



M2177.003100
Deep Learning

[5: Convolutional Neural Nets (Part 2)]

Electrical and Computer Engineering
Seoul National University

© 2018 Sungroh Yoon. this material is for educational uses only. some contents are based on the material provided by other paper/book authors and may be copyrighted by them.

(last compiled at 12:23:00 on 2018/10/03)

Outline

Background

CNN Architectures

AlexNet

VGG

GoogLeNet

ResNet

Recent Architectures

Summary

References

- *Deep Learning* by Goodfellow, Bengio and Courville [▶ Link](#)
 - ▶ Chapter 9
- online resources:
 - ▶ *Deep Learning Specialization (coursera)* [▶ Link](#)
 - ▶ *Stanford CS231n: CNN for Visual Recognition* [▶ Link](#)
 - ▶ *Machine Learning Yearning* [▶ Link](#)

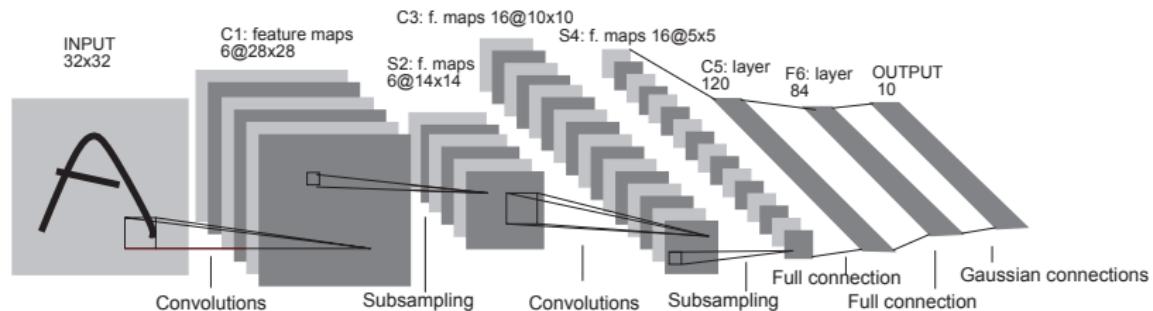
Outline

Background

Summary

CNN Architectures

Classic: LeNet-5

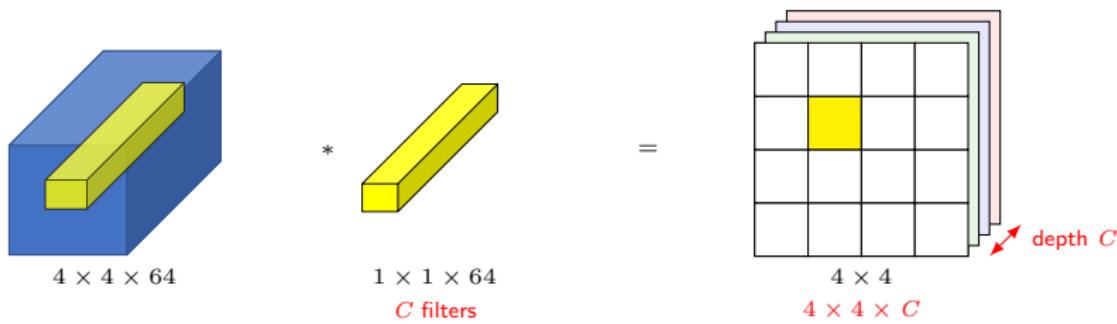


(source: LeCun, 1998)

- CONV-POOL-CONV-POOL-FC-FC
 - ▶ 5×5 conv filters (stride 1)
 - ▶ 2×2 pooling layers (stride 2)

1×1 convolution on volumes

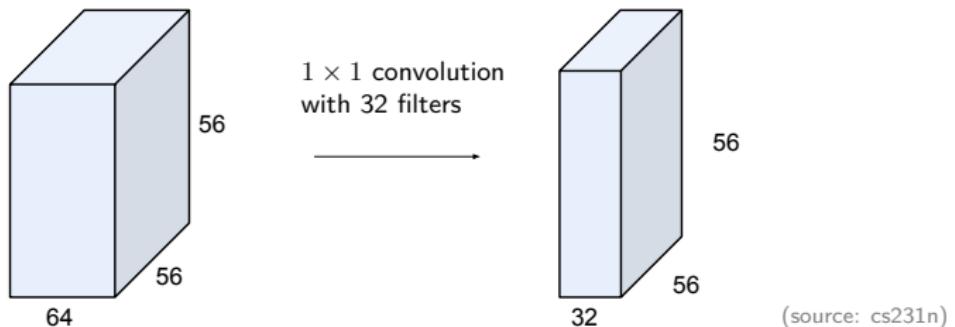
$$\begin{array}{|c|c|c|c|} \hline 6 & 2 & 1 & 1 \\ \hline 9 & 4 & 5 & 6 \\ \hline 2 & 7 & 5 & 3 \\ \hline 8 & 2 & 3 & 5 \\ \hline \end{array}
 \quad *
 \quad \boxed{2}
 \quad =
 \quad
 \begin{array}{|c|c|c|c|} \hline 12 & 4 & 2 & 2 \\ \hline 18 & 8 & 10 & 12 \\ \hline 4 & 14 & 10 & 6 \\ \hline 16 & 4 & 6 & 10 \\ \hline \end{array}$$



(source: coursera)

- nonlinearity (e.g. ReLU) can follow 1×1 convolution → “network in network”

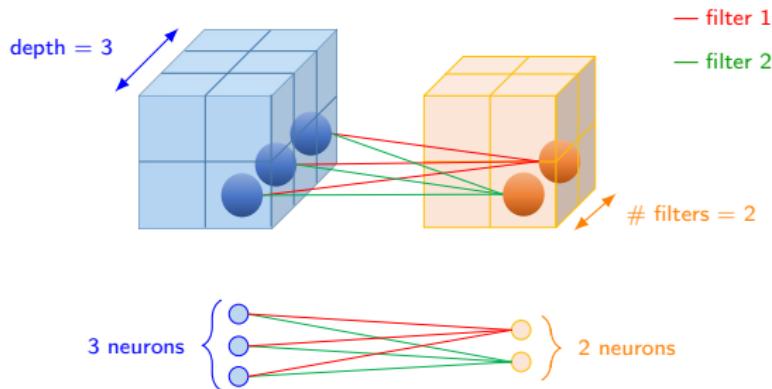
- 1×1 convolution: widely used for depth adjustment
- example:



- ▶ each filter: size $1 \times 1 \times 64$ (performs a 64-dim dot product)
- ▶ preserves spatial dimensions and reduces depth
- ▶ projects depth to lower dimension (combination of feature maps)
- in general
 - ▶ we can reduce/maintain/increase depth using 1×1 convolution

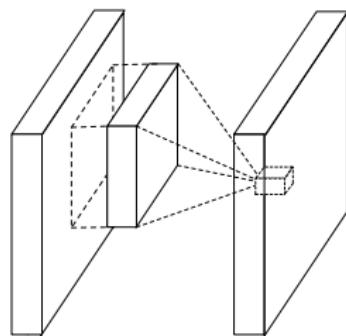
- a set of 1×1 conv filters: can be interpreted as forming an FC layer
 - ▶ **input dimension** of this FC layer
= depth of the input volume to 1×1 conv filters
 - ▶ **output dimension** of this FC layer
= number of 1×1 conv filters

- example: $2 \times 2 \times 3$ volume applied to two $1 \times 1 \times 3$ filters
 - ▶ $2 \times 2 = 4$ instances of an FC layer, which maps 3 neurons to 2 neurons

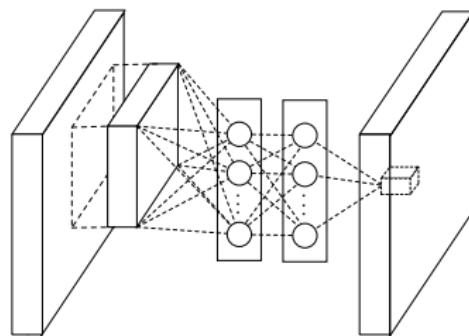


Network in network

- Mlpconv layer with $\overbrace{\text{micro network}}$ within each conv layer
 - composed of FC layer (with 1×1 conv) + nonlinearity
 - ▶ can compute more **abstract features** for local patches
 - ▶ precursor to GoogLeNet and ResNet “bottleneck” layers



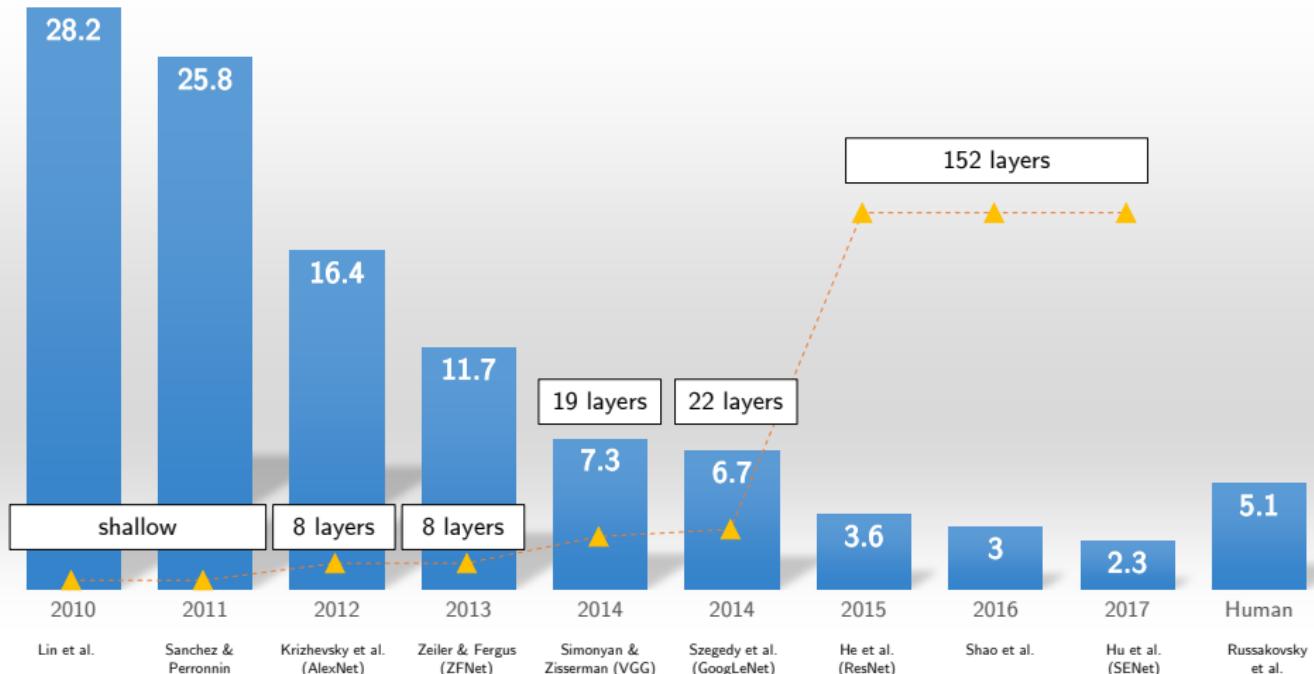
(a) linear convolution layer



(b) Mlpconv layer

(source: Lin et al., 2014)

