Advanced School for Computing and Imaging

UNIVERSITY OF AMSTERDAM

Vision in the Deep Learning Era

Cees Snoek, University of Amsterdam

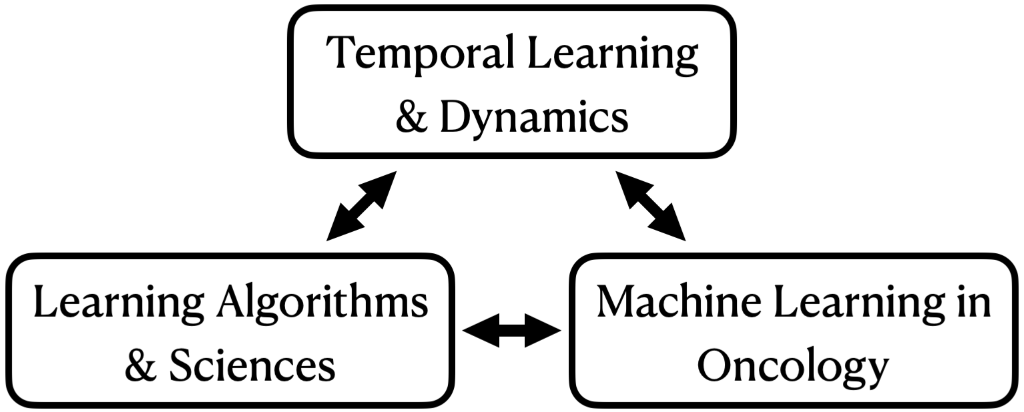
Efstratios Gavves, University of Amsterdam

With an invited tutorial by: Serge Belongie, University of Copenhagen

http://computervisionbylearning.info

Who am I?

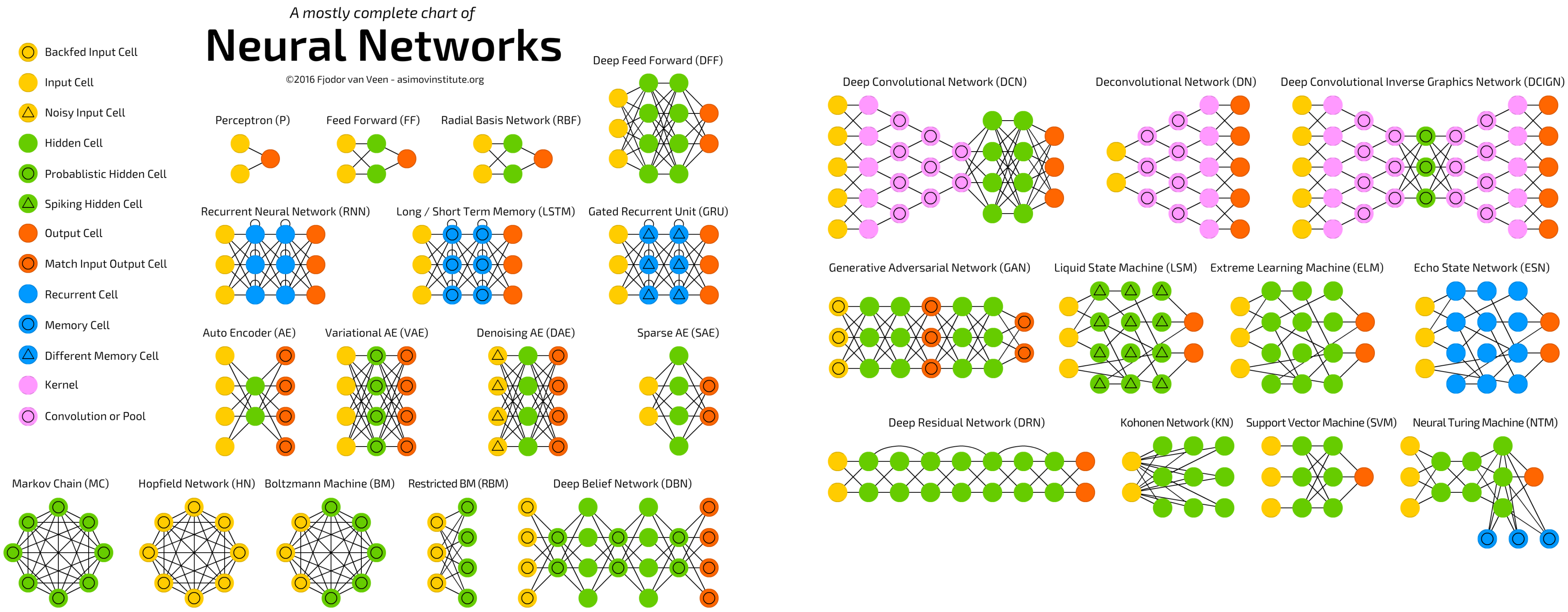
- Associate Professor at the UVA
  - ERC StG and NWO VIDI laureate
  - Co-director of QUVA (QC, Snoek, Welling) & POP-AART (NKI, Elekta, J.J. Sonke)
  - Teaching Deep Learning I & II
- ELLIS Scholar network of excellence in AI
- Co-founder of Ellogon.AI
  - Personalise immunotherapy in oncology with AI



