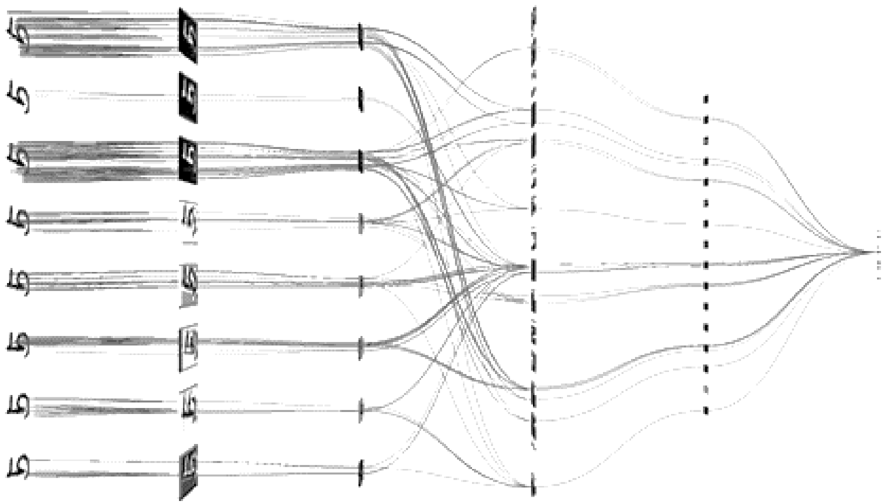
Convolutional Neural Network

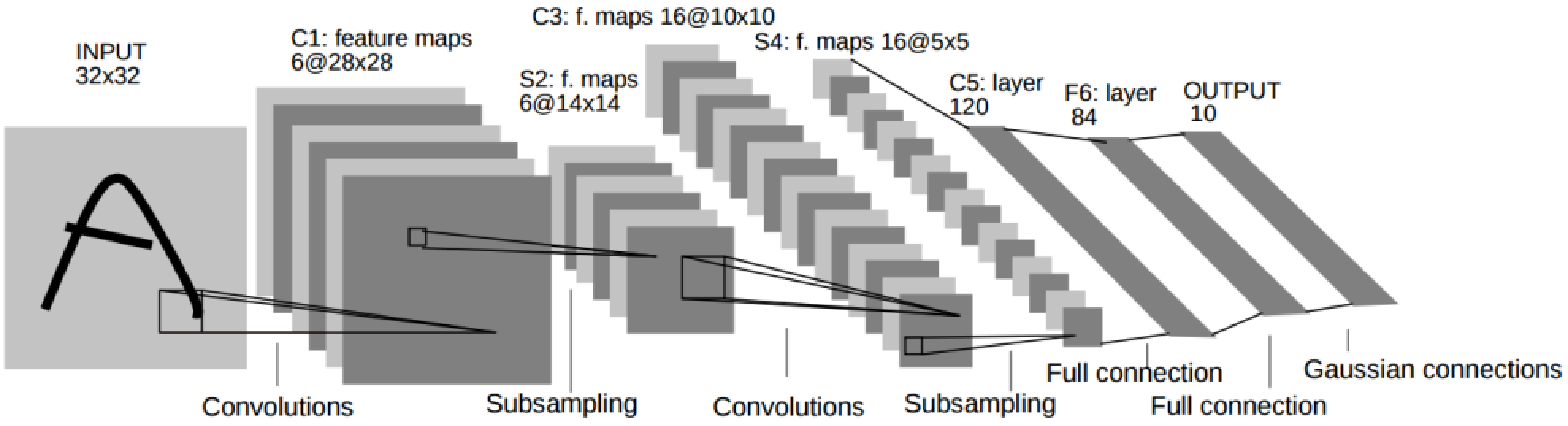
Case Study



Fast Campus
Start Deep Learning with Tensorflow

LeNet-5 (LeCun et al., 1998)

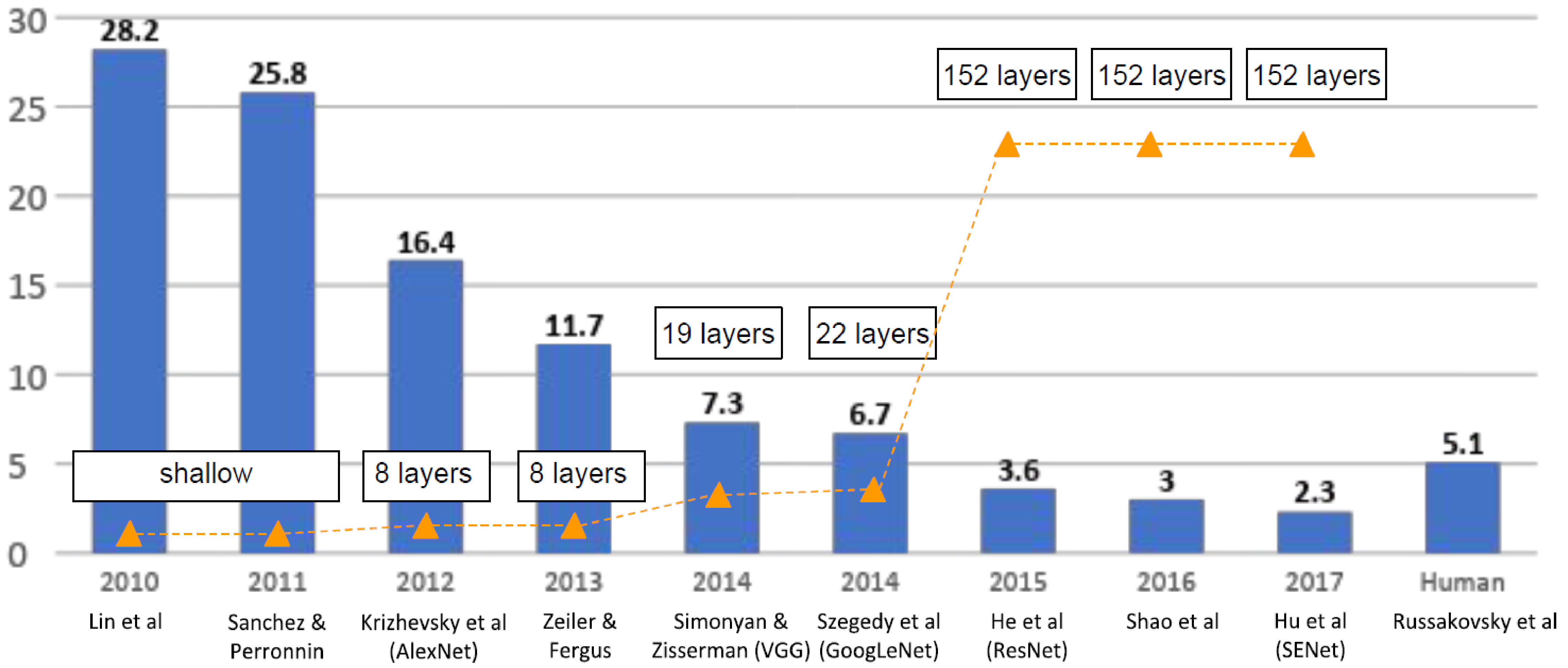
- A father of CNN
- Convolution filters were 5x5, applied at stride 1
- Subsampling(Pooling) layers were 2x2 applied at stride 2
- [Conv – Pool – Conv – Pool – FC – FC – FC]



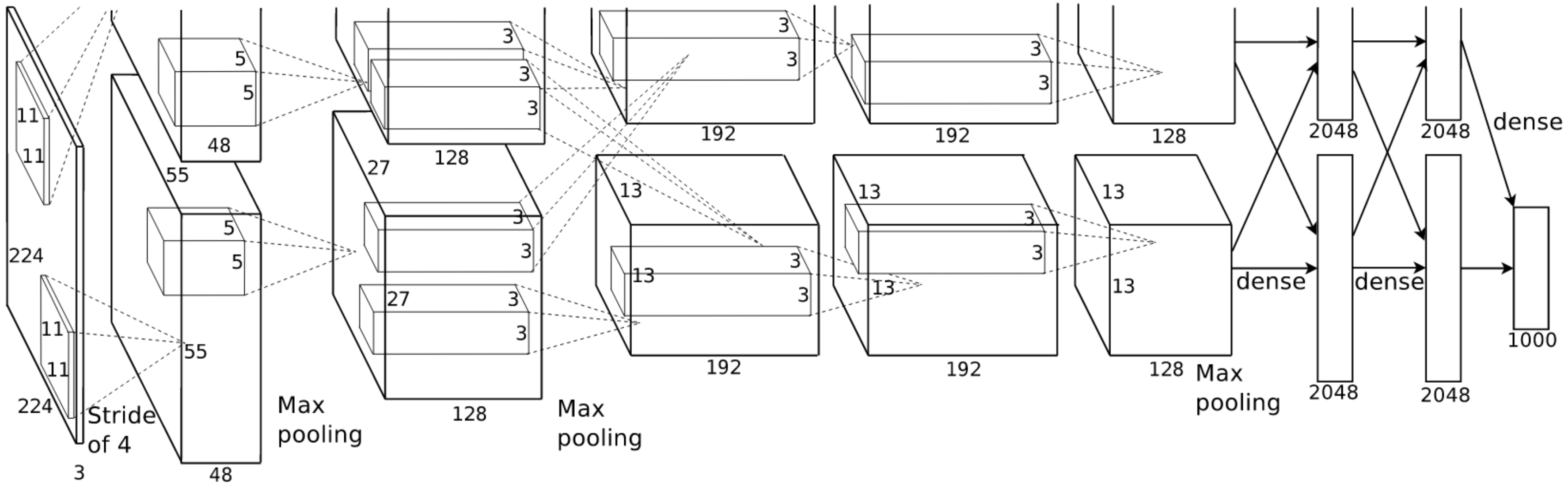
Large Scale Image Classification

- ImageNet
  - Over 15 million labeled high-resolution images
  - Roughly 22,000 categories
  - Collected from the web
  - Labeled by human labelers using Amazon's Mechanical Turk crowd-sourcing tool
- ImageNet Large-Scale Visual Recognition Challenge (ILSVRC)
  - Uses a subset of ImageNet
    - 1,000 categories
    - 1.2 million training images
    - 50,000 validation images
    - 150,000 test images
  - Report two error rates:
    - Top-1 and top-5

