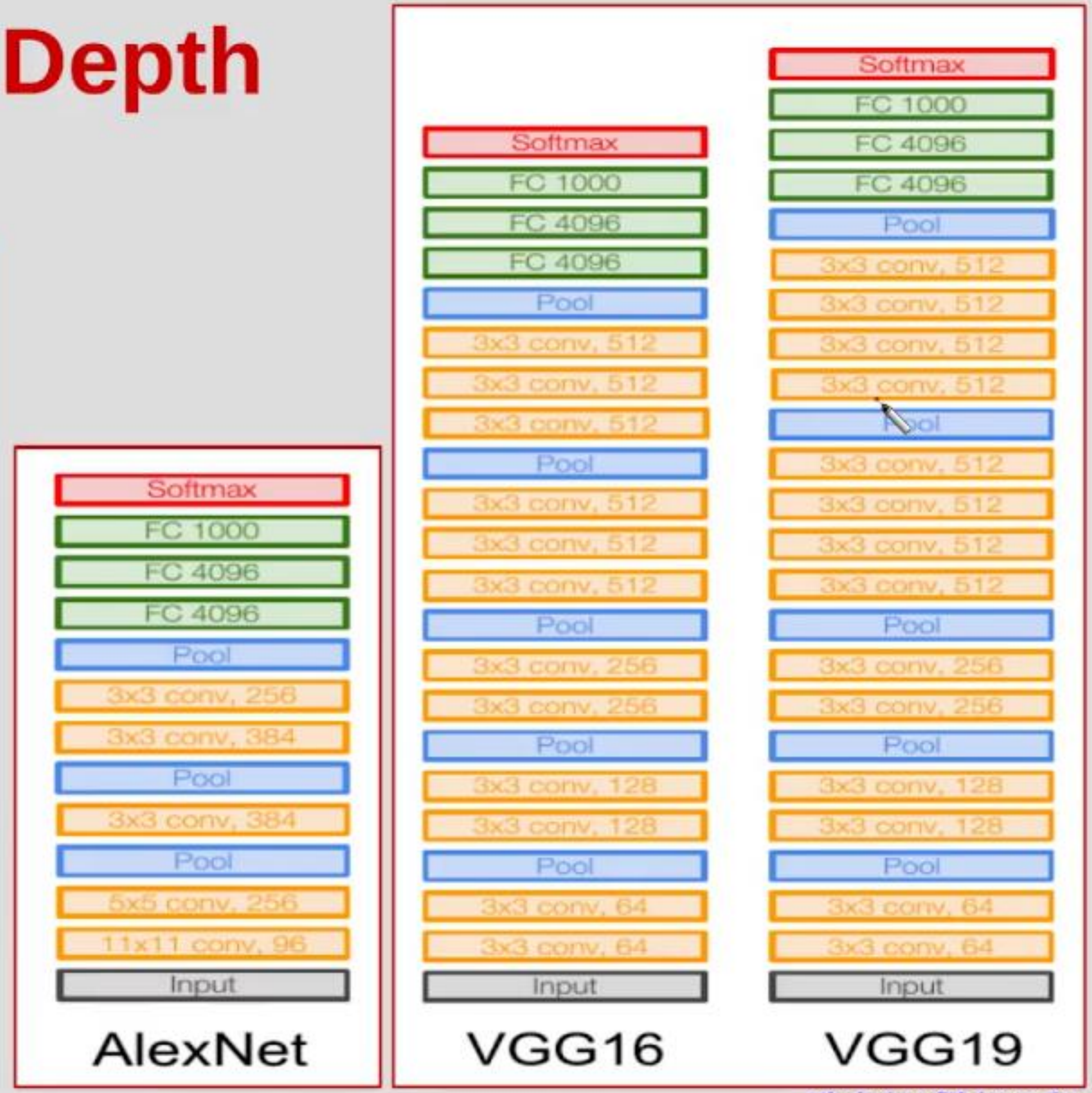
Small filters, Deeper networks

8 layers (AlexNet)
-> 16 - 19 layers (VGG16Net)

Only 3x3 CONV stride 1, pad 1
and 2x2 MAX POOL stride 2

11.7% top 5 error in ILSVRC'13
(ZFNet)
-> 7.3% top 5 error in ILSVRC'14

Stack of three 3x3 conv (stride 1) layers
has same effective receptive field as
one 7x7 conv layer