ImageNet challenge winners



Outline

Background

CNN Architectures

AlexNet

VGG

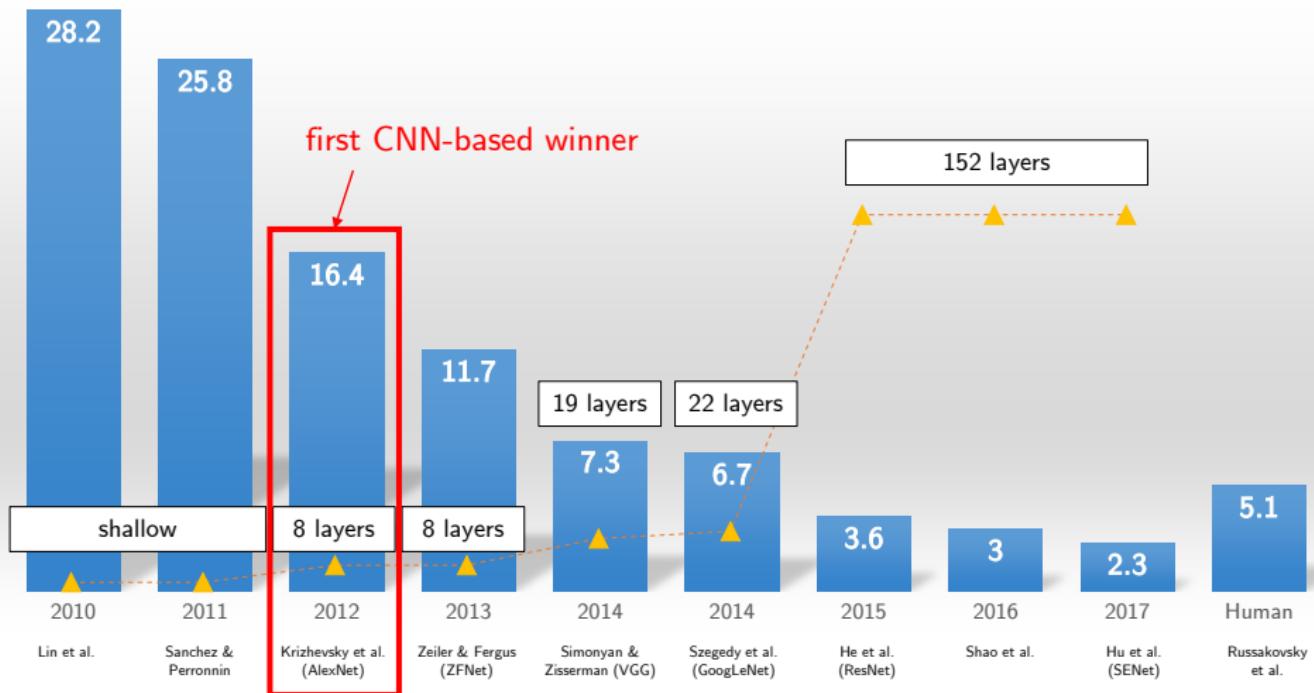
GoogLeNet

ResNet

Recent Architectures

Summary

ImageNet challenge winners

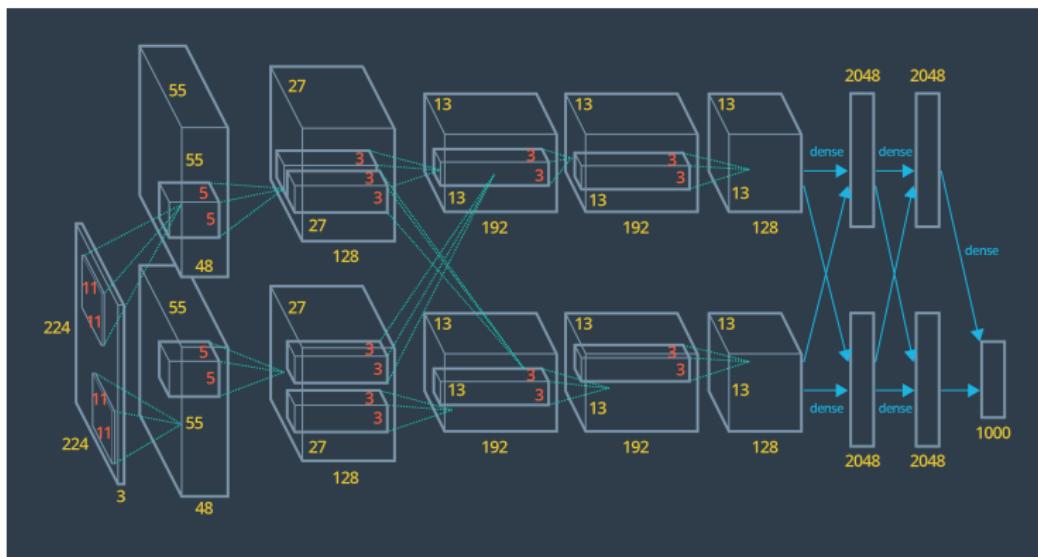


AlexNet

- Alex Krizhevsky, Ilya Sutskever, Geoffrey Hinton (2012)

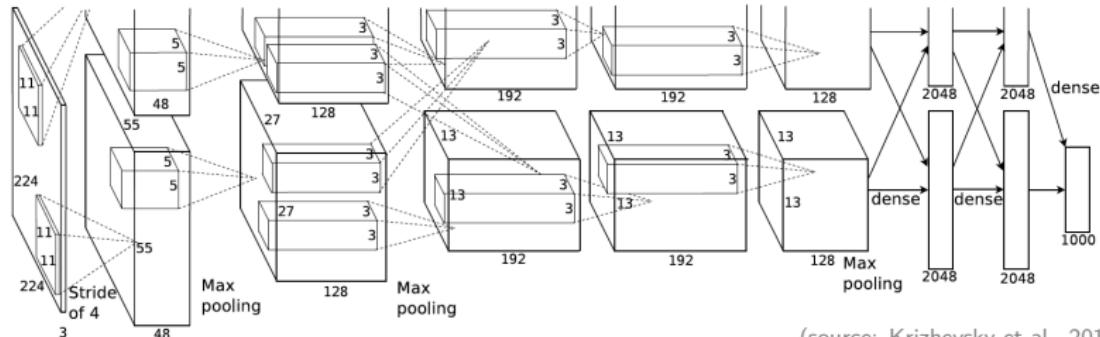
- ▶ ILSVRC 2012 winner

원래 1개로 되는데 그래픽 카드가 후져서 2개로 나누어서 수행했음



(source: yuchao.us)

Architecture



(source: Krizhevsky et al., 2012)

227x227x3	INPUT
55x55x96	CONV1
27x27x96	MAX POOL1
27x27x96	NORM1
27x27x256	CONV2
13x13x256	MAX POOL2
13x13x256	NORM2
13x13x384	CONV3
13x13x384	CONV4
13x13x256	CONV5
6x6x256	MAX POOL3
4096	FC6
4096	FC7
1000	FC8

96 11x11 filters at stride 4, pad 0
3x3 filters at stride 2
normalization layer
256 5x5 filters at stride 1, pad 2
3x3 filters at stride 2
normalization layer
384 3x3 filters at stride 1, pad 1
384 3x3 filters at stride 1, pad 1
256 3x3 filters at stride 1, pad 1
3x3 filters at stride 2
4096 neurons
4096 neurons
1000 neurons (class scores)

- total number of parameters

▶ 60M

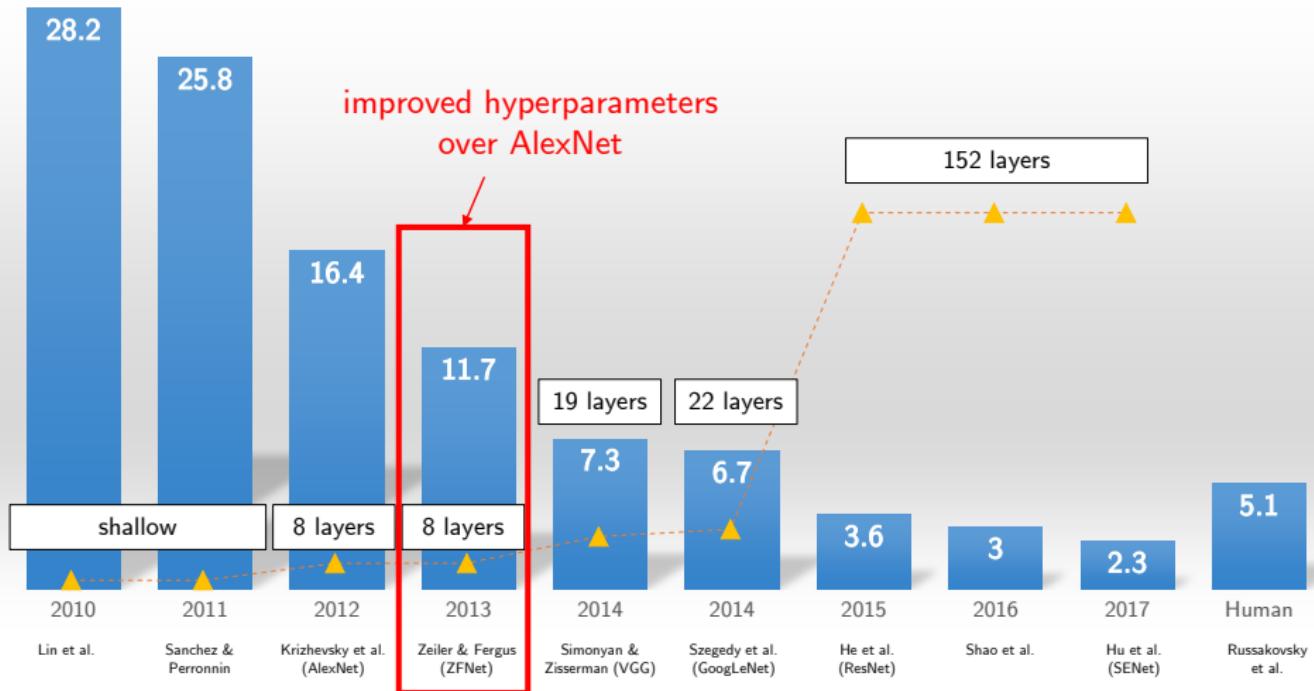
- details:

- ▶ first use of ReLU
- ▶ used normalization (NORM) layers (not common anymore) -> 현재는 batch norm 이용
- ▶ heavy data augmentation
- ▶ dropout: 0.5
- ▶ batch size: 128
- ▶ SGD + momentum (0.9)
- ▶ learning rate: 10^{-2}
(reduced by 10 manually when validation accuracy plateaus)
- ▶ L2 weight decay: 5×10^{-4}
- ▶ 7 CNN ensemble: 18.2% → 15.4%

- trained on GTX 580 GPU (only 3GB memory)

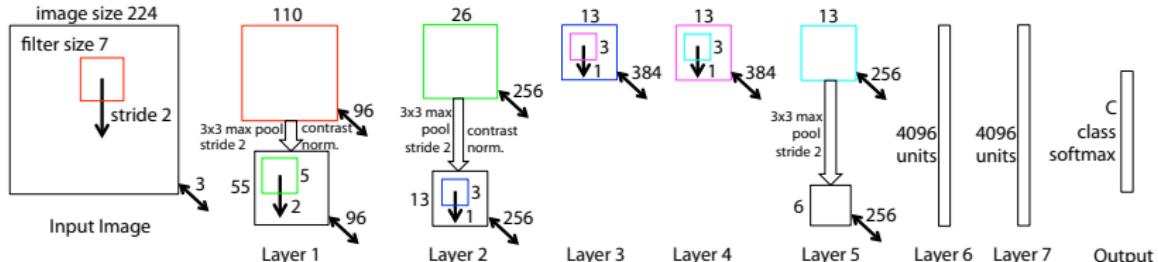
- ▶ network spread across 2 GPUs

ImageNet challenge winners



ZFNet (Zeiler and Fergus, 2013)

hyperparameter 튜닝 외의 의미는 없음



(source: Zeiler and Fergus, 2013)

- the same as AlexNet but
 - ▶ CONV1: change from (11x11 stride 4) to (7x7 stride 2)
 - ▶ CONV3, 4, 5: instead of 384, 384, 256 filters use 512, 1024, 512
 - ▶ ImageNet top 5 error: 16.4% → 11.7%

Outline

Background

CNN Architectures

AlexNet

VGG

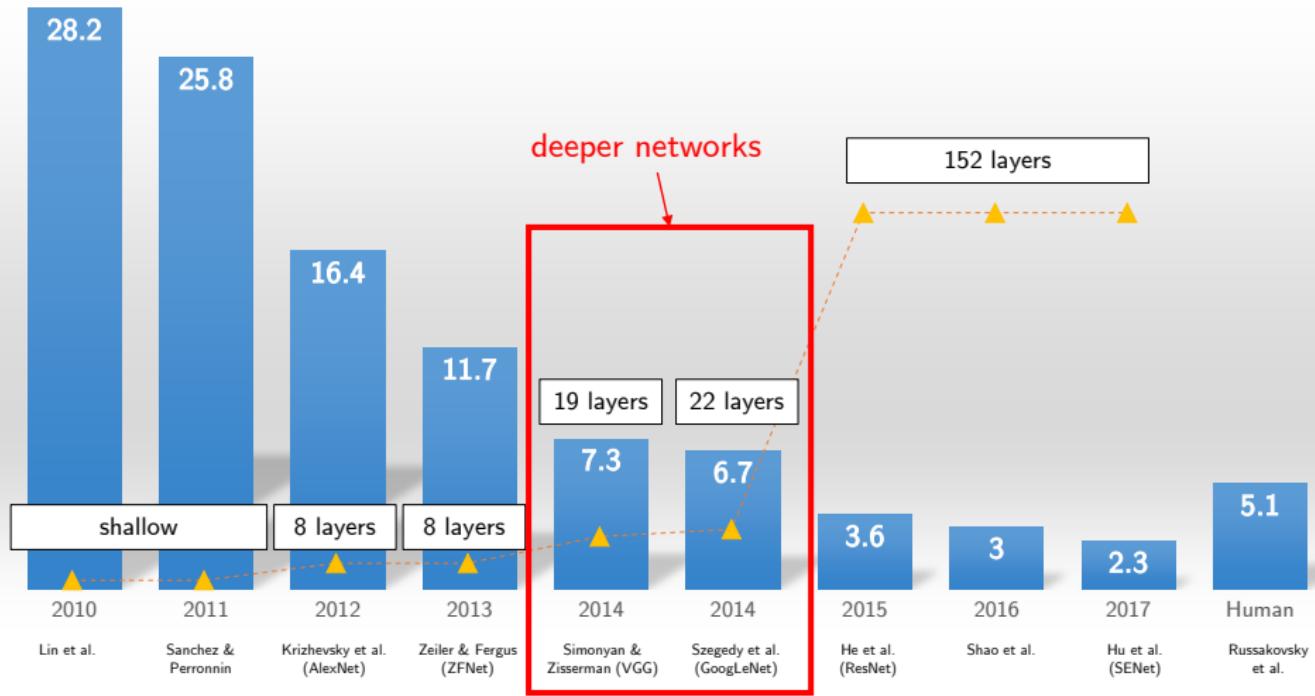
GoogLeNet

ResNet

Recent Architectures

Summary

ImageNet challenge winners



- Simonyan and Zisserman (2014)
- key idea: small filters, deeper networks

▶ only

3x3 CONV stride 1, pad 1

2x2 MAX POOL stride 2

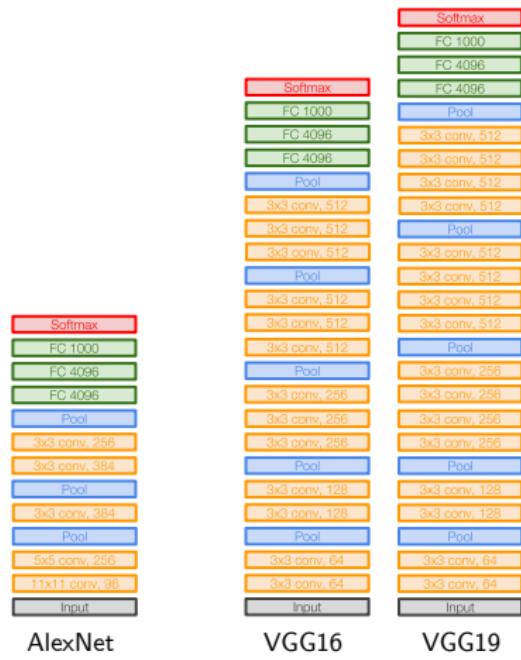
- ILSVRC top 5 error

▶ 11.7% (ZFNet, 2013)

→ 7.3% (VGG, 2014)

- two versions: VGG16, VGG19

▶ VGG19 only slightly better
(use more memory)



(source: cs231n)

Why use smaller filters?

- consider stacking three 3x3 conv (stride 1) layers

- benefits

- ▶ its effective receptive field
= that of one 7x7 conv layer

- ▶ but deeper
- ⇒ more non-linearities
- ▶ and fewer parameters¹:

$$3 \times (3^2 C^2) \text{ vs } 7^2 C^2$$

7x7 필터 1겹 == 3x3 필터 3겹(size의 측면에서)
but 더 deep 하기 때문에 non-linearity에서 좋고
parameter 숫자도 더 적음

$$7 \times 7 \Rightarrow 49, 3 \times 3 \times 3 = 27$$



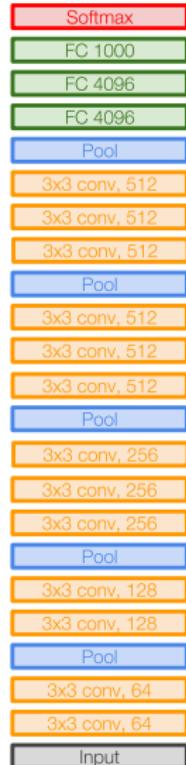
(source: cs231n)

¹assuming C channels per layer and C filters per layer

Architecture (VGG16)

parameter 꽤 많음 (AlexNet이 10M개였음)

layer type	dimension	memory (96MB/image)	parameters (138M total)
INPUT	224x224x3	224*224*3=150K	0
CONV3-64	224x224x64	224*224*64=3.2M	(3*3*3)*64 = 1,728
CONV3-64	224x224x64	224*224*64=3.2M	(3*3*64)*64 = 36,864
POOL2	112x112x64	112*112*64=800K	0
CONV3-128	112x112x128	112*112*128=1.6M	(3*3*64)*128 = 73,728
CONV3-128	112x112x128	112*112*128=1.6M	(3*3*128)*128 = 147,456
POOL2	56x56x128	56*56*128=400K	0
CONV3-256	56x56x256	56*56*256=800K	(3*3*128)*256 = 294,912
CONV3-256	56x56x256	56*56*256=800K	(3*3*256)*256 = 589,824
CONV3-256	56x56x256	56*56*256=800K	(3*3*256)*256 = 589,824
POOL2	28x28x256	28*28*256=200K	0
CONV3-512	28x28x512	28*28*512=400K	(3*3*256)*512 = 1,179,648
CONV3-512	28x28x512	28*28*512=400K	(3*3*512)*512 = 2,359,296
CONV3-512	28x28x512	28*28*512=400K	(3*3*512)*512 = 2,359,296
POOL2	14x14x512	14*14*512=100K	0
CONV3-512	14x14x512	14*14*512=100K	(3*3*512)*512 = 2,359,296
CONV3-512	14x14x512	14*14*512=100K	(3*3*512)*512 = 2,359,296
CONV3-512	14x14x512	14*14*512=100K	(3*3*512)*512 = 2,359,296
POOL2	7x7x512	7*7*512=25K	0
FC	1x1x4096	4096	7*7*512*4096 = 102,760,448
FC	1x1x4096	4096	4096*4096 = 16,777,216
FC	1x1x1000	1000	4096*1000 = 4,096,000