UNIVERSITY OF AMSTERDAM

ELLOGON.AI

Neural Network Summary



Alexnet

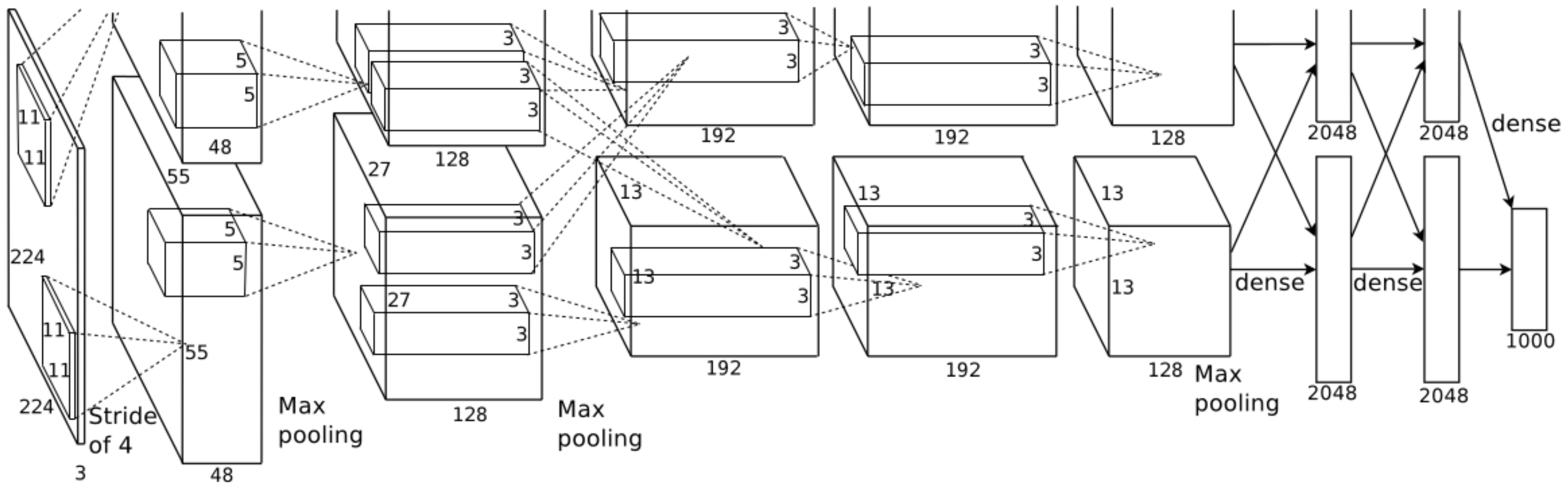