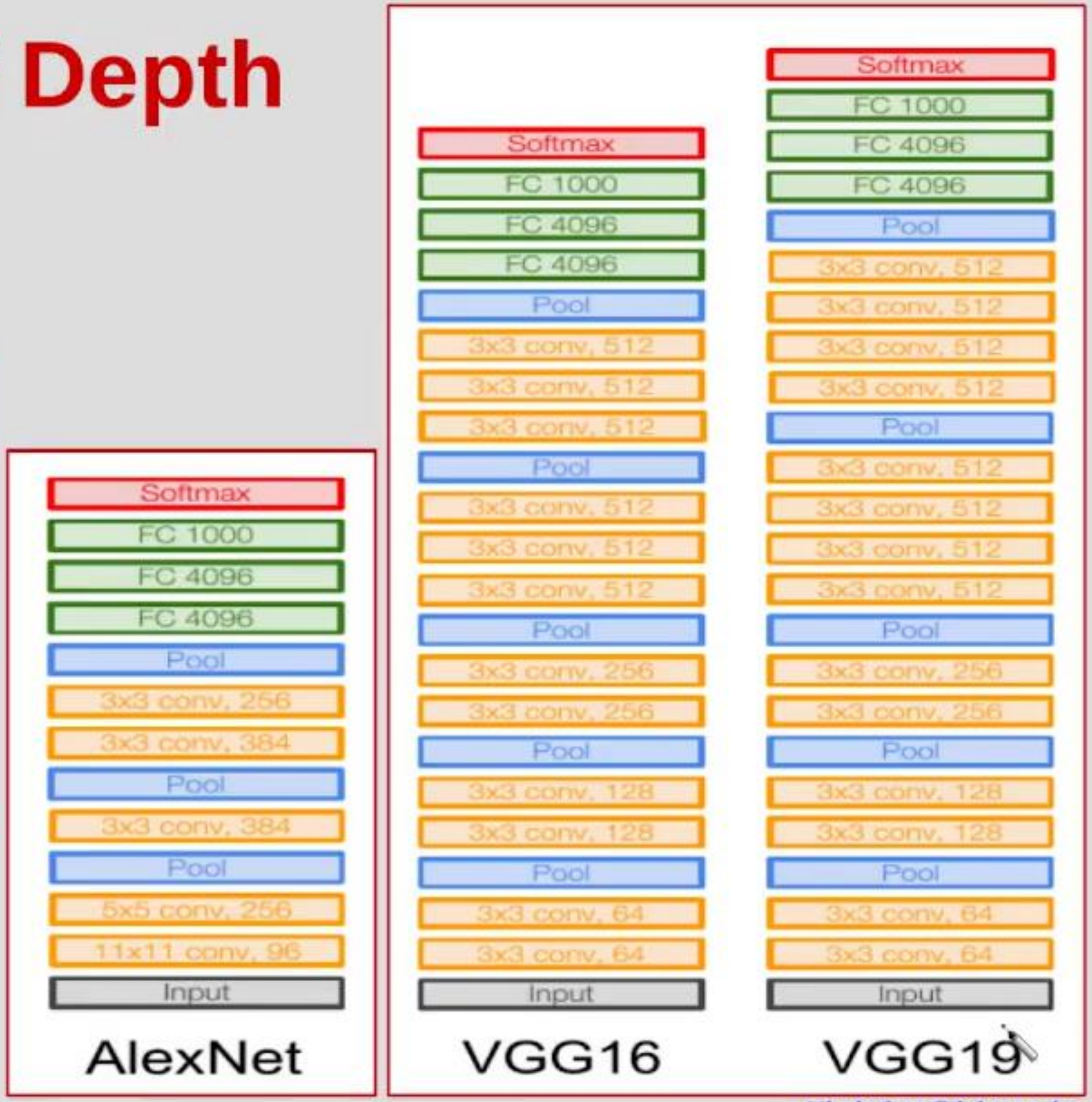


Revolution of Depth

Stack of three 3x3 conv (stride 1) layers has same **effective receptive field** as one 7x7 conv layer

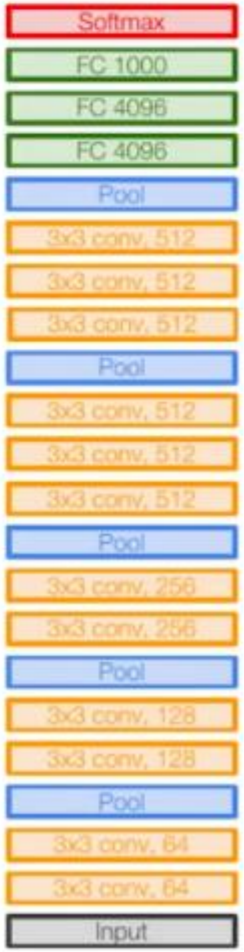
But deeper, more non-linearities

And fewer parameters: $3 * (3^2C^2)$ vs. 7^2C^2 for C channels per layer



(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000



VGG16

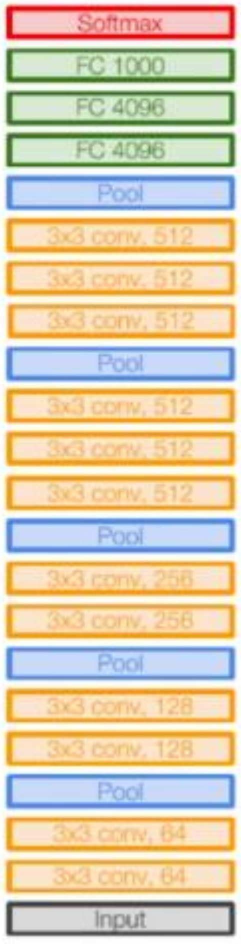
TOTAL memory: 24M * 4 bytes ~= 96MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