VGG16

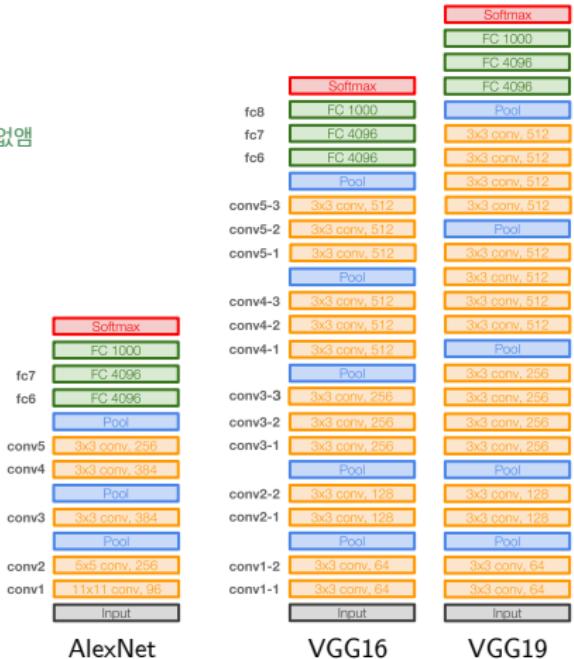
- ▶ most **memory**: in early _____
 - ▶ most **parameters**: in late ^{fc} layer

- ILSVRC'14 ranking: 2nd in classification, 1st in localization

- details:

- ▶ similar training procedure as AlexNet
- ▶ no local response normalization layer 없음 normalization (LRN)
- ▶ use **ensembles** for best results
- ▶ **FC7 features** generalize well to other tasks

FC7 layer가 generalize가 잘되서 train이 끝난 후 이 layer만 빼서 FC8 layer(classifier)만 바꿔서 쓸 수 있음



(source: cs231n)

Outline

Background

CNN Architectures

AlexNet

VGG

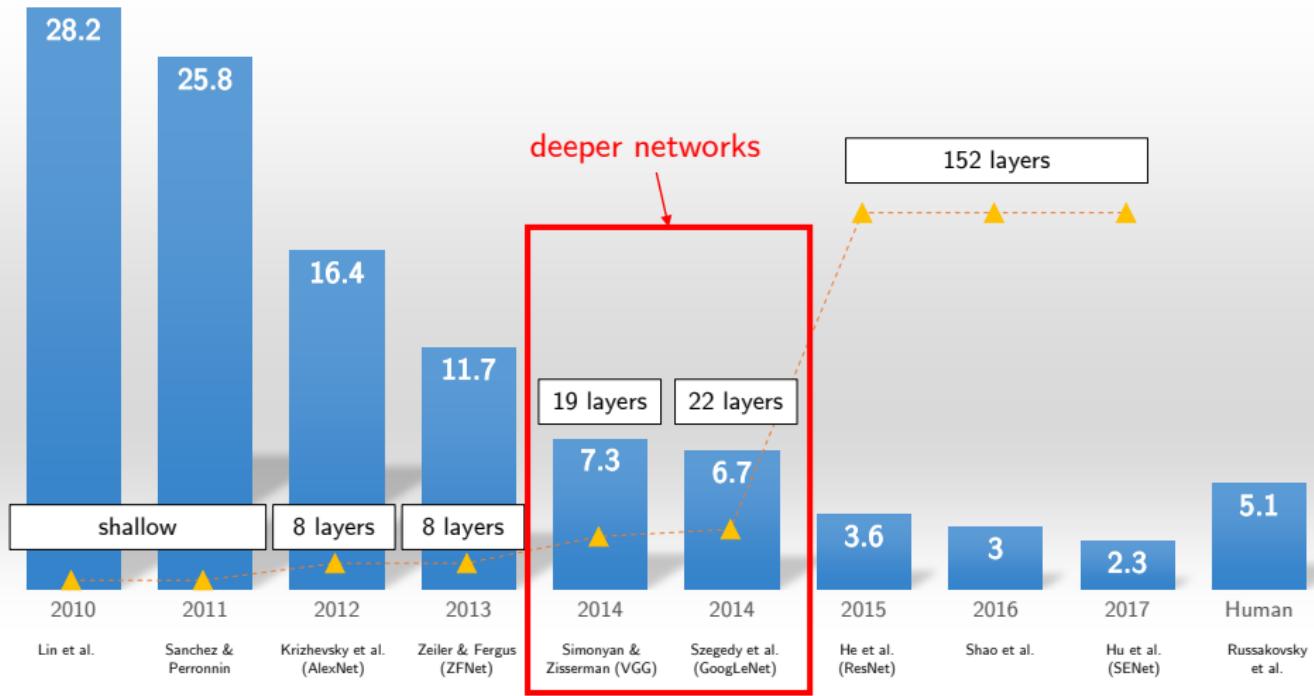
GoogLeNet

ResNet

Recent Architectures

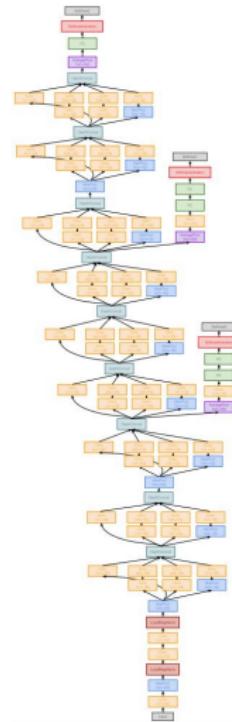
Summary

ImageNet challenge winners



GoogLeNet

- Szegedy et al. (2014)
- key idea: deeper networks with **computational efficiency**
 - ▶ 22 layers
 - ▶ efficient “inception” module
 - ▶ minimal use of FC layers
 - ▶ only 5 million parameters! parameter 수가 굉장히 적음!!!
(12x less than AlexNet)
 - ▶ ILSVRC'14 classification winner (6.7% top 5 error)



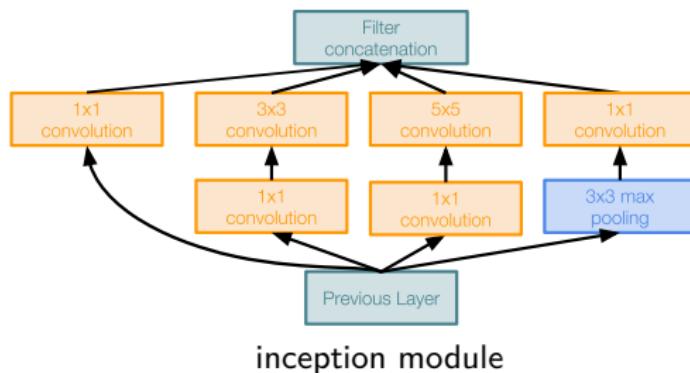
(source: cs231n)



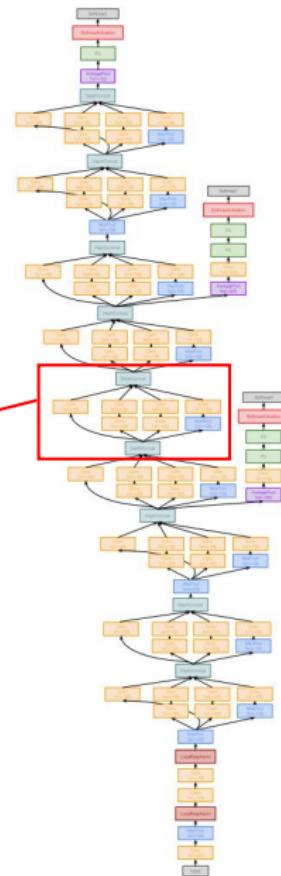
(source: Warner Bros. Pictures, <http://knowyourmeme.com/memes/we-need-to-go-deeper>)

- inception module

- ▶ design a good **local network** topology
(network within a network)
- ▶ then stack these modules

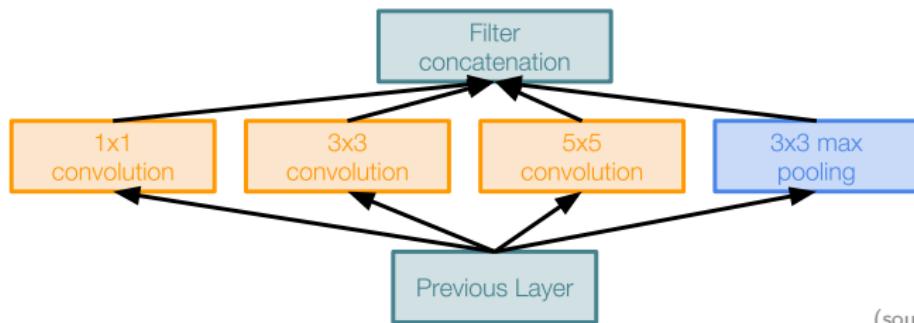


어떤 filter size가 좋을지 다 해봄
게다가 이 작업은 parallel하게 수행 가능함



(source: cs231n)

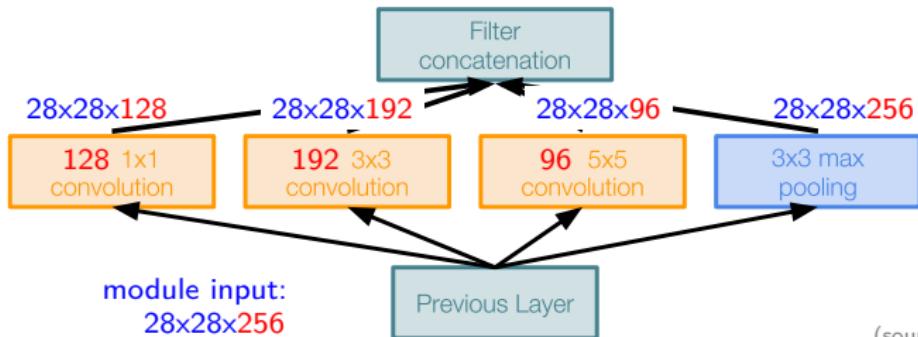
Naïve inception module



(source: cs231n)