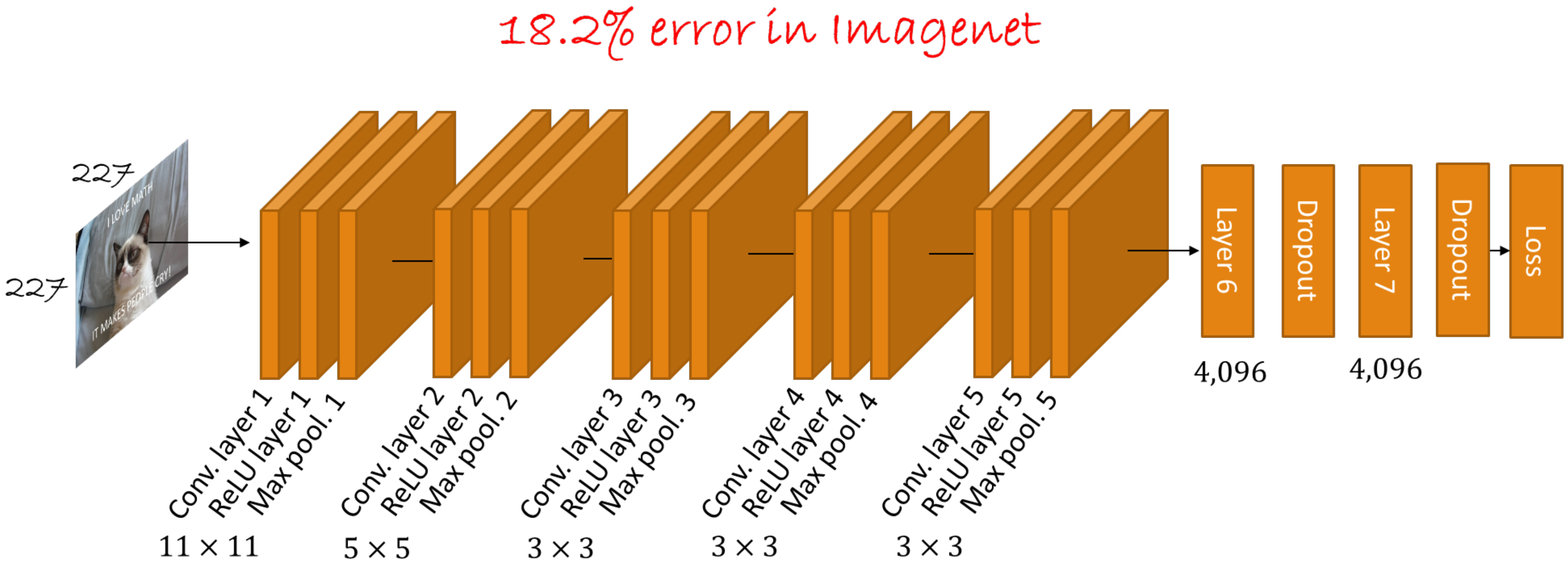
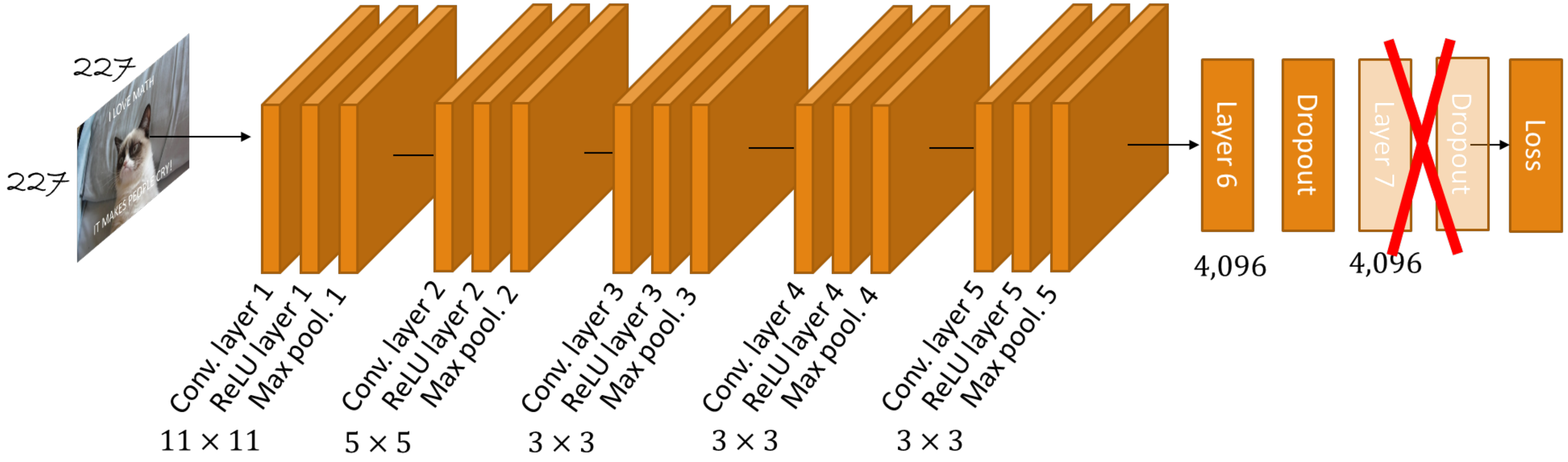