(not counting biases)

INPUT: [224x224x3] memory: 224*224*3=150K params: 0
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*3)*64 = 1,728
CONV3-64: [224x224x64] memory: 224*224*64=3.2M params: (3*3*64)*64 = 36,864
POOL2: [112x112x64] memory: 112*112*64=800K params: 0
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*64)*128 = 73,728
CONV3-128: [112x112x128] memory: 112*112*128=1.6M params: (3*3*128)*128 = 147,456
POOL2: [56x56x128] memory: 56*56*128=400K params: 0
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*128)*256 = 294,912
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
CONV3-256: [56x56x256] memory: 56*56*256=800K params: (3*3*256)*256 = 589,824
POOL2: [28x28x256] memory: 28*28*256=200K params: 0
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*256)*512 = 1,179,648
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
CONV3-512: [28x28x512] memory: 28*28*512=400K params: (3*3*512)*512 = 2,359,296
POOL2: [14x14x512] memory: 14*14*512=100K params: 0
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
CONV3-512: [14x14x512] memory: 14*14*512=100K params: (3*3*512)*512 = 2,359,296
POOL2: [7x7x512] memory: 7*7*512=25K params: 0
FC: [1x1x4096] memory: 4096 params: 7*7*512*4096 = 102,760,448
FC: [1x1x4096] memory: 4096 params: 4096*4096 = 16,777,216
FC: [1x1x1000] memory: 1000 params: 4096*1000 = 4,096,000

Maximum Memory Usage

Maximum Number of Parameters

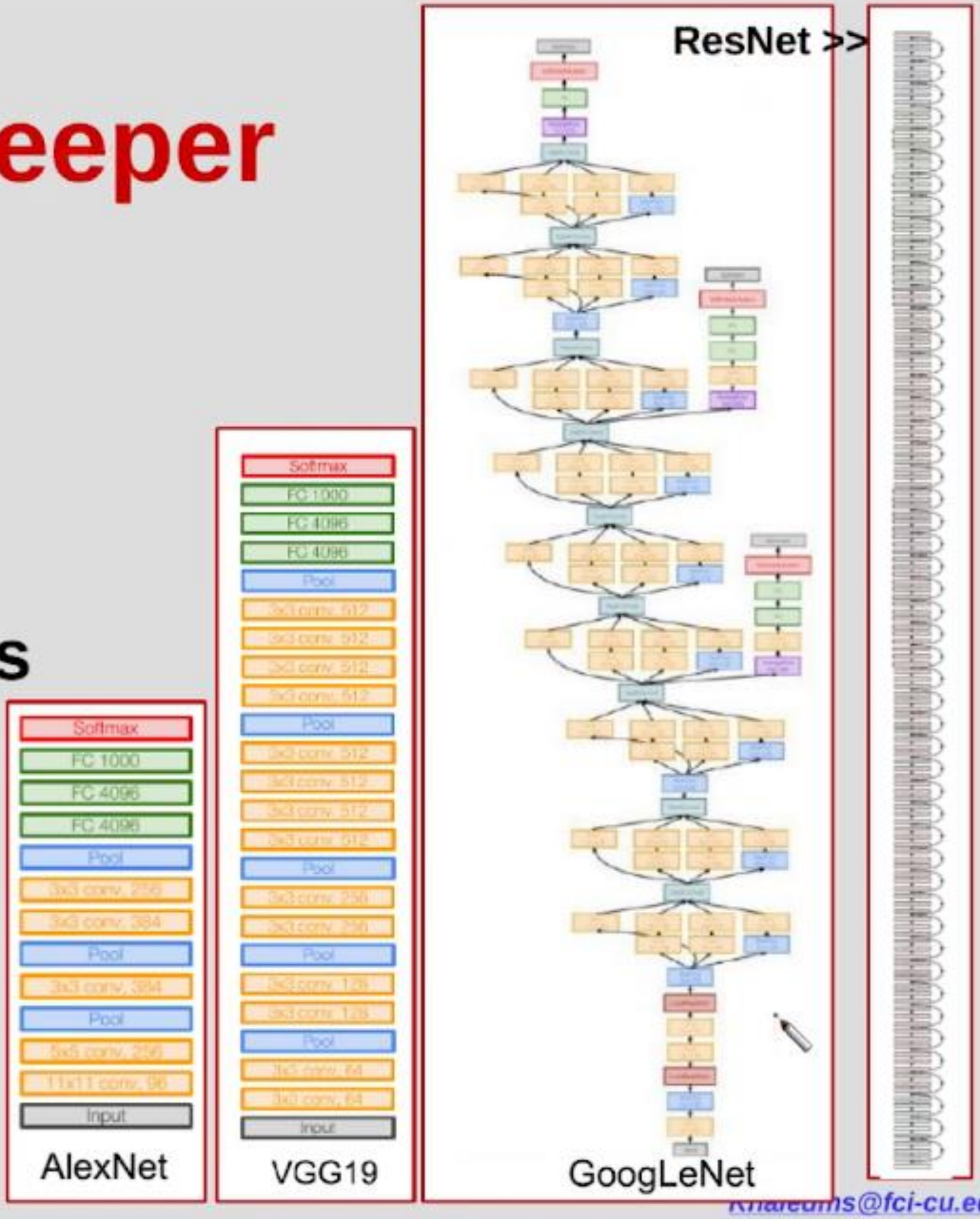


VGG16

TOTAL memory: 24M * 4 bytes ~= 96MB / image (only forward! ~*2 for bwd)
TOTAL params: 138M parameters