- apply **parallel filter** operations on the input
 - ▶ multiple receptive field sizes (1×1 , 3×3 , 5×5) for convolution
 - ▶ pooling (3×3)
- concatenate all filter outputs together: **depth-wise stacking**

- problem with this idea: compute-intensive(expensive)



(source: cs231n)

- output size after filter concatenation: 529k**

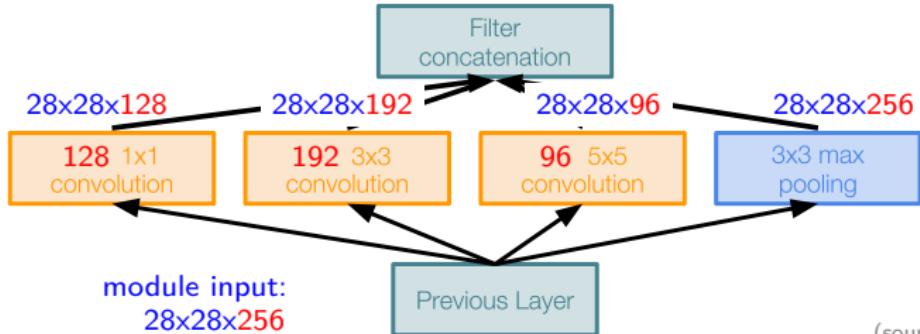
$$\blacktriangleright 28 \times 28 \times (128 + 192 + 96 + 256) = 28 \times 28 \times 672$$

filter의 depth는 input depth와
같으므로 256임!!! 빨간 숫자는
filter depth가 아니라 filter 갯
수!!!, pooling의 경우는 빨간색이
filter depth를 의미하므로 256!

- total number of convolution operations: 854M**

$$\blacktriangleright \underbrace{28 \times 28 \times 128 \times 1 \times 1 \times 256}_{(1 \times 1 \text{ conv, } 128)} + \underbrace{28 \times 28 \times 192 \times 3 \times 3 \times 256}_{(3 \times 3 \text{ conv, } 192)} + \underbrace{28 \times 28 \times 96 \times 5 \times 5 \times 256}_{(5 \times 5 \text{ conv, } 96)}$$

\Rightarrow very expensive to compute



(source: cs231n)

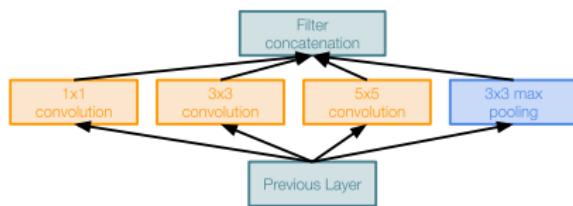
- another challenge:
 - ▶ pooling layer **preserves feature depth**
 - ⇒ total depth after concatenation → can **only grow** at every layer

- solution
 - ▶ “bottleneck” layers
- ↑
 use 1×1 convolution to reduce feature depth

Inception module

- comparison:

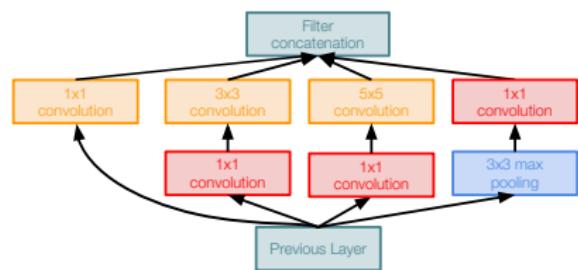
naïve inception module



(source: cs231n)

1x1 convolution(빨간 직사각형)을 통해 dimension reduction을 수행함, pooling의 경우 dimension reduction을 먼저하면 데이터 손실이 너무 커서 pooling 이후에 dimension reduction 수행

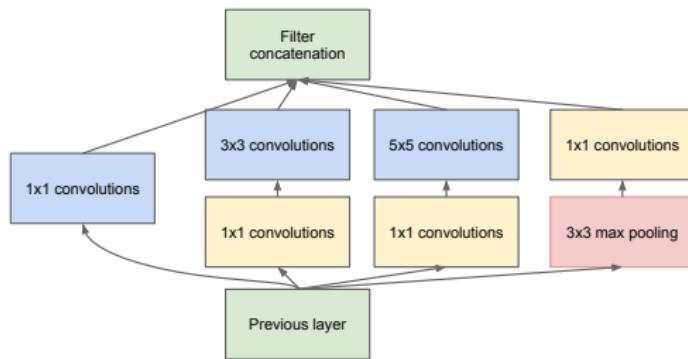
inception module with dimension reduction



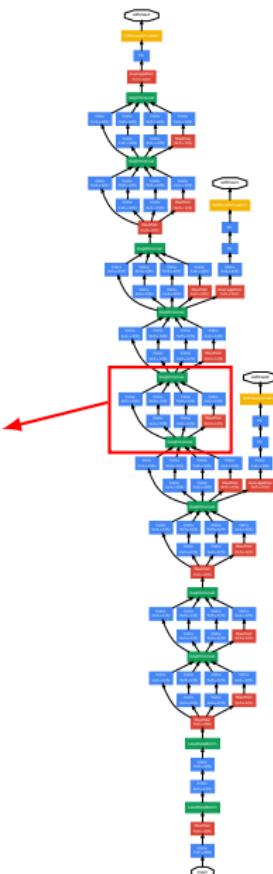
- ▶ 1x1 conv “**bottleneck**” layers
- ▶ the same setup as on page 29:
845M ops → 358M ops

GoogLeNet

- stacked inception modules
 - ▶ with dimension reduction on top of each other

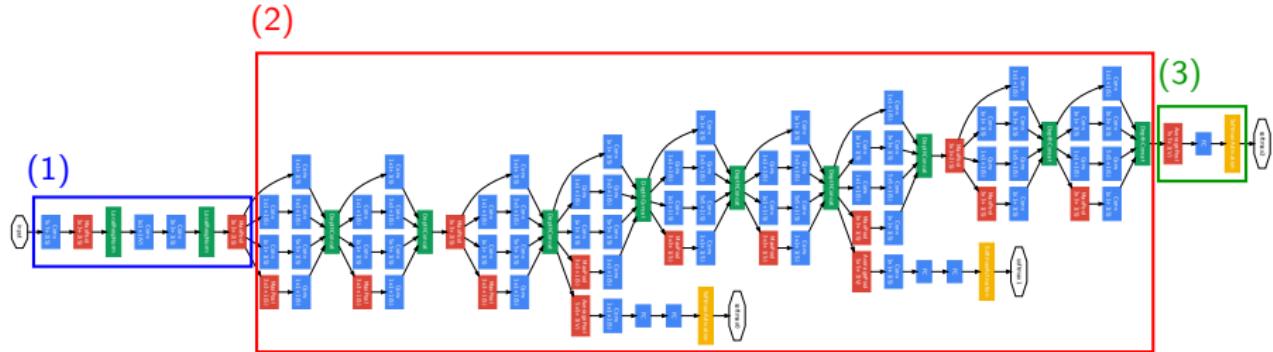


파란색 => filter size 별 convolution,
노란색 => dimension reduction
빨간색 => pooling



(source: Szegedy et al., 2014)

- full GoogLeNet architecture:



(source: Szegedy et al., 2014)

(1) stem network: CONV-POOL-2xCONV-POOL

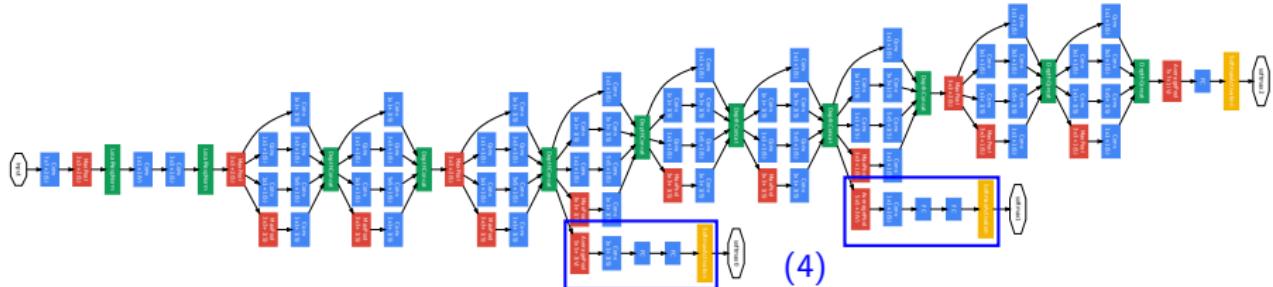
(2) stacked Inception modules

(1): 보통 맨 앞 layer에 이 작업을 수행,

(3) classifier output

(2): inception modules

- full GoogLeNet architecture:



(source: Szegedy et al., 2014)

(4) => gradient booster

(4) auxiliary classification outputs: AvgPOOL-1x1CONV-FC-FC-SOFTMAX

▷ to inject additional gradient at lower layers

- total 22 layers with weights
 - ▶ parallel layers count as 1 layer \Rightarrow 2 layers per inception module
 - ▶ auxiliary output layers: not counted in