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

Architectural details

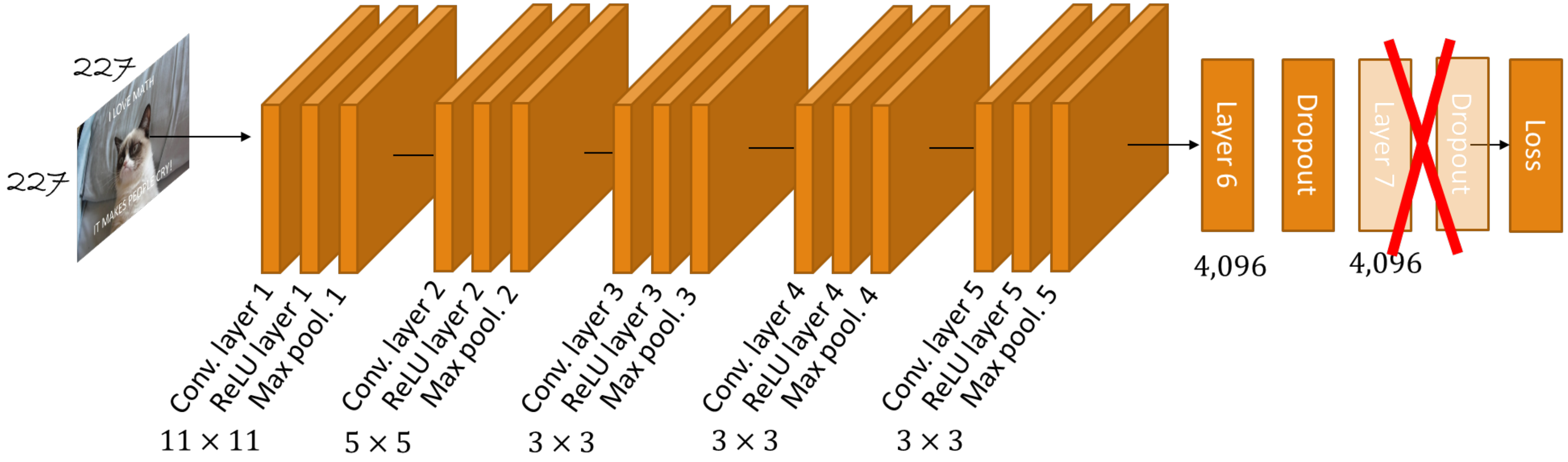


Removing layer 7: What happens?