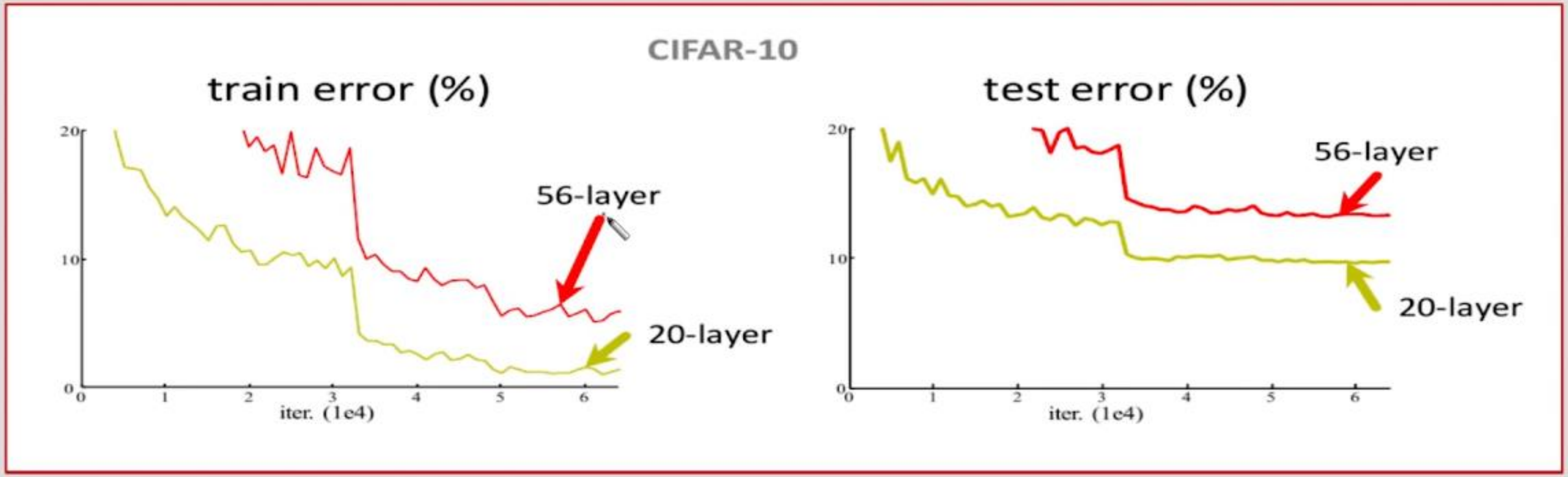
Go Deeper

AlexNet (2012) 8 Layers
VGG (2014) 19 Layers
GooLeNet (2014) 22 Layers
ResNet (2015) 152 Layers



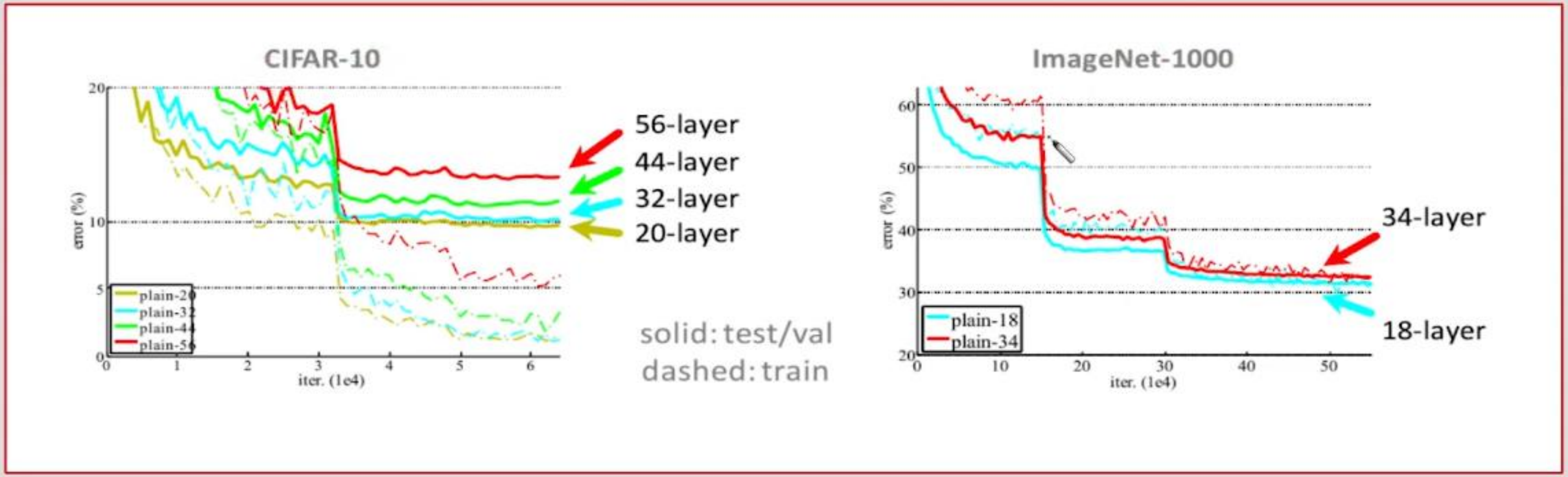
Training problem of “Stacking Layers”
Deep Architectures