Outline

Background

CNN Architectures

AlexNet

VGG

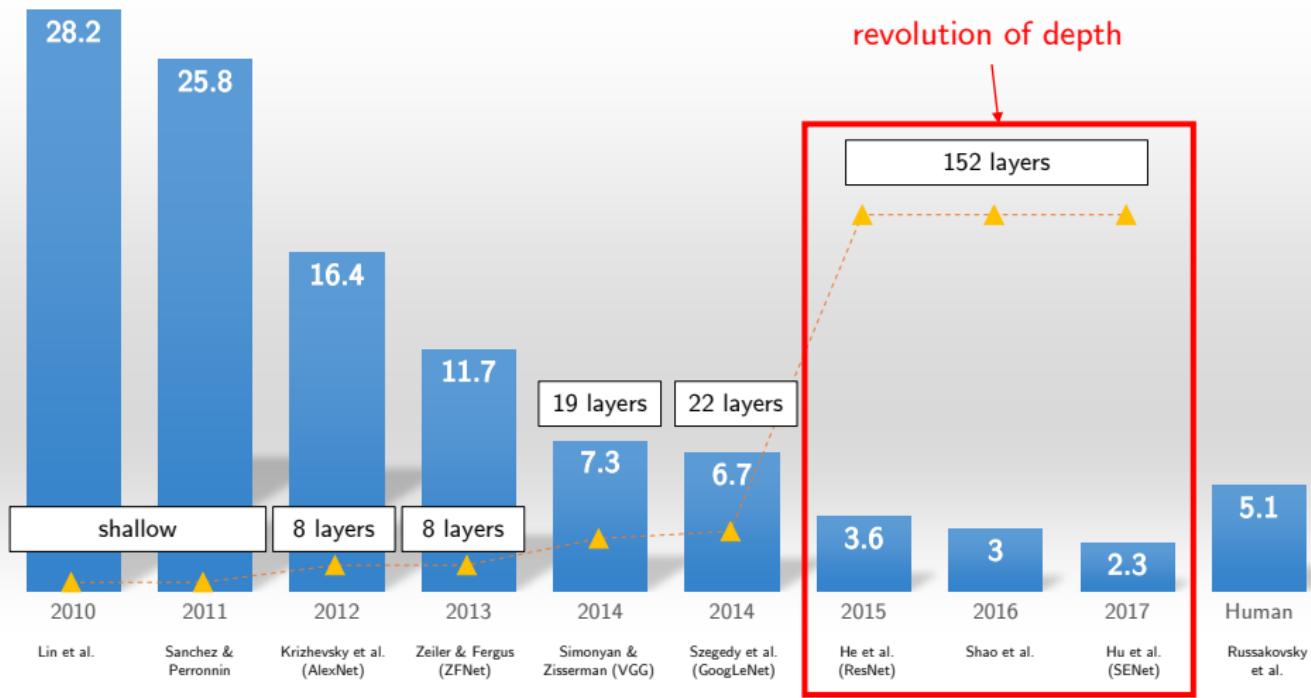
GoogLeNet

ResNet

Recent Architectures

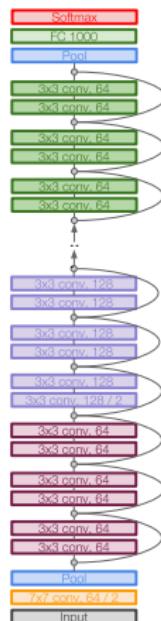
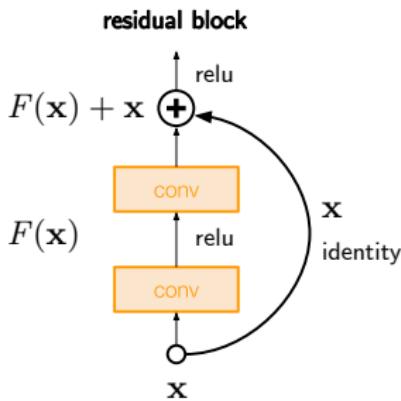
Summary

ImageNet challenge winners



ResNet

- He et al. (2015)
 - key idea: **very deep** nets using _____ connections
 - ▶ 152-layer model for ImageNet
 - ▶ ILSVRC'15 classification winner²
(3.57% top 5 error)



(source: cs231n)

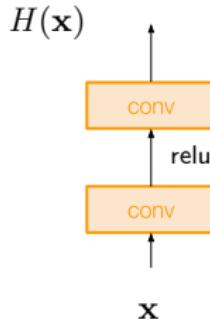
²swept all classification and detection competitions in ILSVRC'15 and COCO'15

- intuition:
 - ▶ if trained appropriately, deeper models should be able to perform
 - ▷ **at least as well** as shallower models
- a solution by construction:
 - ▶ **copy** the learned layers from the shallower model
 - ▶ set additional layers to _____ **mapping**

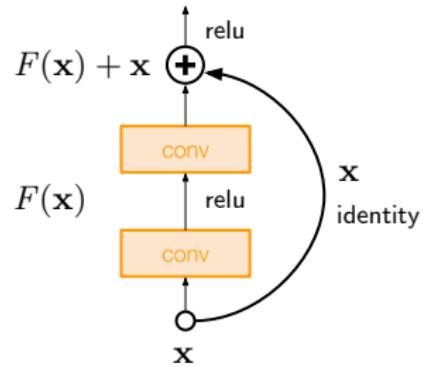
Residual block

- use network layers to fit a _____ mapping: $F(\mathbf{x}) = \overbrace{H(\mathbf{x}) - \mathbf{x}}$
 - instead of directly trying to fit a desired underlying mapping $H(\mathbf{x})$

“plain” layers



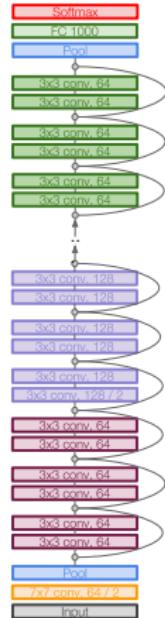
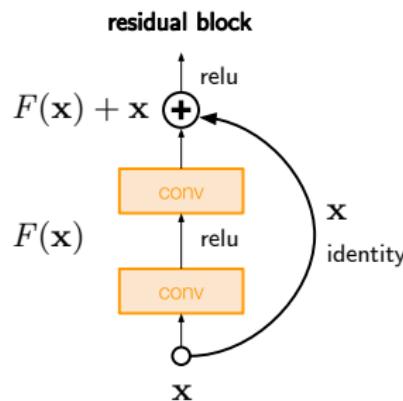
residual block



(source: cs231n)

Architecture

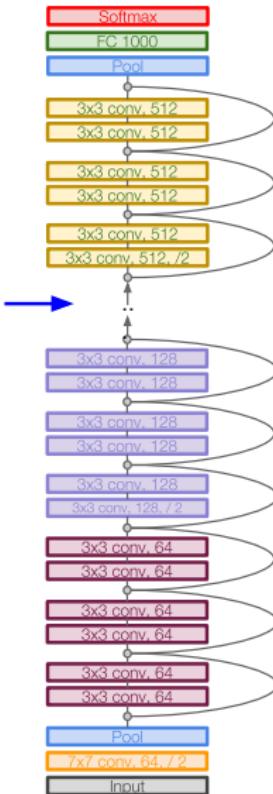
- stack residual blocks
- every residual block
 - ▶ has two 3×3 conv layers
- periodically
 - ▶ double # filters
 - ▶ downsample spatially (stride 2)
- at the beginning
 - ▶ additional conv layer
- no FC layers at the end
 - ▶ only FC1000 to output classes



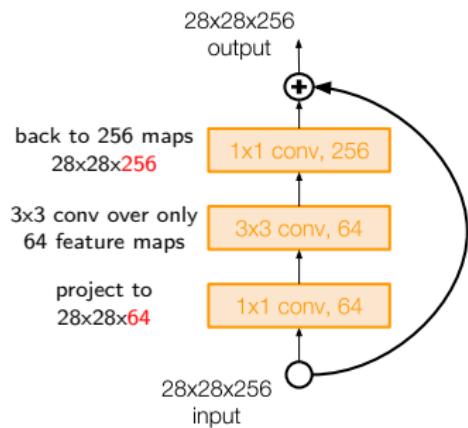
(source: cs231n)

- total depths:

- 34, 50, 101, or 152 layers for ImageNet



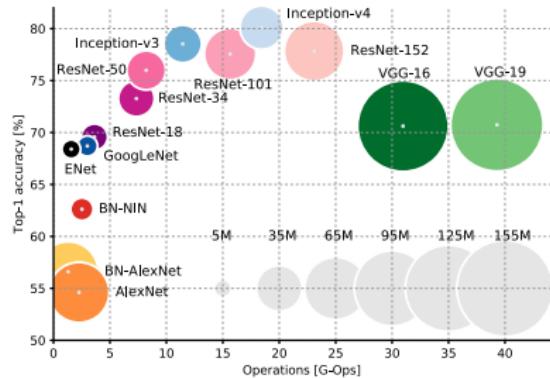
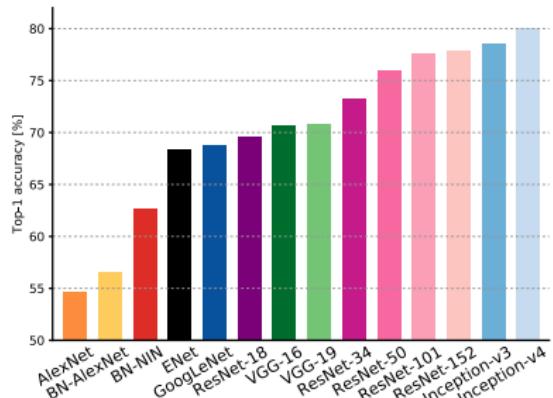
- for deeper nets (50+ layers)
 - use “_____” layers to improve efficiency (similar to GoogLeNet)



(source: cs231n)

- training details:
 - ▶ *batch normalization* after every CONV layer
 - ▶ *Xavier initialization*: initial weight $\sim \mathcal{N}(0, 1/n)$ where $n = \#$ neurons
 - ▶ SGD + momentum (0.9)
 - ▶ learning rate: 0.1 (divided by 10 when validation error plateaus)
 - ▶ mini-batch size: 256
 - ▶ weight decay: 10^{-5}
 - ▶ no dropout used
- results
 - ▶ ILSVRC 2015 winner in all five main tracks $\underbrace{(3.6\% \text{ top 5 error})}_{\uparrow}$
better than “_____ performance” (Russakovsky, 2014)

Comparison³



(source: Canziani et al., 2017)

- ▶ Inception-v4: ResNet + Inception
- ▶ VGG: highest memory, most operations
- ▶ _____: most efficient
- ▶ AlexNet: smaller compute, still memory heavy, lower accuracy
- ▶ _____: moderate efficiency depending on model, highest accuracy

³(in left figure) x-axis: amount of operations for a single forward pass; circle size \propto # parameters