ImageNet Classification Results



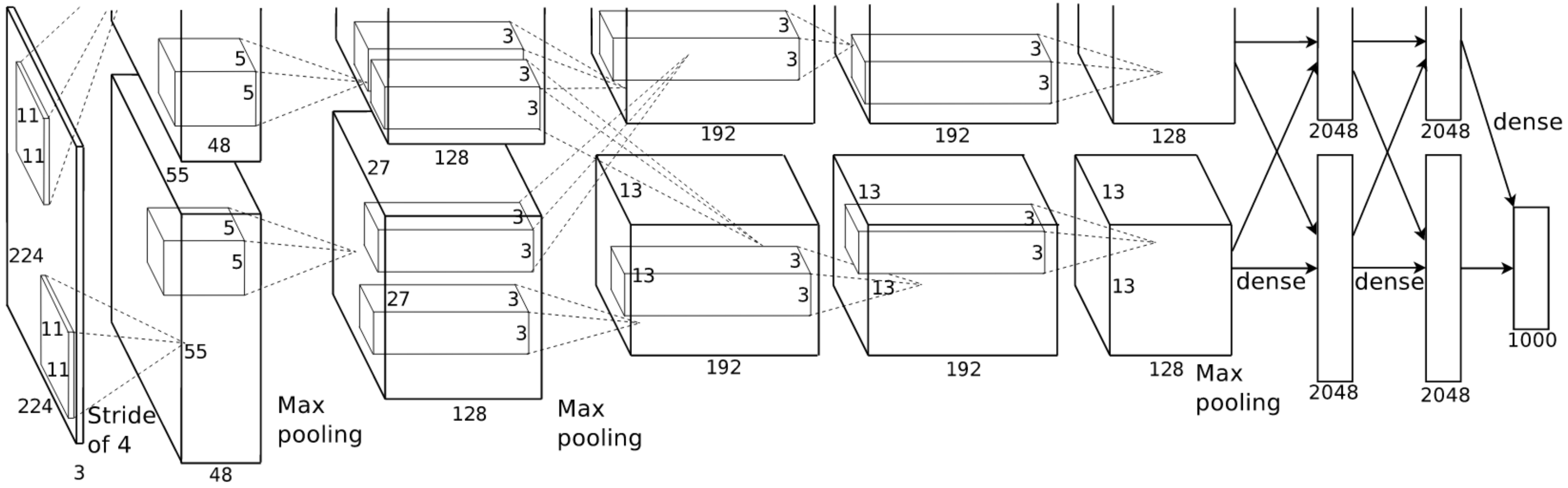
AlexNet (Krizhevsky, 2012)



- First use of ReLU
- Used norm layers(not common anymore)
- Data Augmentation
- Dropout 0.5
- Batch size 128
- SGD Momentum 0.9
- Learning rate 0.01, reduced by 10
- L2 weight decay 5e-4
- 7 CNN ensemble: 18.2% → 15.4%

Slide Credit : Stanford CS231n

AlexNet (Krizhevsky, 2012)



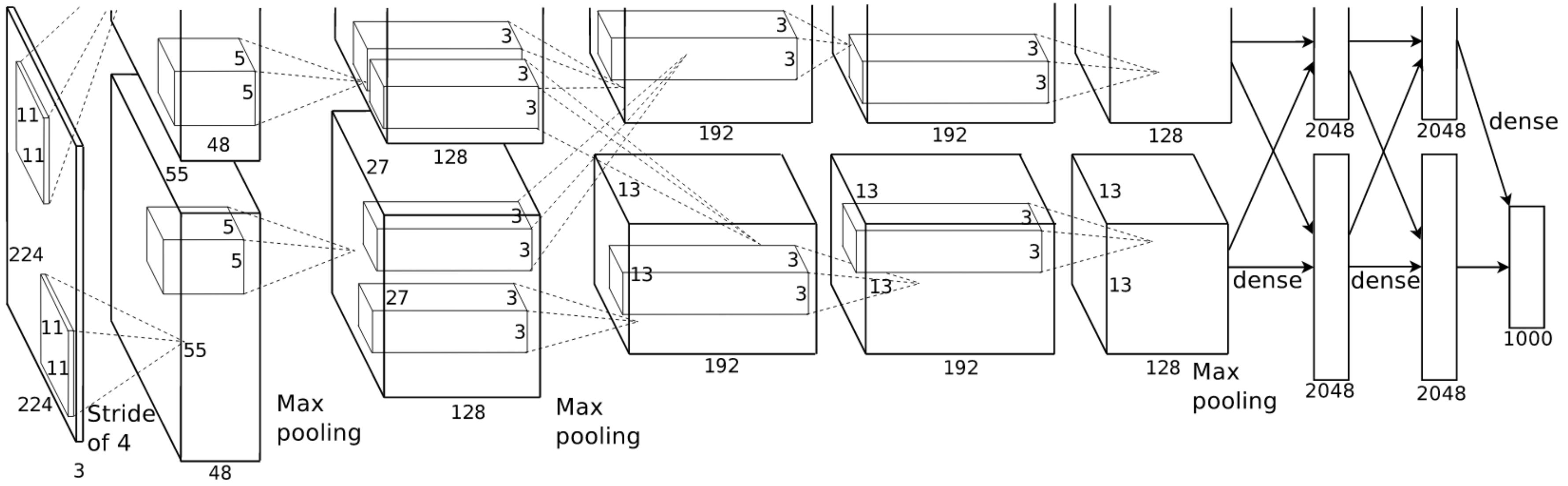
- Input: 227 x 227 x 3 images
- First layer(CONV1): 96 11x1 filters applied at stride 4

→Output size : 55 x 55 x 96

→# of parameters : (11 x 11 x 3) x 96 = 35K

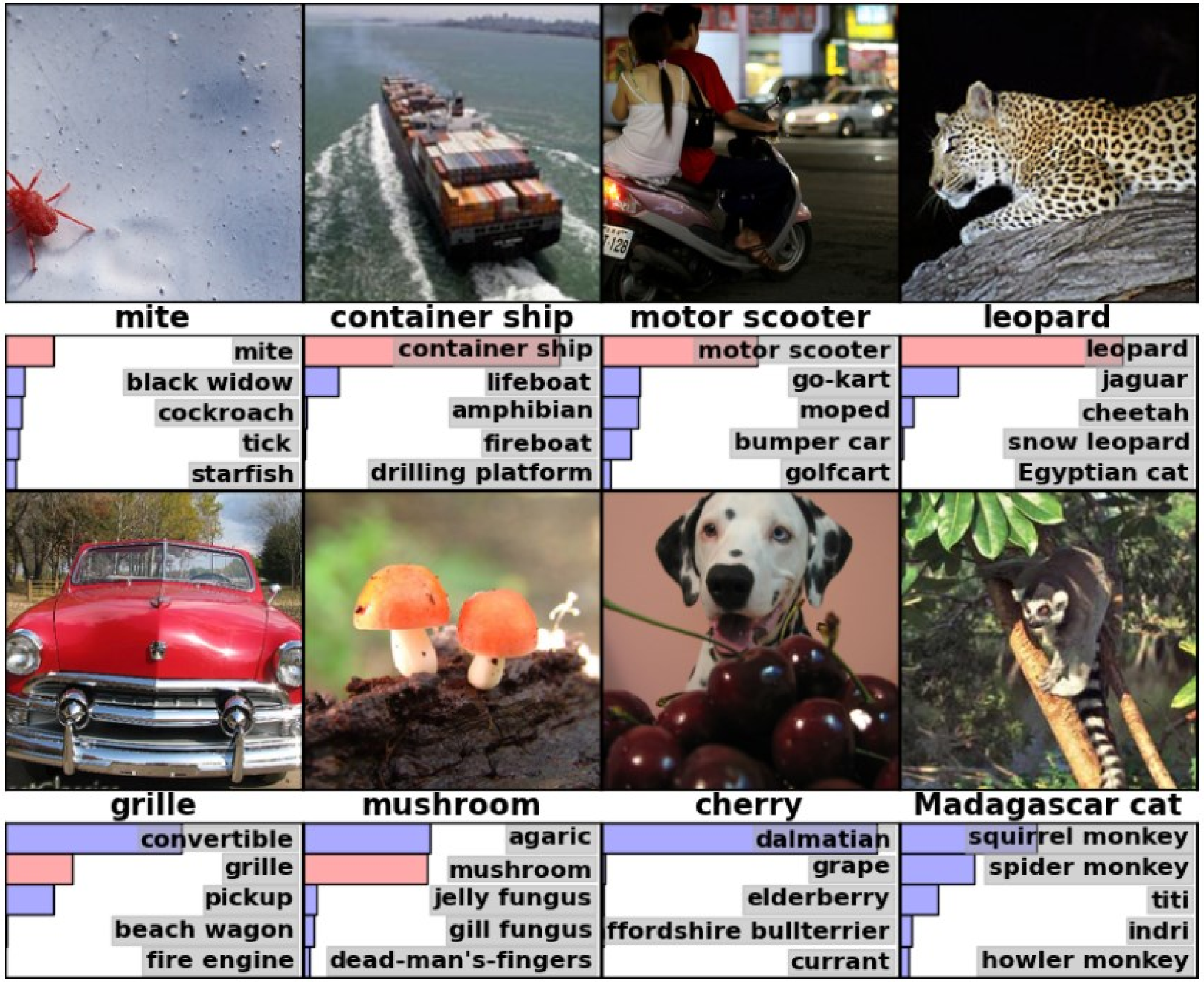
Slide Credit : Stanford CS231n

AlexNet (Krizhevsky, 2012)



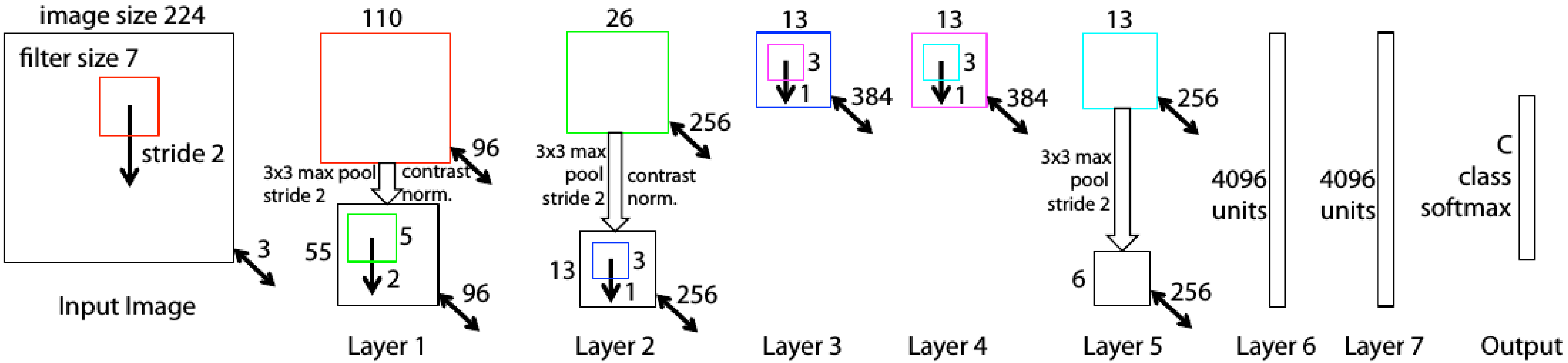
- [Conv1 – Pool1 – Norm1 – Conv2 – Pool2 – Norm2 – Conv3 – Conv4 – Conv5 – Pool 5 – FC6 – FC7 – FC8]
- 7 hidden layers, 650,000 neurons, 60M parameters
- Training for 1 week, using 2-GPUs
  - Trained on GTX 580 GPU with only 3GB of memory. Network spread across 2 GPUs, half the neurons(feature maps) on each GPU

AlexNet Result



"ImageNet Classification with Deep Convolutional Neural Networks "

ZFNet (Zeiler and Fergus, 2013)