Error Behavior when Increasing Number of Layers (Train and Test Errors)



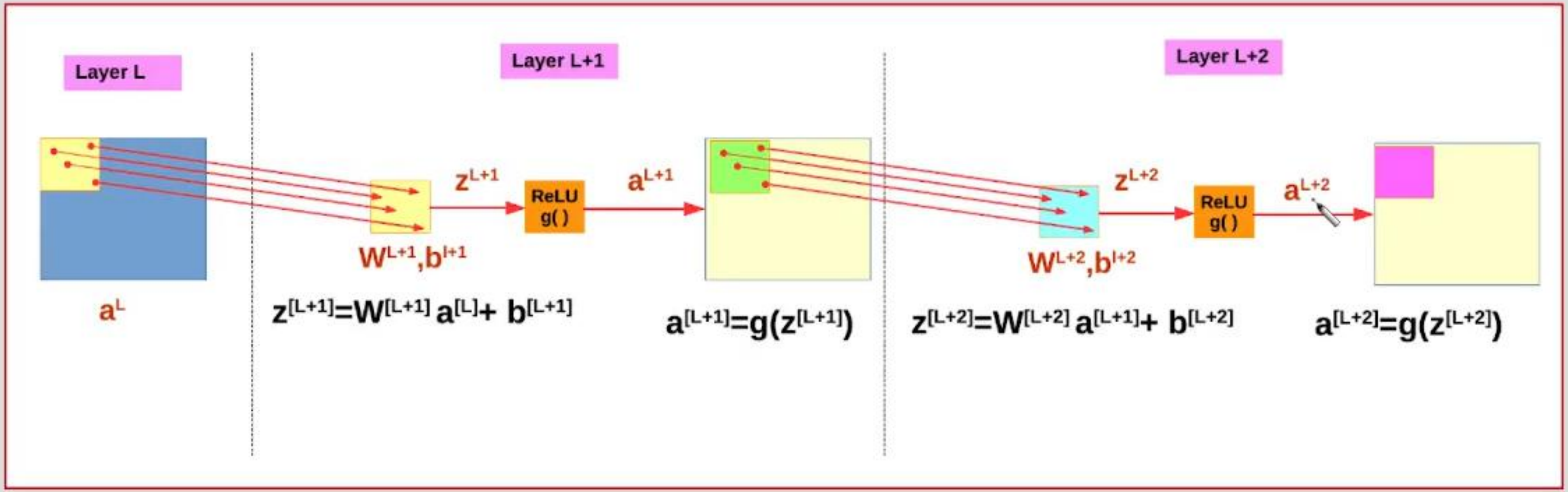
Plain nets: stacking 3x3 convlayers...
56-layer net has higher training errorand test errorthan 20-layer net

Error Behavior when Increasing Number of Layers (Different Datasets)



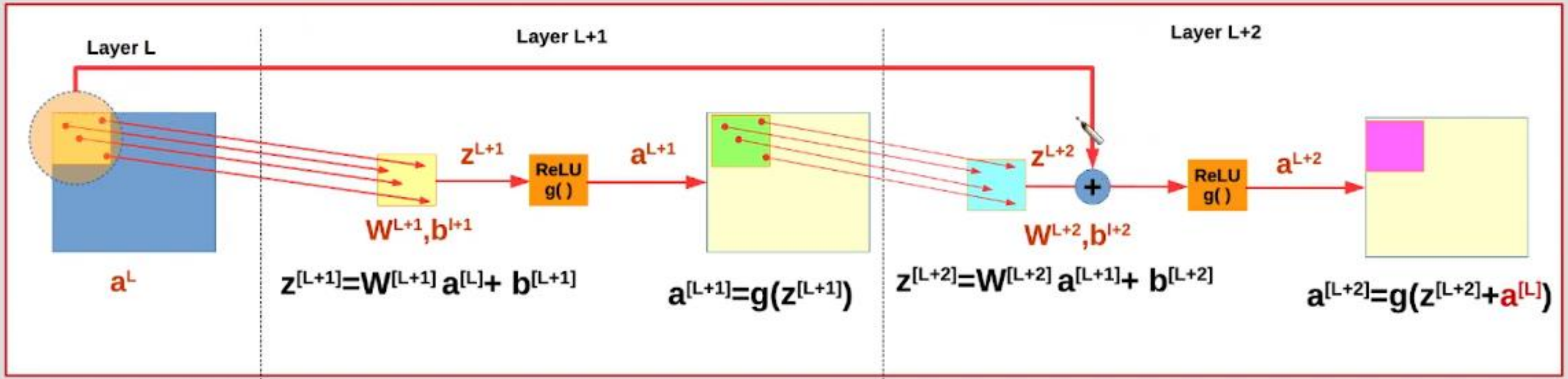
"Overly deep" plain nets have higher training error
A general phenomenon, observed in many datasets