Outline

Background

CNN Architectures

AlexNet

VGG

GoogLeNet

ResNet

Recent Architectures

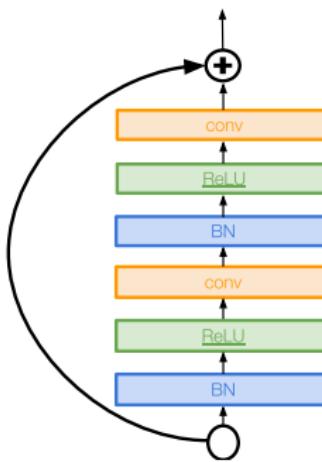
Summary

Improving ResNets

- ideas:
 - ▶ improved residual block
 - ▶ wide ResNet
 - ▶ ResNeXt
 - ▶ stochastic depth
 - ▶ multi-scale ensembling
 - ▶ feature recalibration (SENet)

Identity mappings in deep residual networks (He et al., 2016)

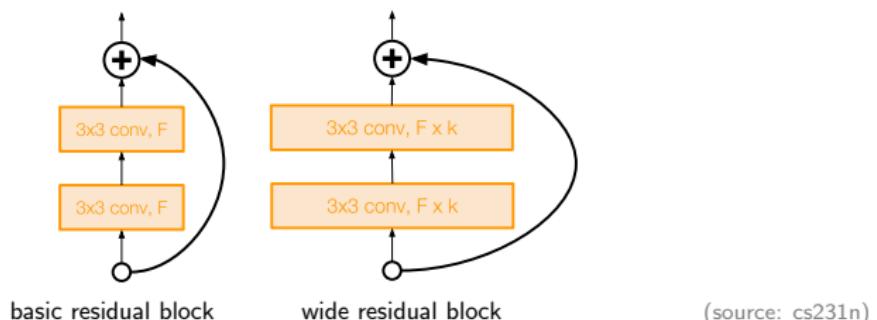
- improved ResNet block design
 - creates a more **direct path** for propagating info throughout net
i.e. moves _____ to residual mapping pathway



(source: He et al.)

Wide ResNet (Zagoruyko et al., 2016)

- the authors argue: **residuals** are the important factor, **not depth**
- use **wider** residual blocks
 - i.e. $F \times k$ filters instead of F filters in each layer
 - 50-layer wide ResNet outperforms 152-layer original ResNet
- computational benefit
 - increasing width (instead of depth)
 - more computationally efficient (_____)

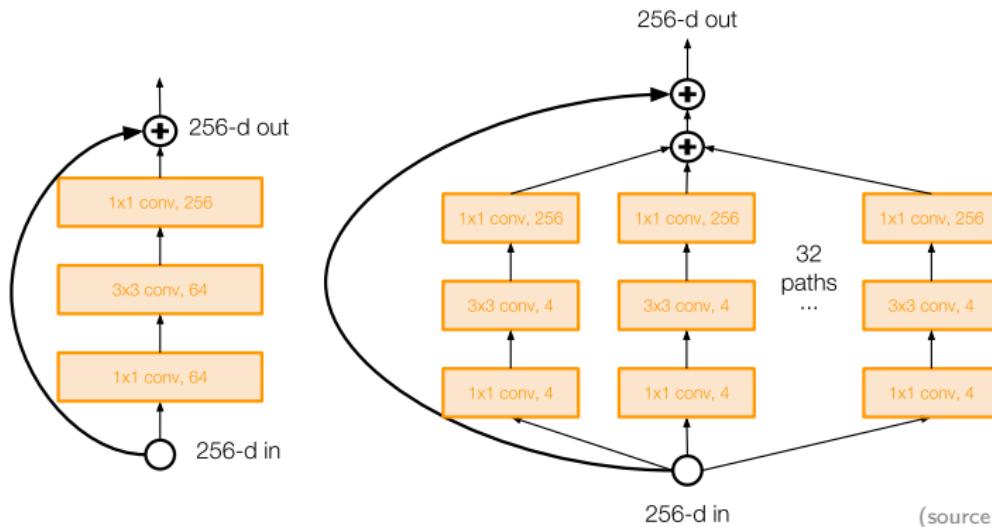


(source: cs231n)

ResNeXt (Xie et al., 2016)

- aggregated residual transformations

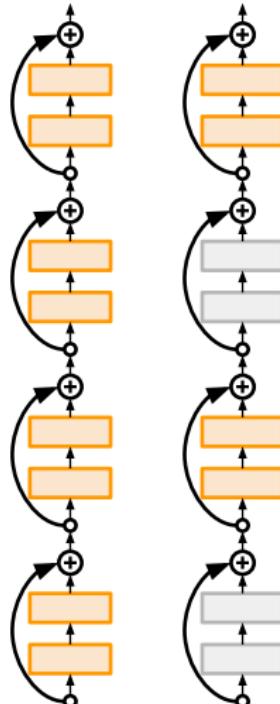
► increases width of residual block through multiple pathways
similar in spirit to inception module



(source: cs231n)

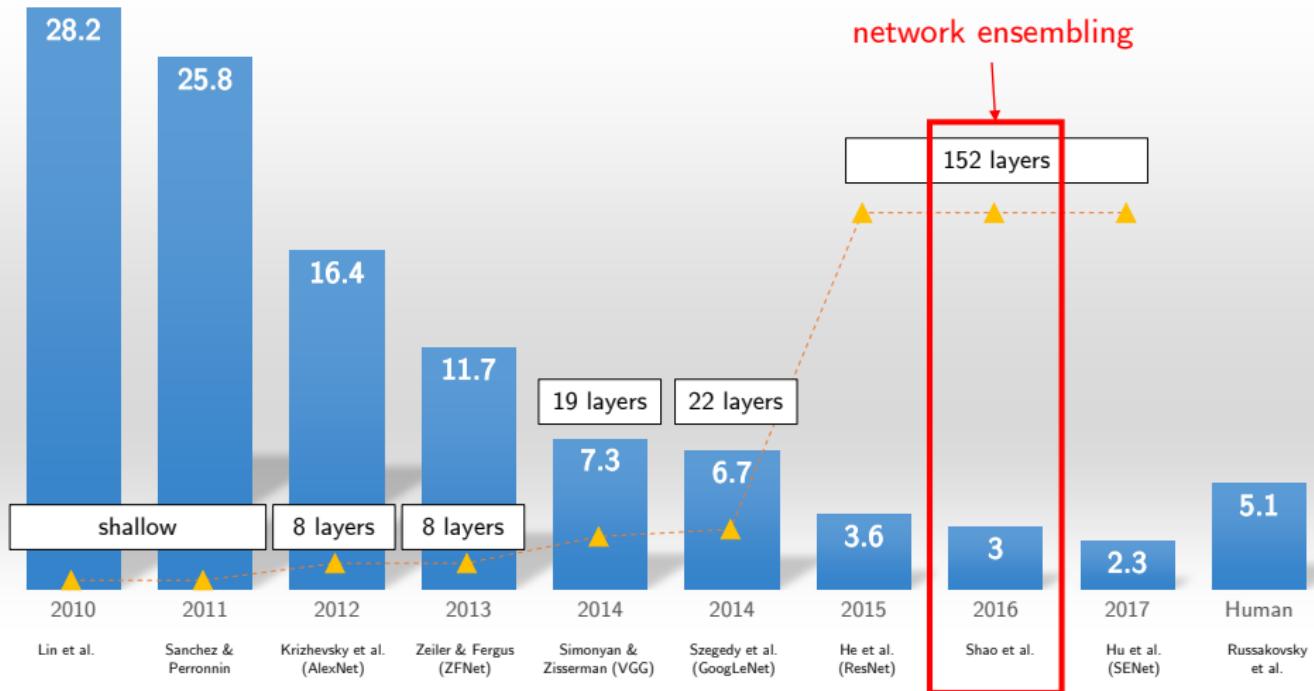
Stochastic depth (Huang et al., 2016)

- motivation:
 - ▶ reduce vanishing gradients and training time through **short networks** during training
- details:
 - ▶ randomly _____ a subset of layers during each training pass
 - ▶ bypass with **identity** function
 - ▶ use full deep network at test time



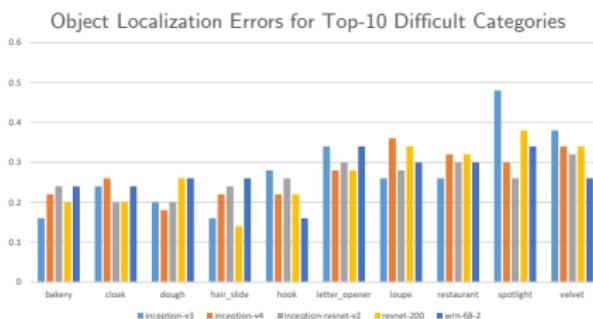
(source: cs231n)

ImageNet challenge winners



Multi-scale ensembling (Shao et al., 2016)

- ILSVRC'16 classification winner⁴
 - ▶ “Good Practices for Deep Feature Fusion”
- idea: multi-scale _____ of
 - ▶ inception, inception-ResNet, ResNet, wide ResNet models