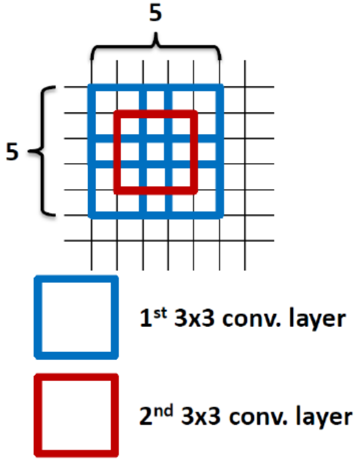


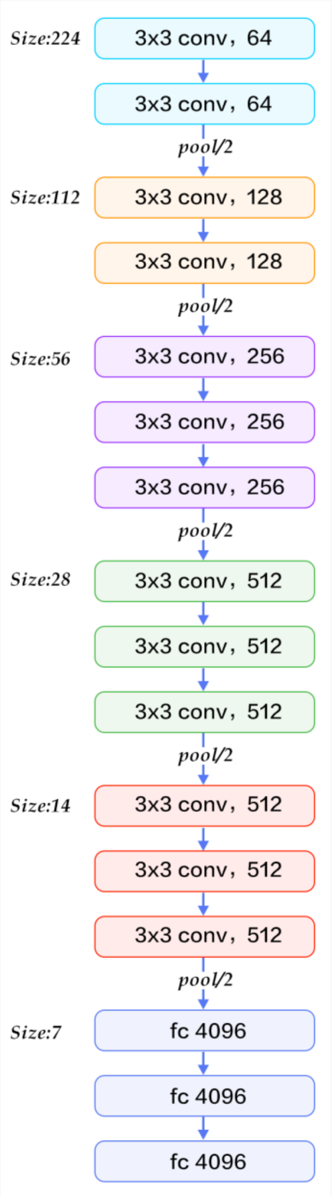
- Similar to AlexNet except:
  - Conv1 – change from (11x11 stride 4) to (7x7 stride 2)
  - Conv3, 4, 5: instead of 384, 384, 256 filters use 512, 1024, 512
  - Top 5 error : 16.4% → 11.7%

VGGNet (Simonyan and Zisserman, 2014)

- ILSVRC'14 2nd in classification, 1st in localization
- Small filters, Deeper networks
  - 8 layers(AlexNet) → 16~19 layers(VGG16, VGG19)
  - Only use 3x3 conv stride 1, pad 1 & 2x2 maxpool stride 2
  - 11.7% top 5 error(ZFNet) → 7.3% top 5 error
- Why use only 3x3 filters?
  - Stack of 3x3 conv layers has same effective receptive field as 5x5 or 7x7 conv layer
  - Deeper means more non-linearities
  - Fewer parameters: 2 x (3 x 3 x C) vs (5 x 5 x C)

    → regularization effect





VGGNet

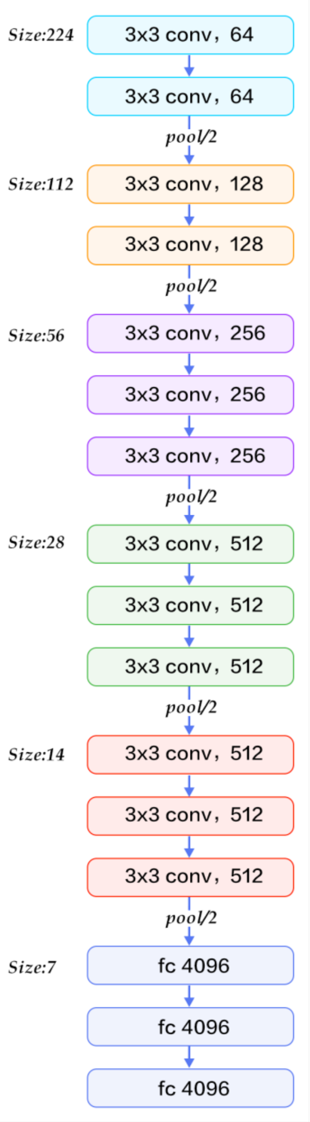
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Memory Usages and Parameters of VGGNet

(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

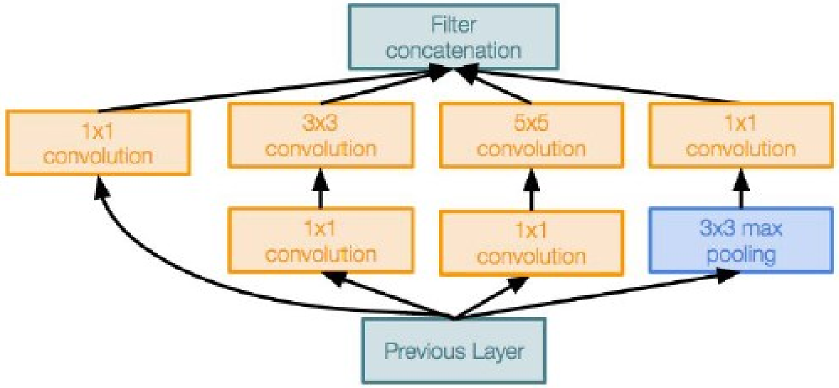
TOTAL memory: 24M * 4 bytes ~= 96MB / image (for a forward pass)
TOTAL params: 138M parameters



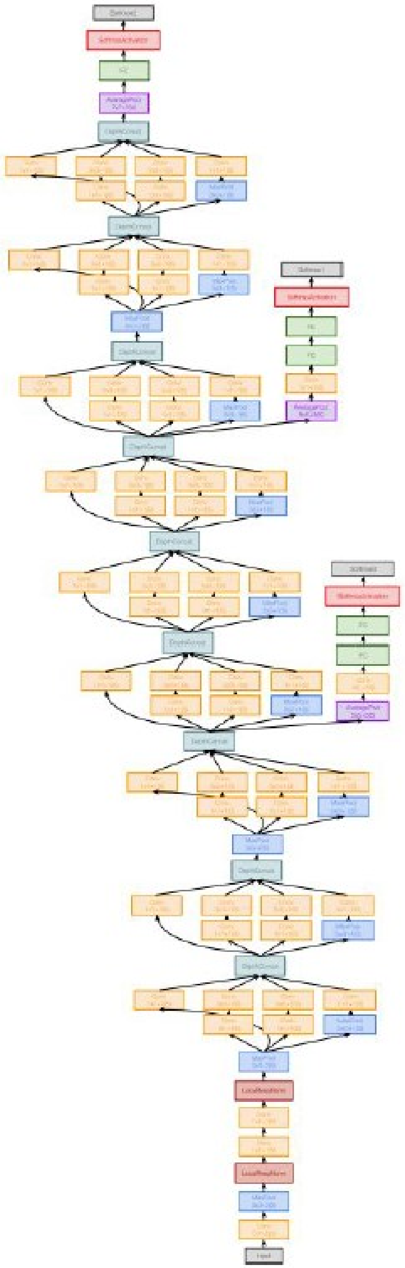
Slide Credit : Stanford CS231n

GoogLeNet (Szegedy, 2014)

- Deeper networks, with computational efficiency
  - 22 layers
  - Efficient "Inception" module
  - Global average pooling
  - Only 5 million parameters
  - ILSVRC'14 classification winner
    - 6.7% top 5 error