Apply Single Filter on One Block

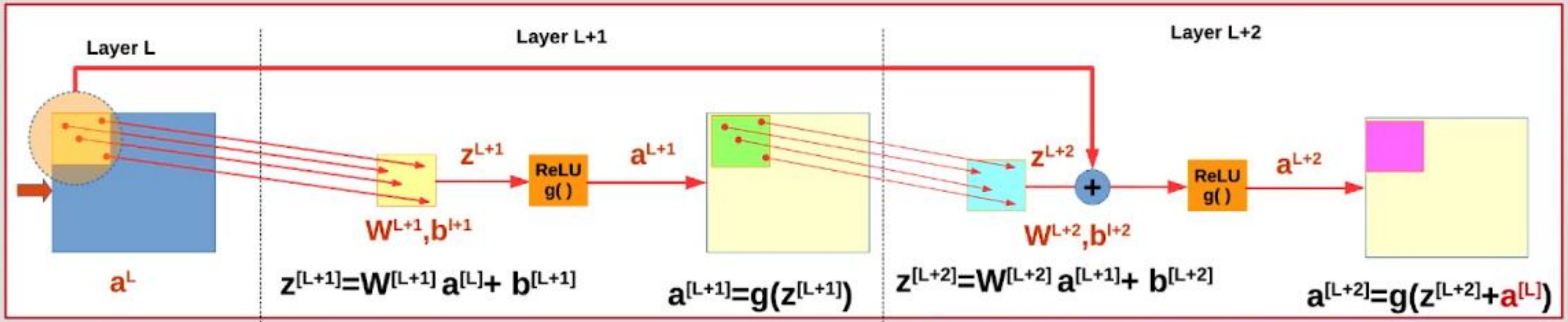


$W^{[L+1]}$	Weights of Filter at Layer "L+1"
$b^{[L+1]}$	Bias of Filter at Layer "L+1"
$z^{[L+1]}$	Output After Applying Filters on Layer "L"
$a^{[L+1]}$	Output of ReLU {g(..)} at Layer "L+1"

Apply Single Filter on One Block and Feedback from Prev Input



Learn Identity Function Using Residual Block



Using **Relu** Activation function

$g(a^{[L]}) = a^{[L]}$ ==> if **a** is positive value

Remember:

Output of ReLU is Always Positive

$a^{[L]}$ is Relu output of Previous Layer ==> $a^{[L]}$ Always Positive

If:

$W^{[L+2]} = 0$

$b^{[L+2]} = 0$

Then

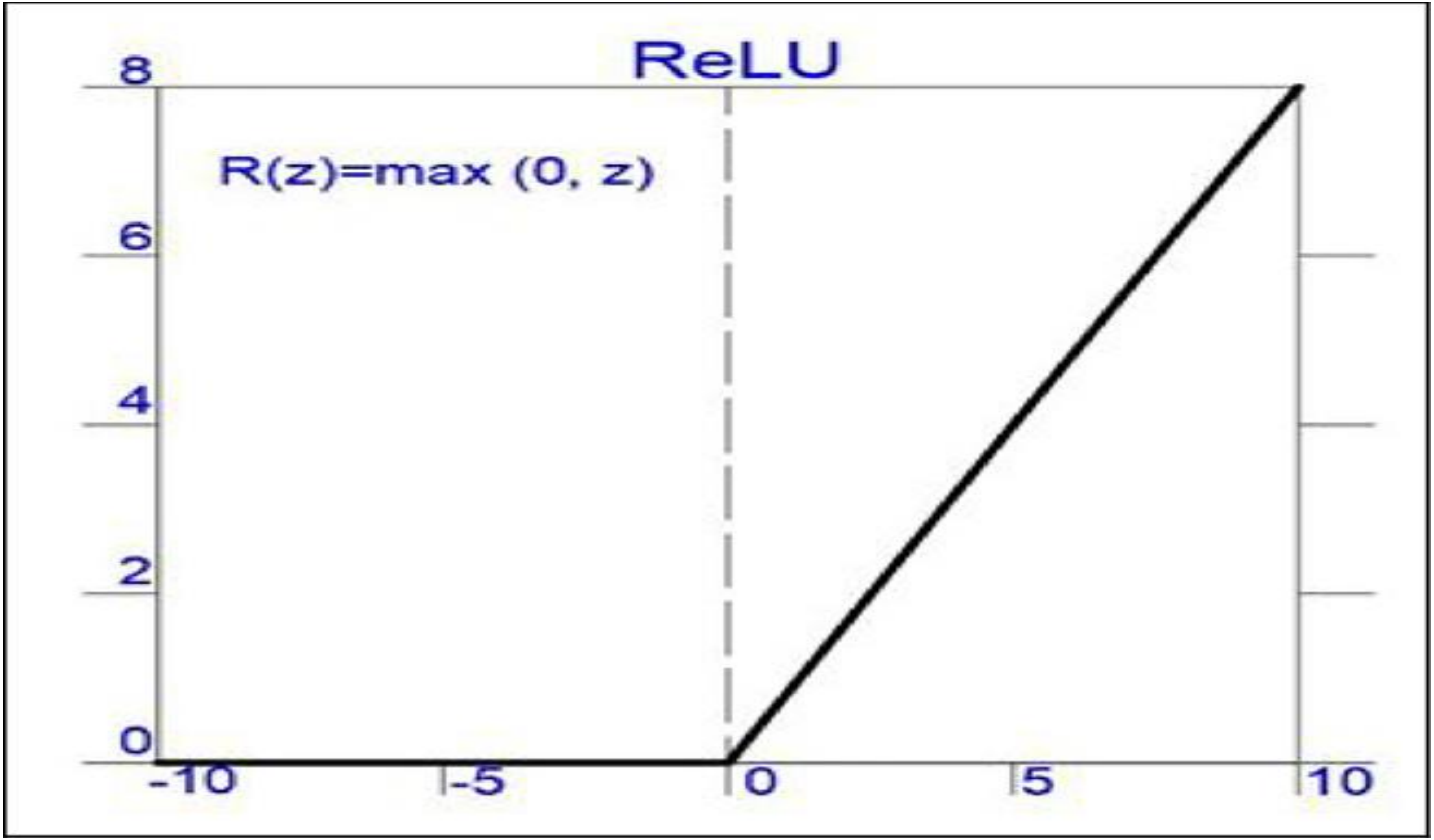
$z^{[L+2]} = 0$

And

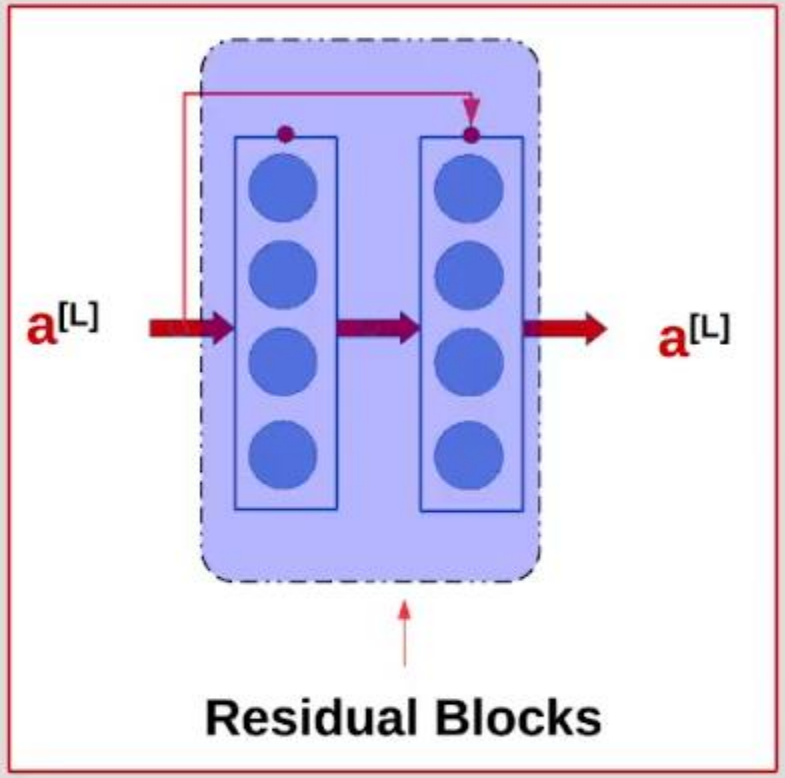
$a^{[L+2]} = g(z^{[L+2]} + a^{[L]}) = a^{[L]}$