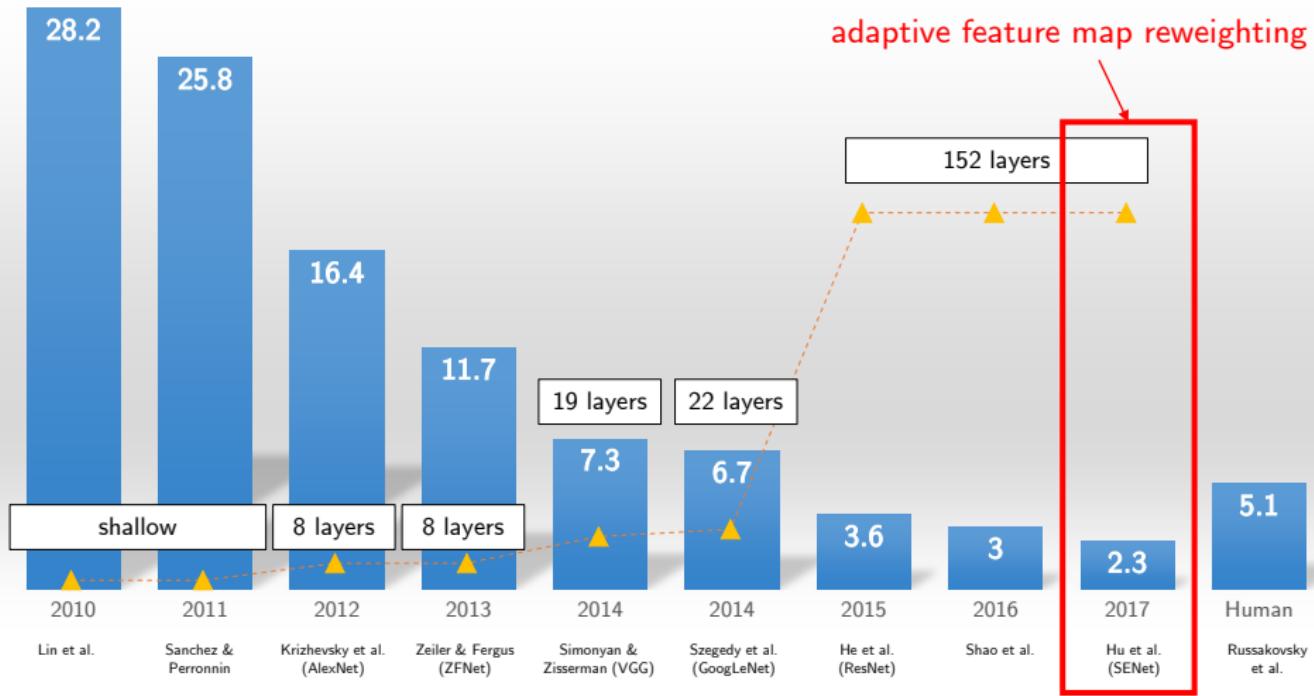


method	error (%)
Resnet-200	4.26
Inception-v3	4.20
Inception-v4	4.01
Inception-Resnet-v2	3.52
Fusion (val)	2.92
Fusion (test)	2.99

(source: Shao et al.)

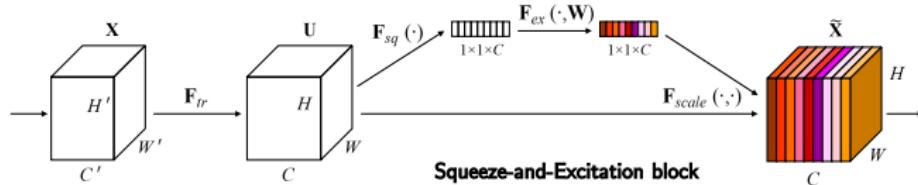
⁴the authors: The Third Research Institute of the Ministry of Public Security, China

ImageNet challenge winners

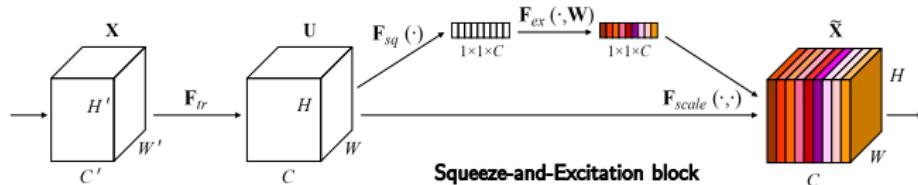


Squeeze-and-Excitation Networks (SENet) (Hu et al., 2017)

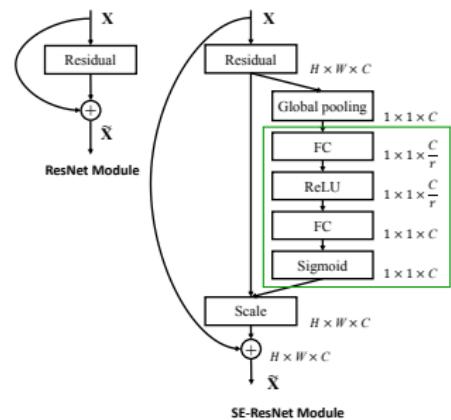
- ILSVRC'17 classification winner
 - ▶ base architecture: ResNeXt-152
 - ▶ introduces SE block (applicable to a variety of nets)
 - ▷ **squeeze**: global information embedding
 - ▷ **excitation**: adaptive recalibration
- main idea:
 - ▶ improve representational power of a network
by modeling interdependencies between _____ of conv features



(source: Hu et al.)



- $\mathbf{F}_{tr} : \mathbf{X}^{H' \times W' \times C'} \mapsto \mathbf{U}^{H \times W \times C}$ (a conv operation)
- $\mathbf{F}_{sq} : \mathbf{U}^{H \times W \times C} \mapsto \mathbf{Z}^{1 \times 1 \times C}$
 - ▶ global average pooling (each feature map → a scalar; depth maintained)
 - ▶ “global information embedding” (squeeze)
- $\mathbf{F}_{ex} : \mathbf{Z}^{1 \times 1 \times C} \mapsto \mathbf{S}^{1 \times 1 \times C}$
 - ▶ $\underbrace{\text{FC} \rightarrow \text{ReLU}}_{\text{compress}} \rightarrow \underbrace{\text{FC} \rightarrow \text{Sigmoid}}_{\text{decompress}}$
 - ▶ to calculate **scale** for each feature map
- $\mathbf{F}_{scale} = \mathbf{S}^{1 \times 1 \times C} \odot \mathbf{U}^{1 \times 1 \times C} = \tilde{\mathbf{X}}$
 - ▶ reweight feature maps
 - ▶ “adaptive feature recalibration” (excitation)

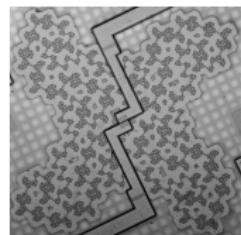


Recent developments

- beyond ResNets:
 - ▶ ultra-deep neural networks without residuals (FractalNet)
 - ▶ densely connected CNNs (DenseNet)
- efficient networks:
 - ▶ Caffe2Go (Facebook), TensorRT (NVIDIA), Core ML (Apple)
 - ▶ SqueezeNet
- meta/automated learning:
 - ▶ Cloud AutoML (Google)

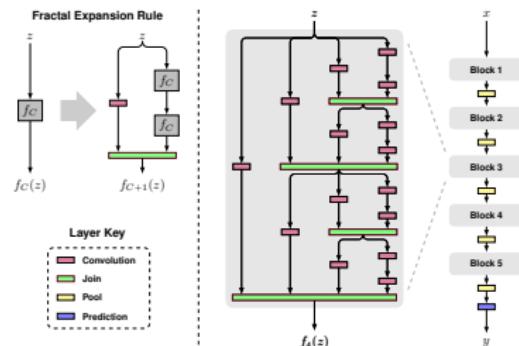
FractalNet (Larsson et al., 2017)

- ultra-deep neural networks without residuals
- argue:
 - key is **transitioning effectively** from shallow to deep
 - ⇒ residual representations are not necessary



(source: Hajimiri et al.)

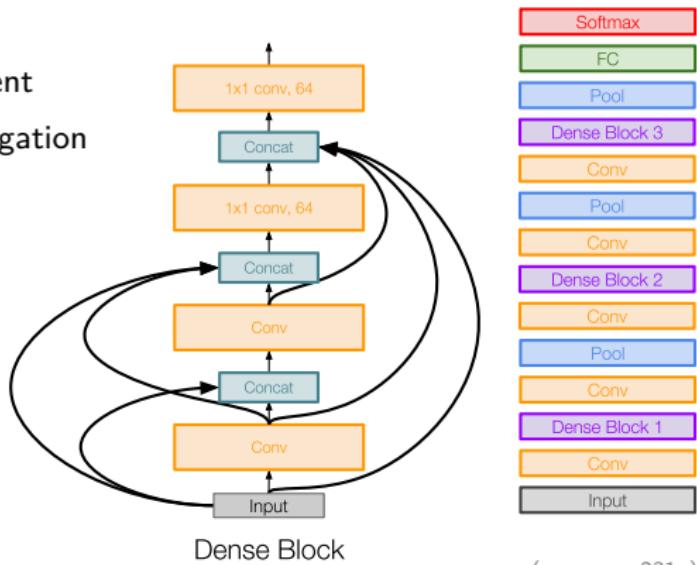
- propose: _____ architecture
 - both shallow/deep paths to output
 - trained with dropping out subpaths
 - full network at test time



(source: Larsson et al.)

DenseNet (Huang et al., 2017)

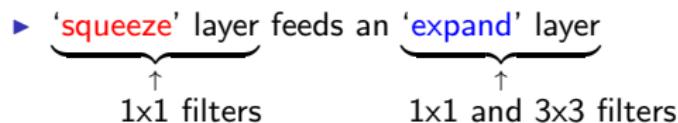
- densely connected convolutional networks
- idea: dense blocks
 - ▶ each layer is connected to _____ layer in feedforward fashion
- benefits:
 - ▶ alleviates vanishing gradient
 - ▶ strengthens feature propagation
 - ▶ encourages feature reuse



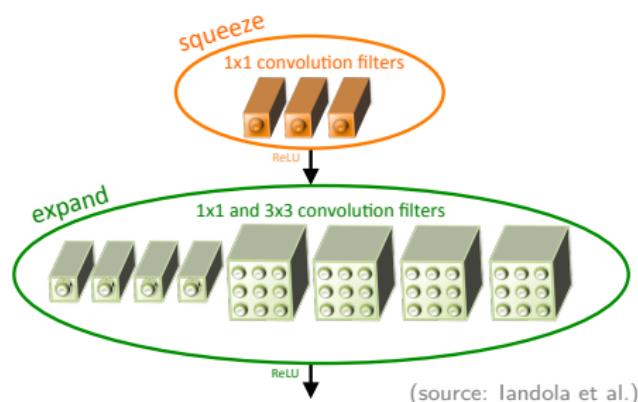
(source: cs231n)

SqueezeNet (Iandola et al., 2017)

- AlexNet-level accuracy with 50x fewer parameters and <0.5Mb model size
- architecture:



- benefits: memory footprints
 - model size: **510x smaller** than AlexNet



Google Cloud AutoML

- learning to learn

Cloud AutoML Vision



(source: Google)

Outline

Background

Summary

CNN Architectures

Summary

- famous four
 - ▶ AlexNet
 - ▶ VGG
 - ▶ GoogLeNet
 - ▶ ResNet
- beyond ResNet
 - ▶ FractalNet
 - ▶ DenseNet
- improving ResNet
 - ▶ wide ResNet
 - ▶ ResNeXt
 - ▶ stochastic depth
 - ▶ SENet
- other ideas
 - ▶ SqueezeNet
 - ▶ autoML