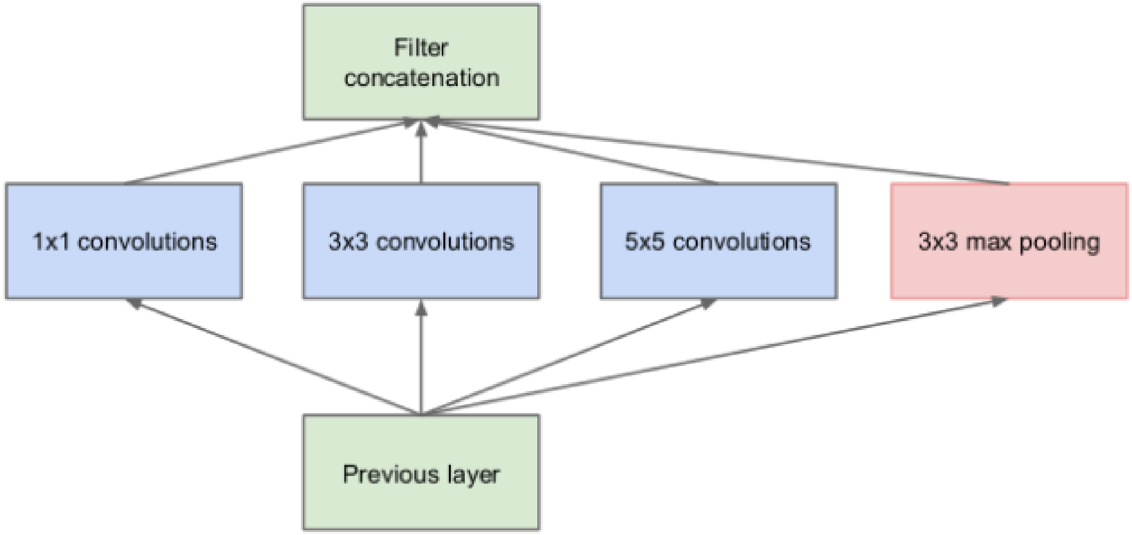
Inception module



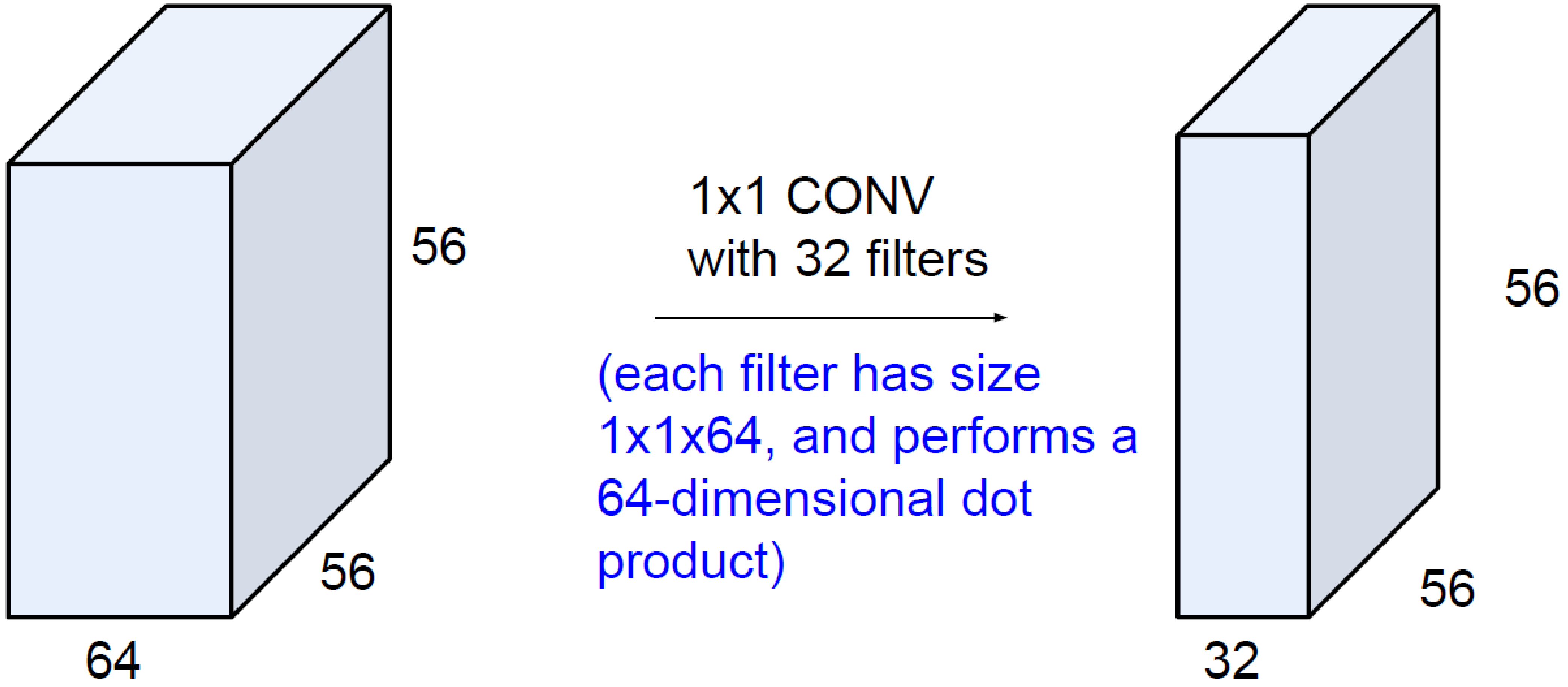
Slide Credit : Stanford CS231n

Inception Module

- Naïve Inception module
  - Apply parallel filter operations on the input from previous layer
    - ➢ Multiple receptive field sizes for convolution
    - ➢ Pooling operation(3x3)
  - Concatenate all filter outputs together channel-wise
  - What is the problem with this?
    - ➢ Very expensive compute
    - ➢ Depth after concatenation can grow and grow at every layer!
    - → Use Bottleneck layers