System Can Lear Identity Function

Khaledms@fci-cu.edu.eg

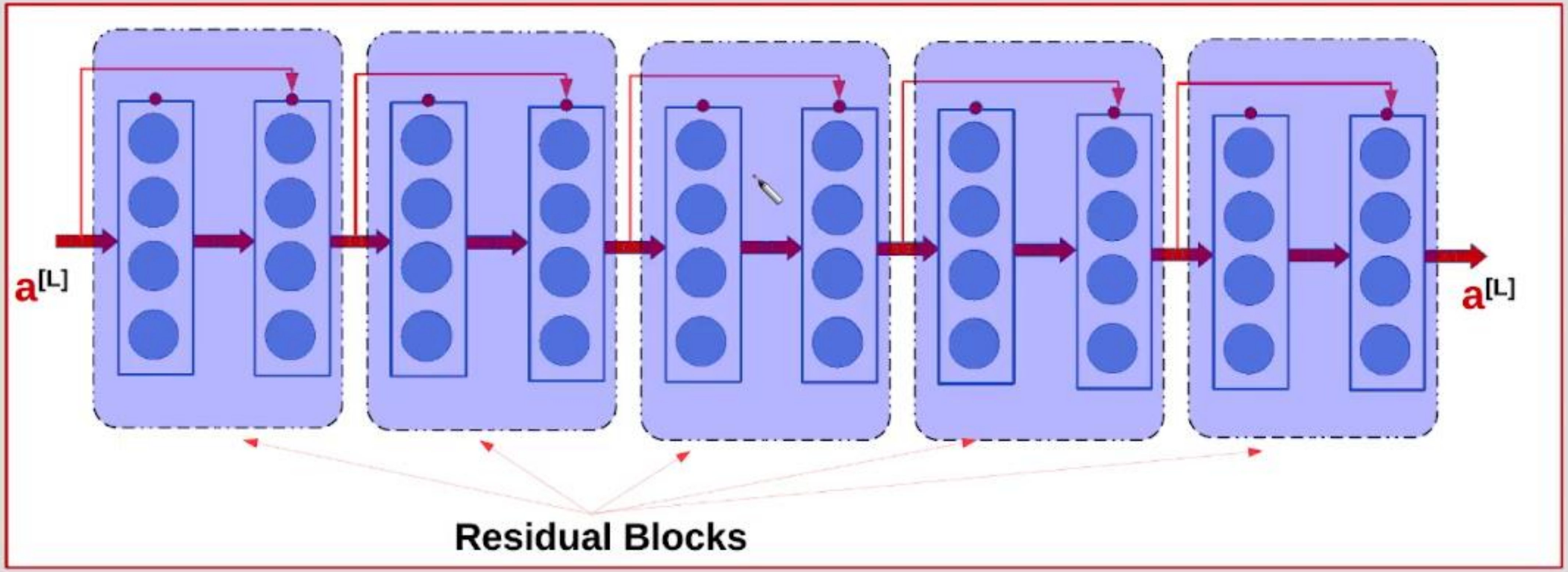

ReLU
R(z)=max (0, z)
8
6
4
2
0
-10
-5
0
5
10

Residual Blocks Can Learn Identity Function



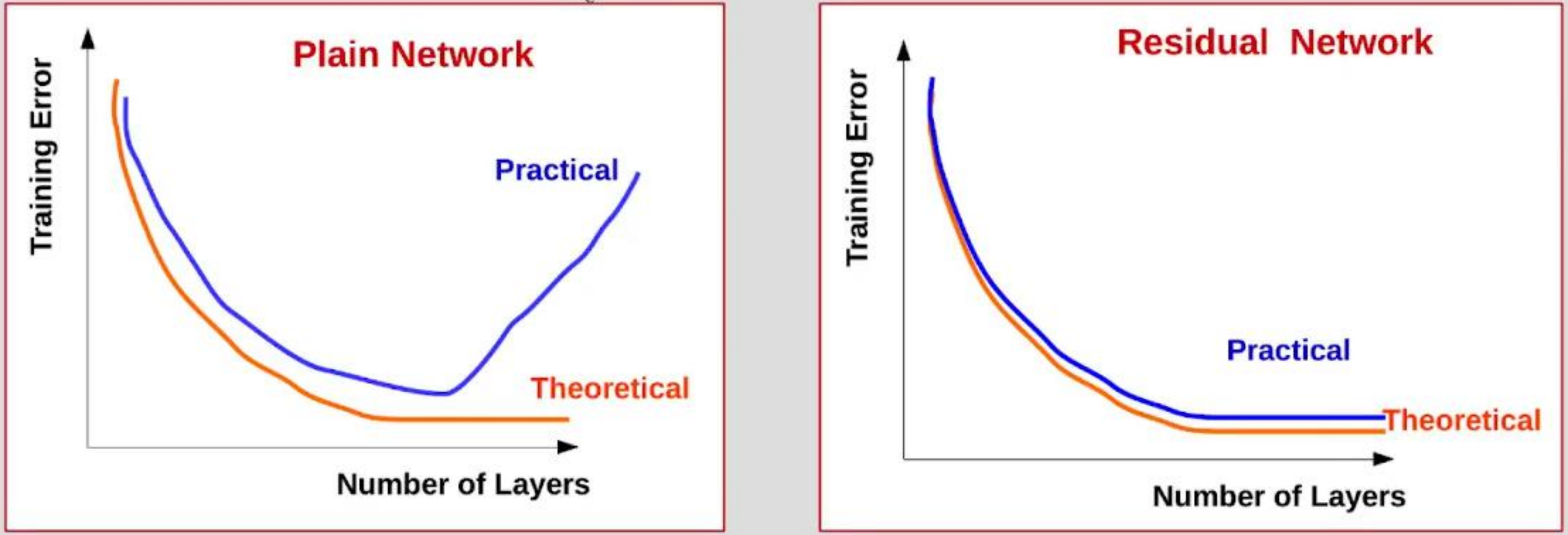
Residual block Can LEARN to Output same input

Cascaded Residual Blocks Can Learn Identity Function



Go Deeper will NOT Harm output of the Network

Plain vs Residual Netwok
effect of increasing Number of Layers



ResNet Architecture