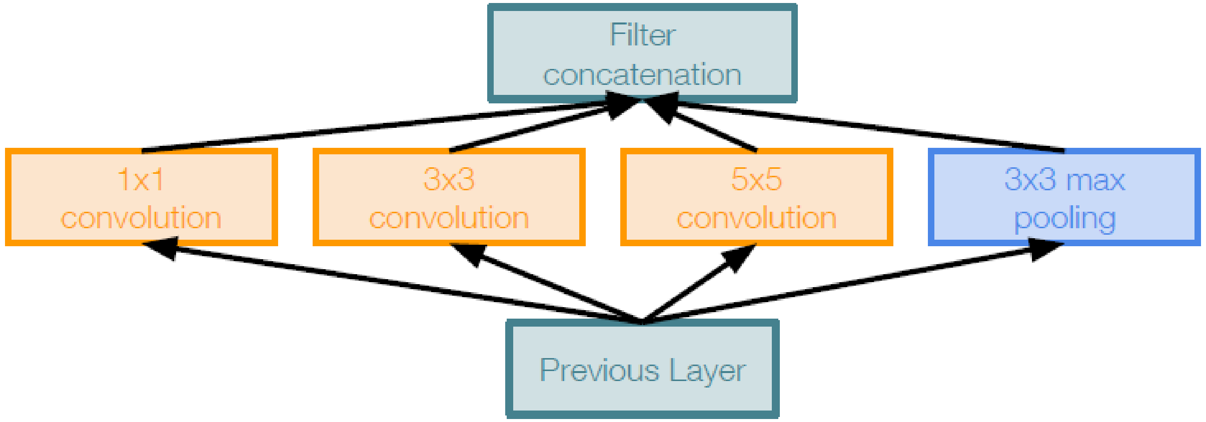


1x1 convolutions



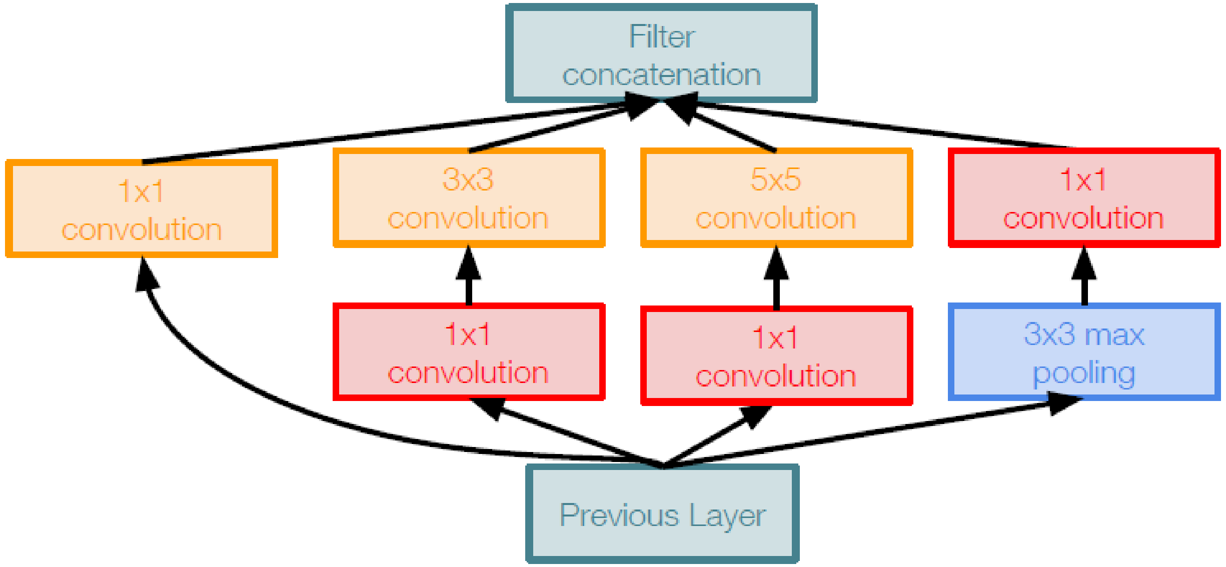
Slide Credit : Stanford CS231n

Inception Module



Naive Inception module

1x1 conv "bottleneck" layers

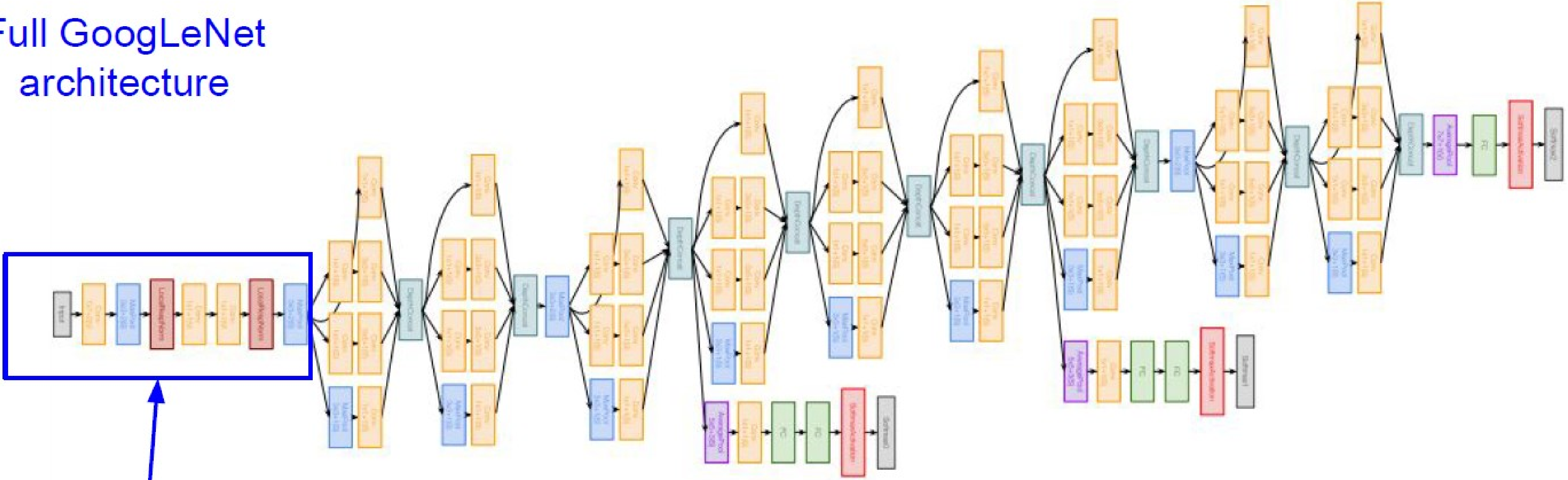


Inception module with dimension reduction

Slide Credit : Stanford CS231n

GoogLeNet

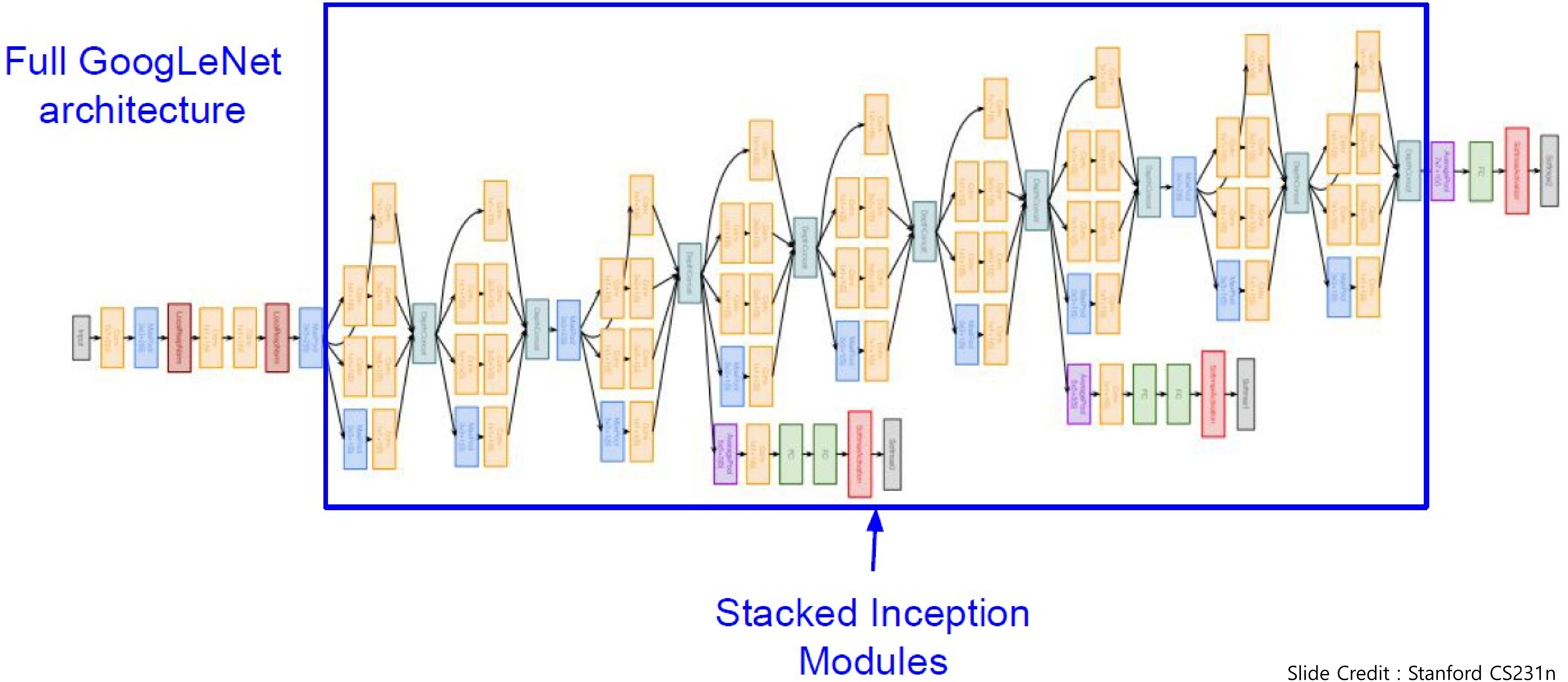
Full GoogLeNet architecture



Stem Network:
Conv-Pool-
2x Conv-Pool

Slide Credit : Stanford CS231n

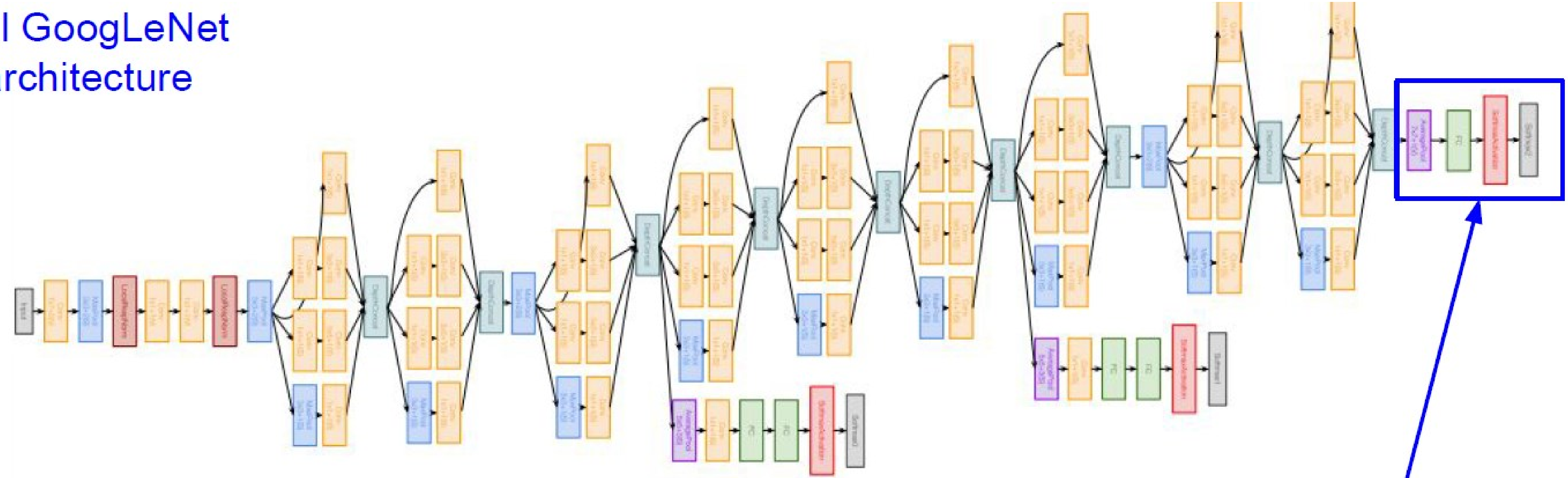
GoogLeNet



Slide Credit : Stanford CS231n

GoogLeNet

Full GoogLeNet architecture

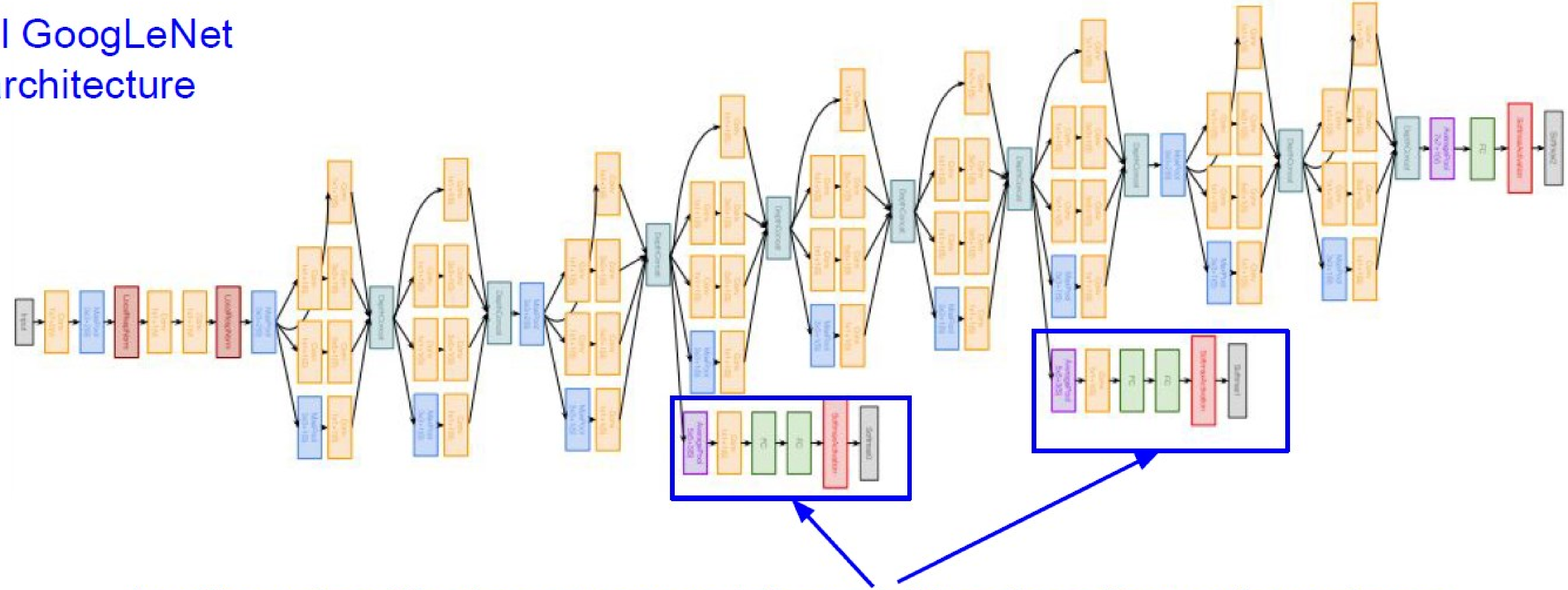


Classifier output

Slide Credit : Stanford CS231n

GoogLeNet

Full GoogLeNet architecture



Auxiliary classification outputs to inject additional gradient at lower layers
(AvgPool-1x1Conv-FC-FC-Softmax)

Slide Credit : Stanford CS231n

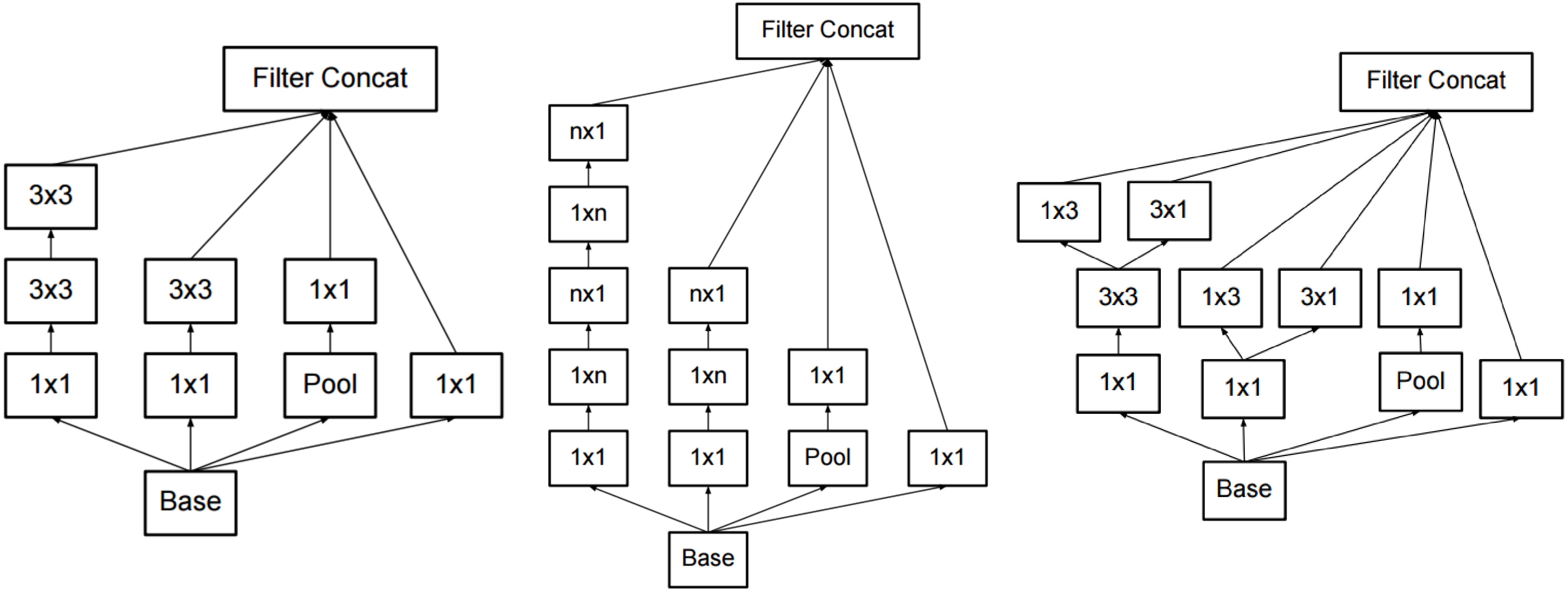
GoogLeNet

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Table 1: GoogLeNet incarnation of the Inception architecture

Inception-v3

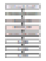
- Factorization of filters



ResNet (He, 2015)

- Revolution of Depth

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)

ResNet, 152 layers
(ILSVRC 2015)

ResNet

- Swept 1st place in all ILSVRC and COCO 2015 competitions

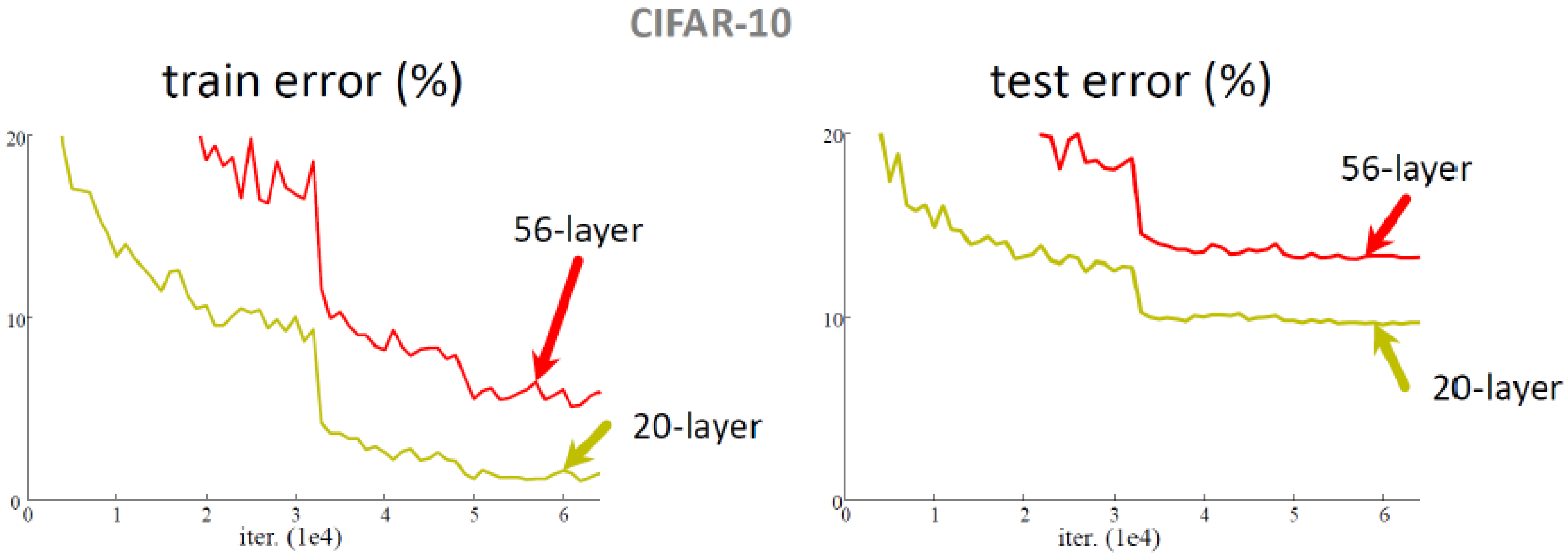
MSRA @ ILSVRC & COCO 2015 Competitions

- **1st places in all five main tracks**
  - ImageNet Classification: *"Ultra-deep"* (quote Yann) 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

- ILSVRC'15 classification winner(3.6% top 5 error) – better than "human performance" (Russakovsky 2014)

ResNet

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

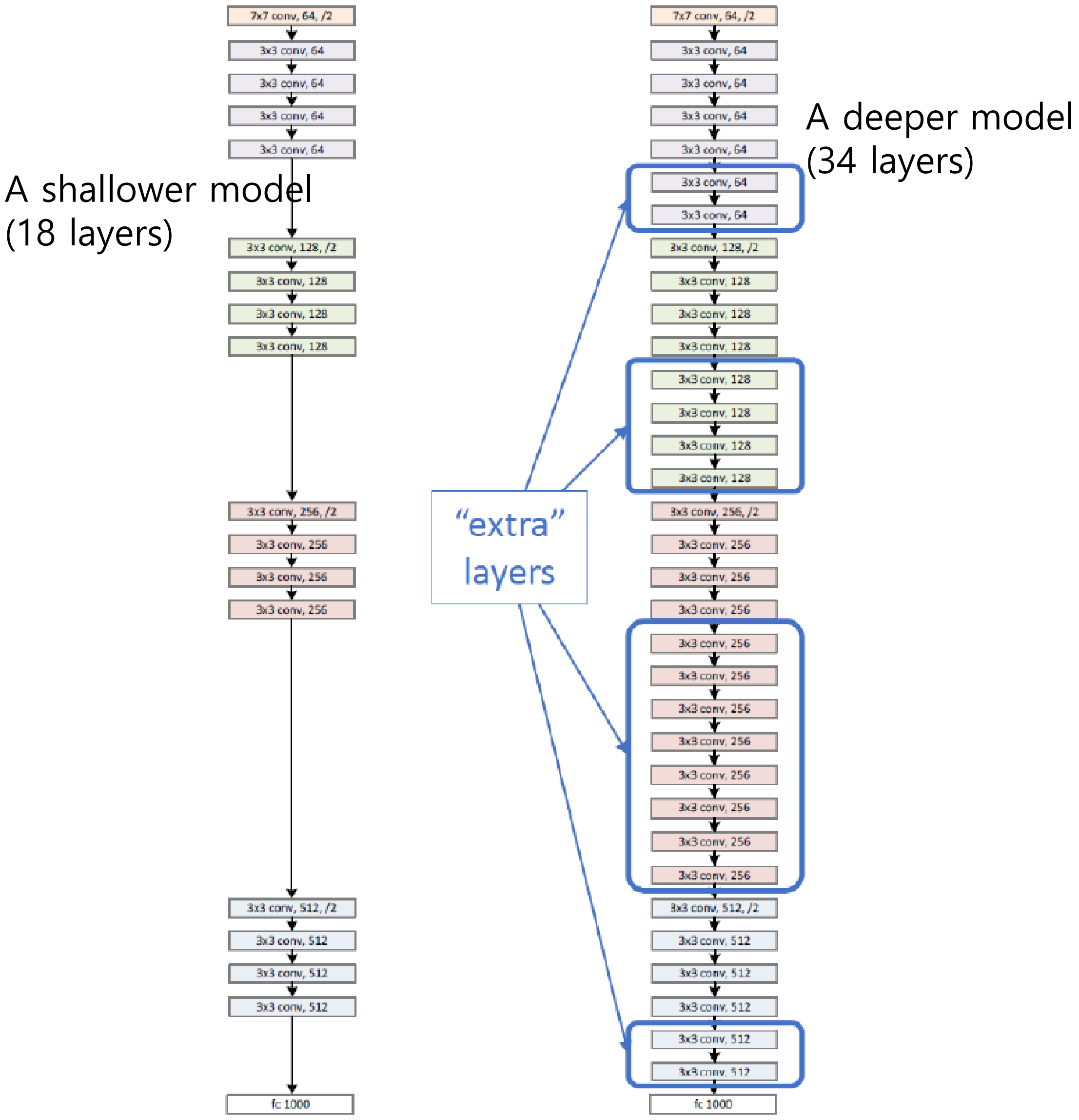


- 56-layer model performs worse on both training and test error
  - The deeper model performs worse, but it's not caused by overfitting!

Slide Credit : Stanford CS231n

ResNet

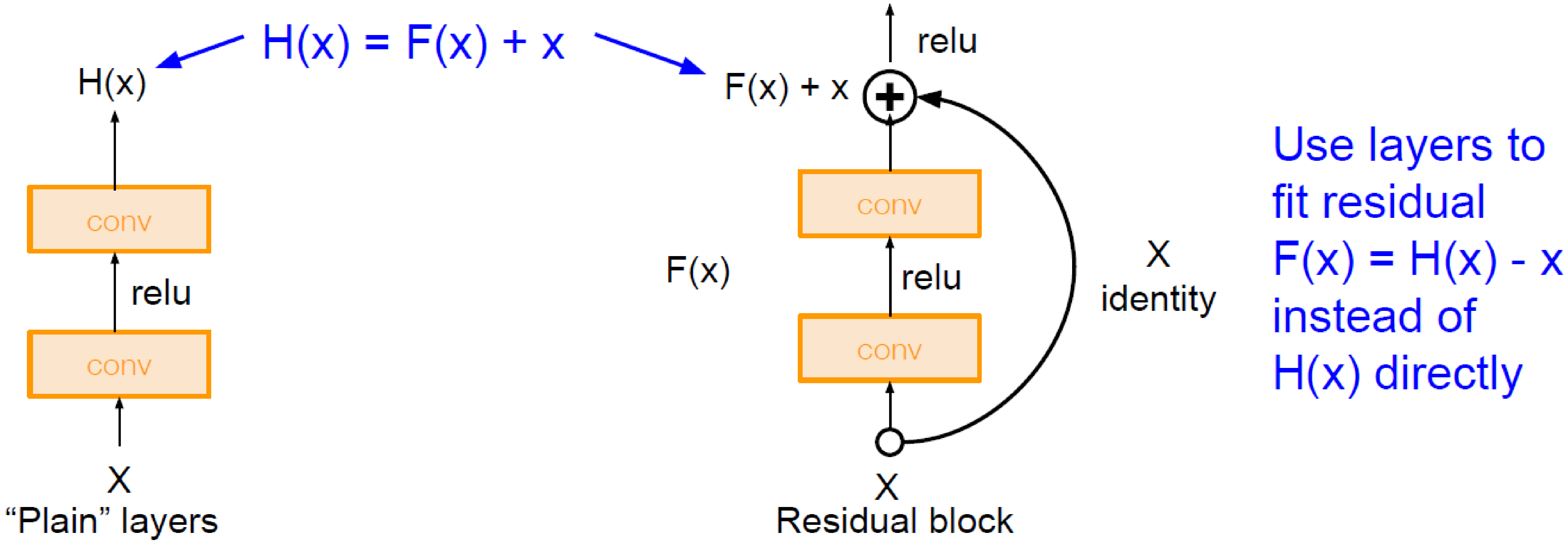
- Hypothesis: the problem is an *optimization* problem, deeper models are harder to optimize
  - The deeper model should be able to perform at least as well as the shallower model
  - A solution by construction is copying the learned layers from the shallower model and setting additional layers to identity mapping



Slide Credit : Stanford CS231n

ResNet

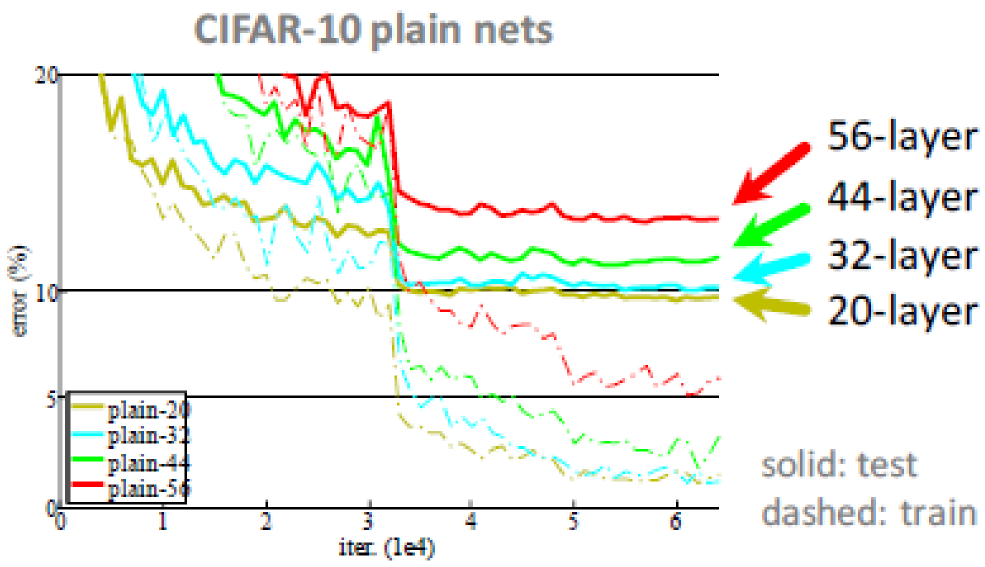
- Solution: Use network layers to fit a residual mapping instead of directly trying to fit a desired underlying mapping

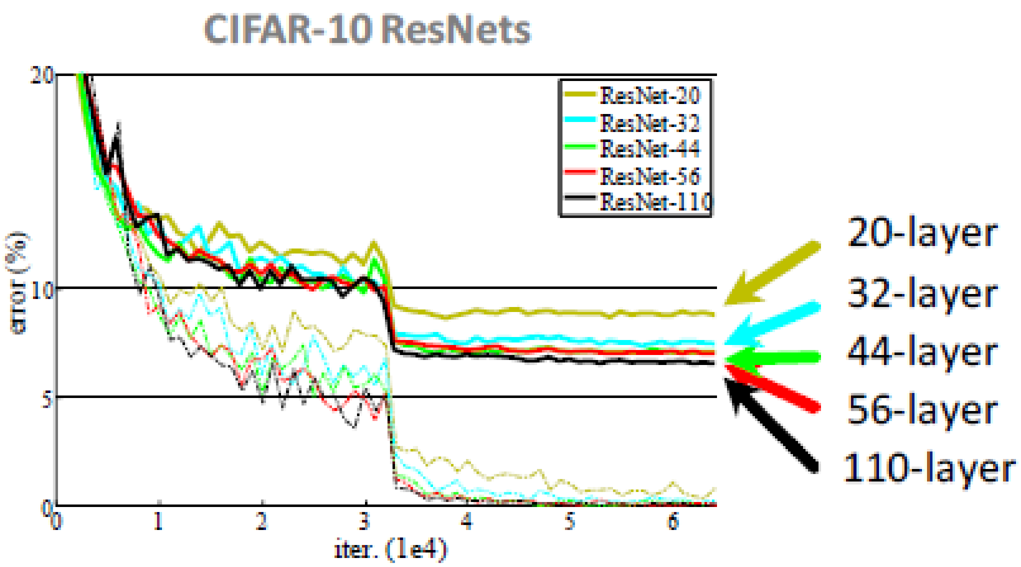


Slide Credit : Stanford CS231n

ResNet – Experimental Results

CIFAR-10 experiments

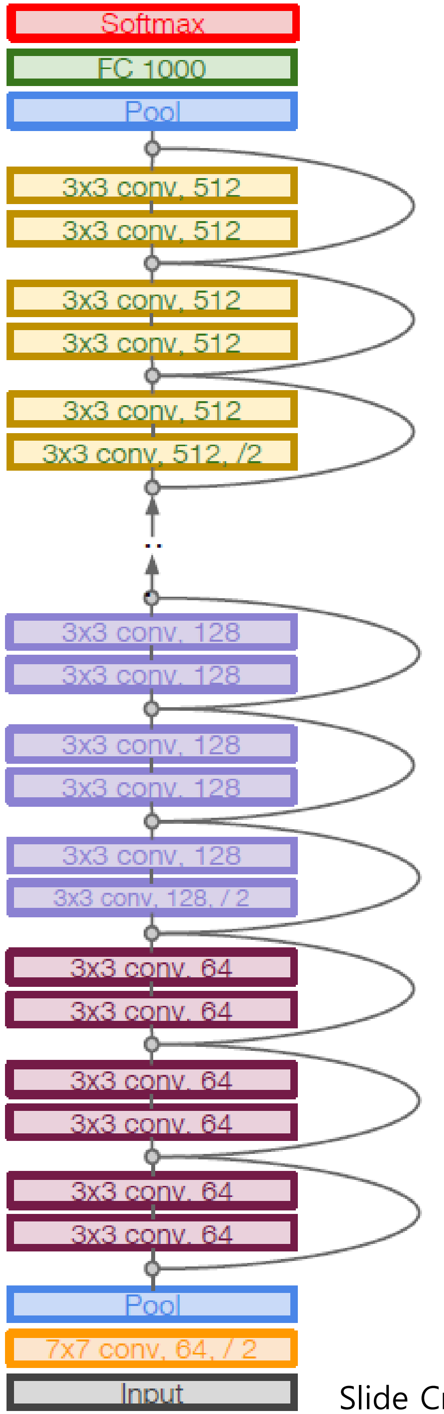




- Deep ResNets can be trained without difficulties
- Deeper ResNets have **lower training error**, and also lower test error

ResNet

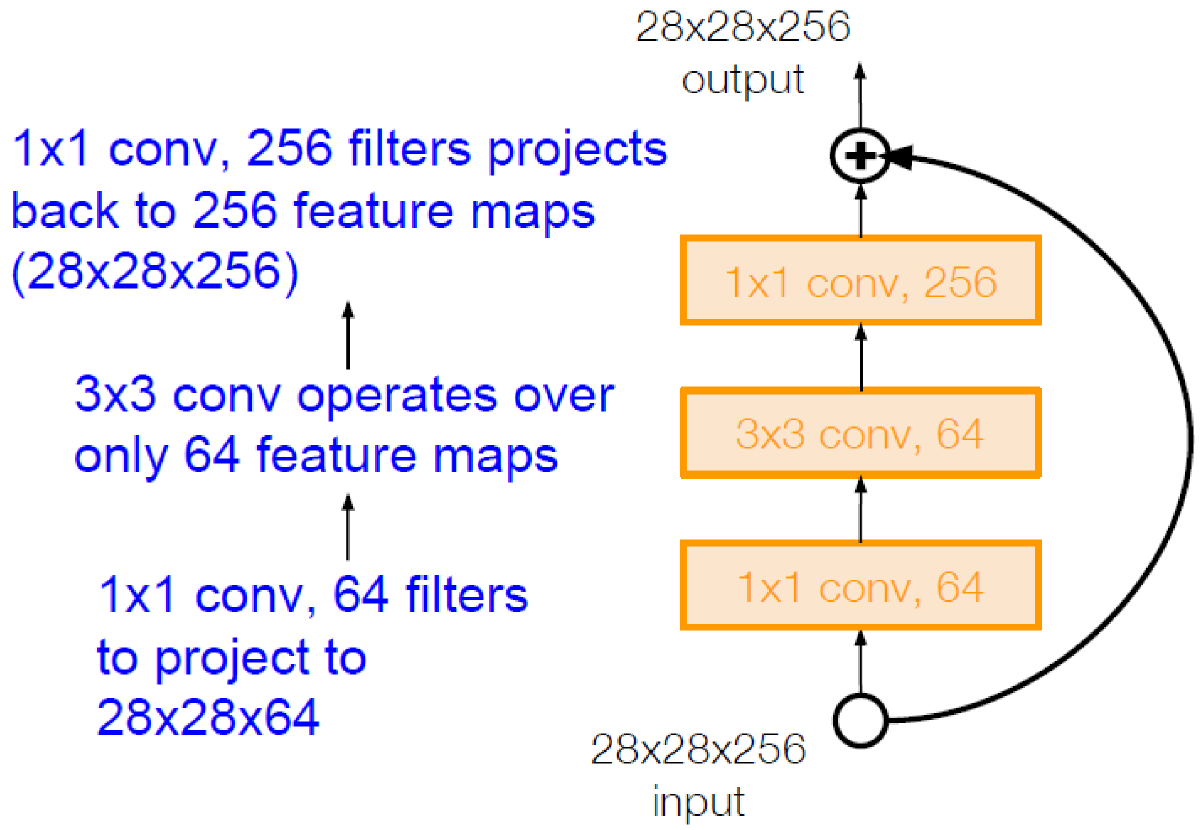
- Full ResNet architecture
  - Stack residual blocks
  - Every residual block has two 3x3 conv layers
  - Periodically double # of filters and downsample spatially using stride 2
  - Additional conv layer at the beginning
  - No FC layers at the end(only FC 100 to output classes)



Slide Credit : Stanford CS231n

Bottleneck Architecture

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



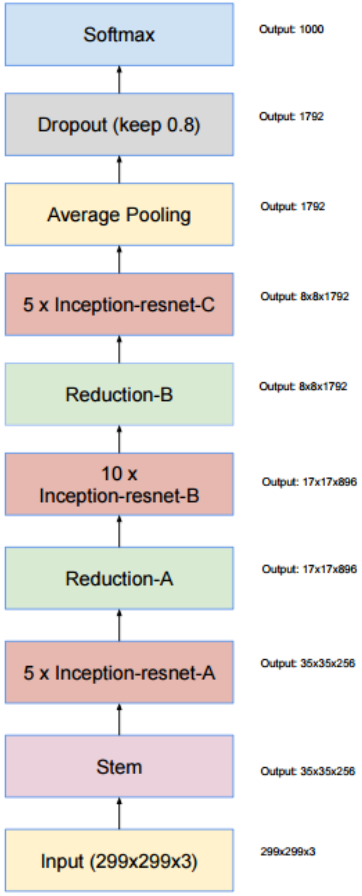
Slide Credit : Stanford CS231n

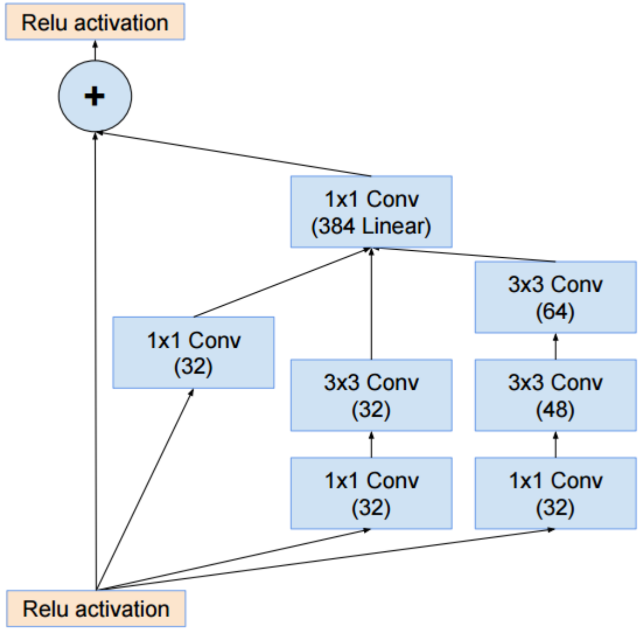
ResNet

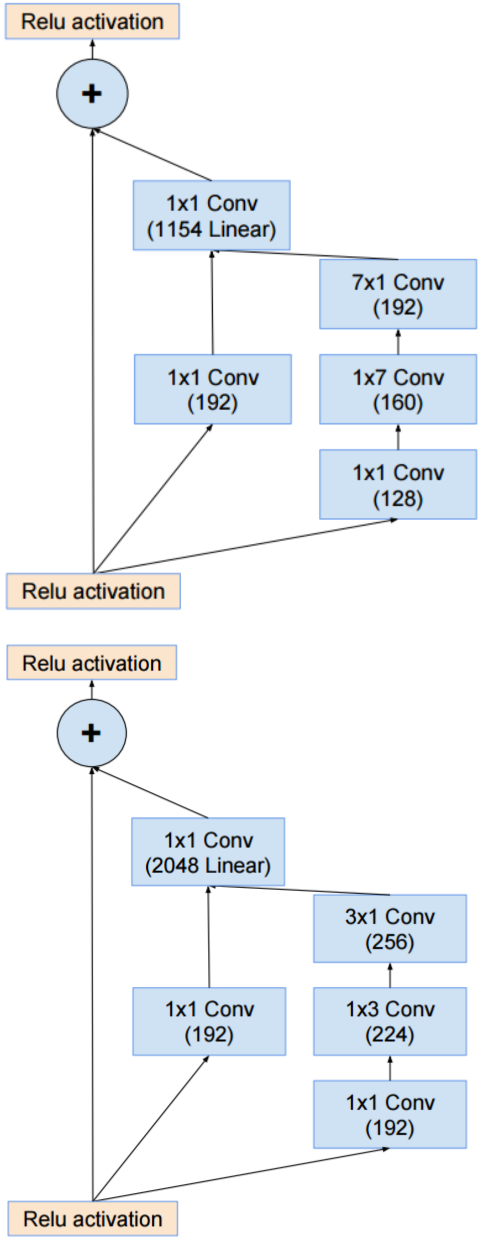
- Batch Normalization after every Conv layer
- Xavier/2 initialization from He et al.
- SGD + Momentum(0.9)
- Learning rate: 0.1, divided by 10 when validation error plateaus
- Mini-batch size 256
- Weight decay of 1e-5
- No dropout used

Inception-ResNet

- Inception + ResNet

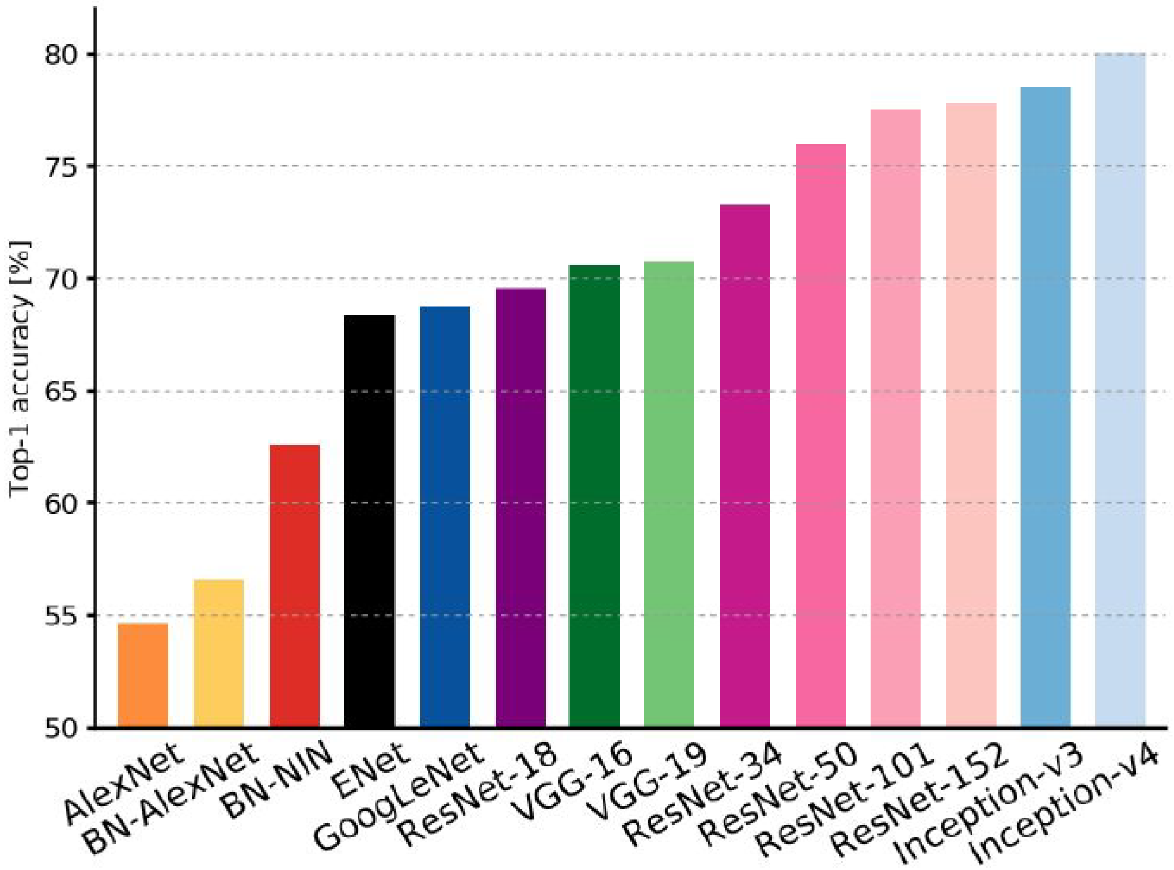


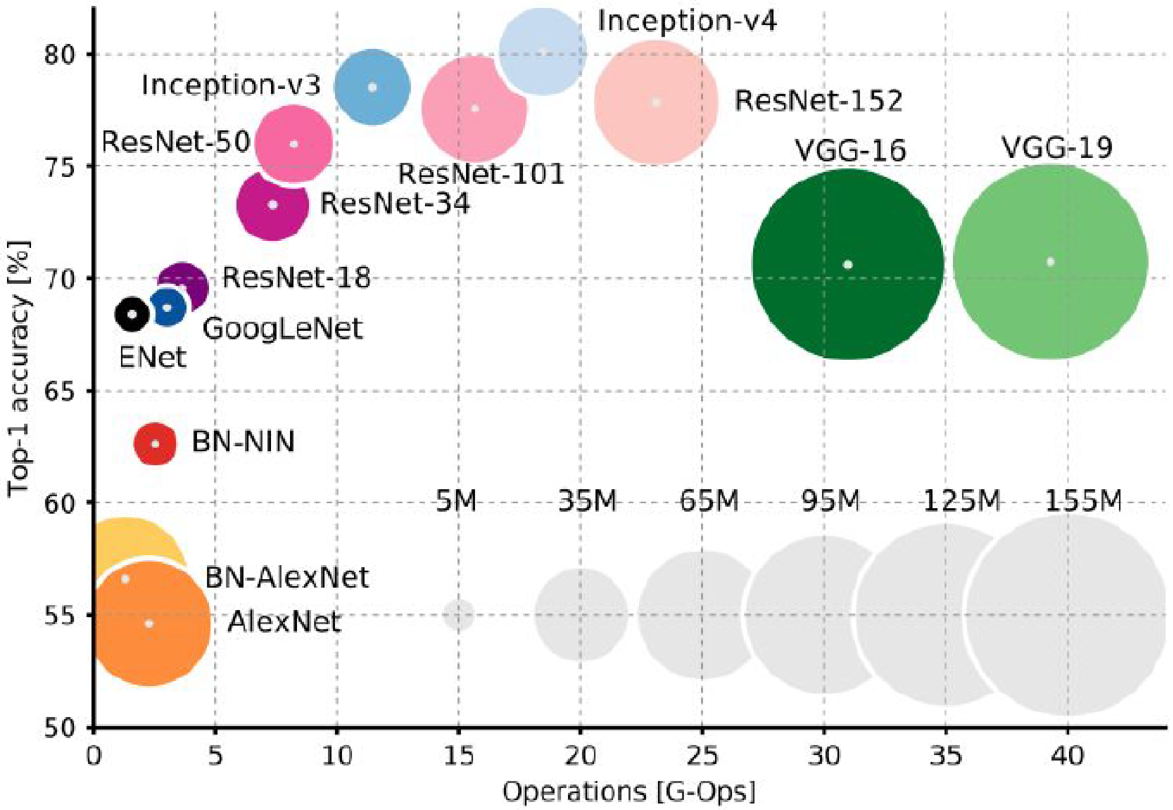




"Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning"

Comparing Complexity





An Analysis of Deep Neural Network Models for Practical Applications, 2017.

Slide Credit : Stanford CS231n

2016 ILSVRC Classification Result

Team name	Entry description	**Classification error**	Localization error
Trimps-Soushen	Ensemble 2	0.02991	0.077668
Trimps-Soushen	Ensemble 3	0.02991	0.077087
Trimps-Soushen	Ensemble 4	0.02991	0.077429
ResNeXt	Ensemble C, weighted average, tuned on val. [No bounding box results]	0.03031	0.737308
CU-DeepLink	GrandUnion + Fused-scale EnsembleNet	0.03042	0.098892
CU-DeepLink	GrandUnion + Multi-scale EnsembleNet	0.03046	0.099006
CU-DeepLink	GrandUnion + Basic Ensemble	0.03049	0.098954
ResNeXt	Ensemble B, weighted average, tuned on val. [No bounding box results]	0.03092	0.737484
CU-DeepLink	GrandUnion + Class-reweighted Ensemble	0.03096	0.099369
CU-DeepLink	GrandUnion + Class-reweighted Ensemble with Per-instance Normalization	0.03103	0.099349
ResNeXt	Ensemble C, weighted average. [No bounding box results]	0.03124	0.737526
Trimps-Soushen	Ensemble 1	0.03144	0.079068
ResNeXt	Ensemble A, simple average. [No bounding box results]	0.0315	0.737505
SamExynos	3 model only for classification	0.03171	0.236561
ResNeXt	Ensemble B, weighted average. [No bounding box results]	0.03203	0.737681
KAISTNIA_ETRI	Ensembles A	0.03256	0.102015
KAISTNIA_ETRI	Ensembles C	0.03256	0.102056
KAISTNIA_ETRI	Ensembles B	0.03256	0.100676
DeepIST	EnsembleC	0.03291	1.0
DeepIST	EnsembleD	0.03294	1.0
DGIST-KAIST	Weighted sum #1 (five models)	0.03297	0.489969
DGIST-KAIST	Weighted sum #2 (five models)	0.03324	1.0
NUIST	prefer multi class prediction	0.03351	0.094058
KAISTNIA_ETRI	Ensembles A (further tuned in class-dependent model I)	0.03352	0.100552
KAISTNIA_ETRI	Ensembles B (further tuned in class-dependent models I)	0.03352	0.099286
DGIST-KAIST	Averaging five models	0.03357	1.0
DGIST-KAIST	Averaging six models	0.03357	1.0
DGIST-KAIST	Averaging four models	0.03378	0.490373

2017 ILSVRC Classification Result

- http://image-net.org/challenges/LSVRC/2017/results

Team name	Entry description	**Classification error**	Localization error
WMW	Ensemble C [No bounding box results]	0.02251	0.590987
WMW	Ensemble E [No bounding box results]	0.02258	0.591018
WMW	Ensemble A [No bounding box results]	0.0227	0.591153
WMW	Ensemble D [No bounding box results]	0.0227	0.591039
WMW	Ensemble B [No bounding box results]	0.0227	0.59106
Trimps-Soushen	Result-1	0.02481	0.067698
Trimps-Soushen	Result-2	0.02481	0.06525
Trimps-Soushen	Result-3	0.02481	0.064991
Trimps-Soushen	Result-4	0.02481	0.065261
Trimps-Soushen	Result-5	0.02481	0.065302
NUS-Qihoo_DPNs (CLS-LOC)	[E2] CLS:: Dual Path Networks + Basic Ensemble	0.0274	0.088093
NUS-Qihoo_DPNs (CLS-LOC)	[E1] CLS:: Dual Path Networks + Basic Ensemble	0.02744	0.088269
BDAT	provide_class	0.02962	0.086942
BDAT	provide_box	0.03158	0.081392
MIL_UT	Ensemble of 9 models (classification-only)	0.03205	0.596164
SIIT_KAIST-SKT	ensemble 2	0.03226	0.128924