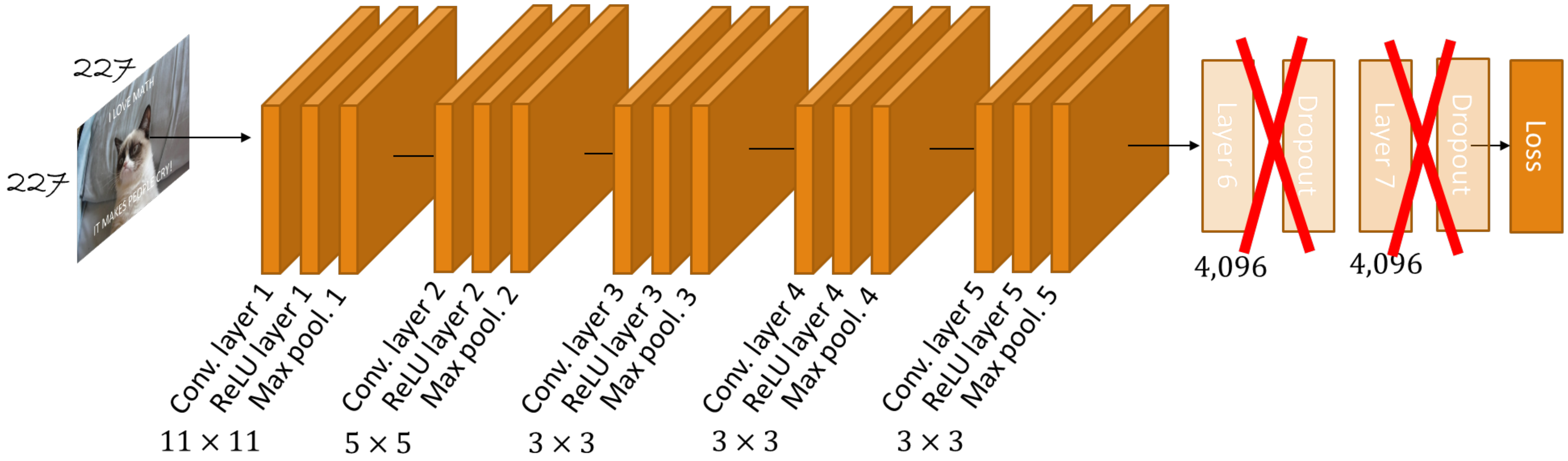
Removing layer 7

1.1% drop in performance, 16 million less parameters

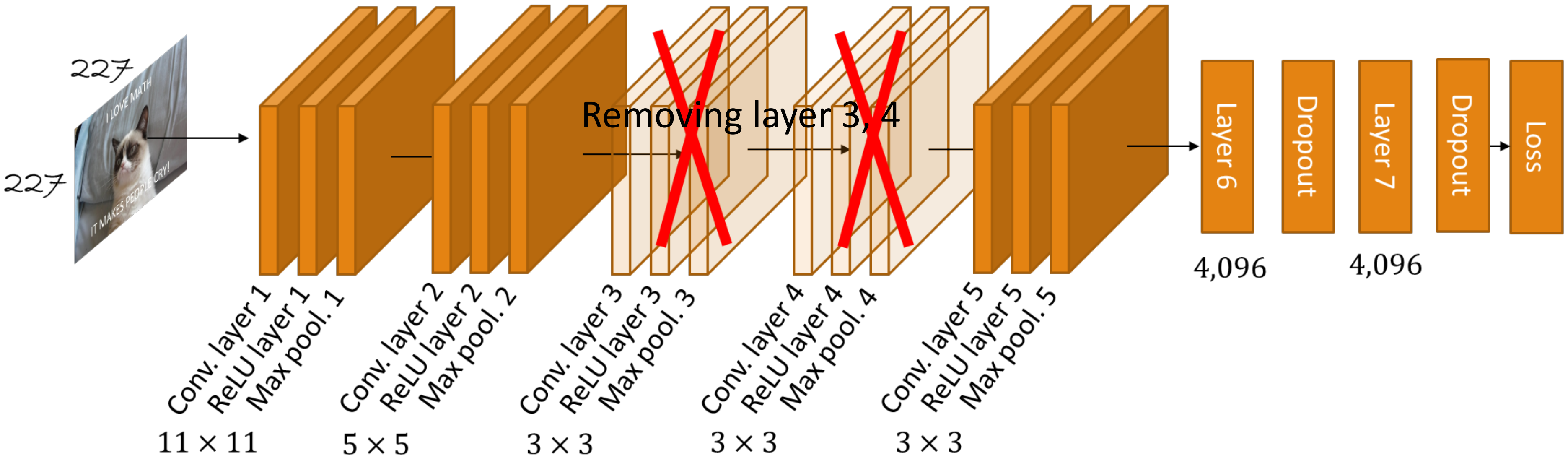


Removing layer 6, 7

5.7% drop in performance, 50 million less parameters

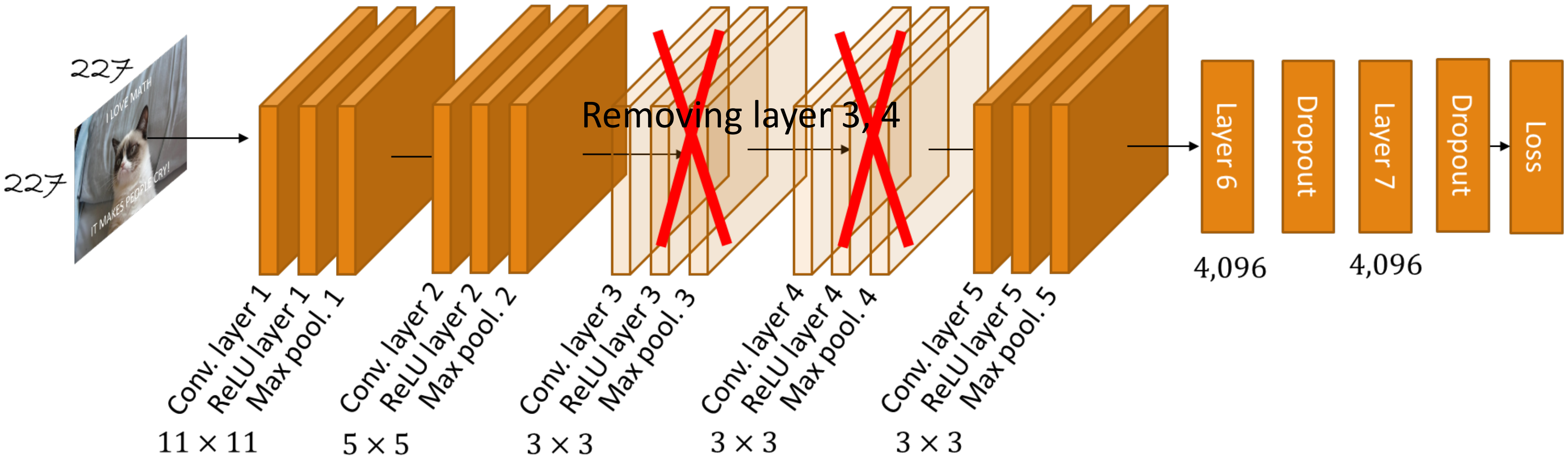


Removing layer 3, 4



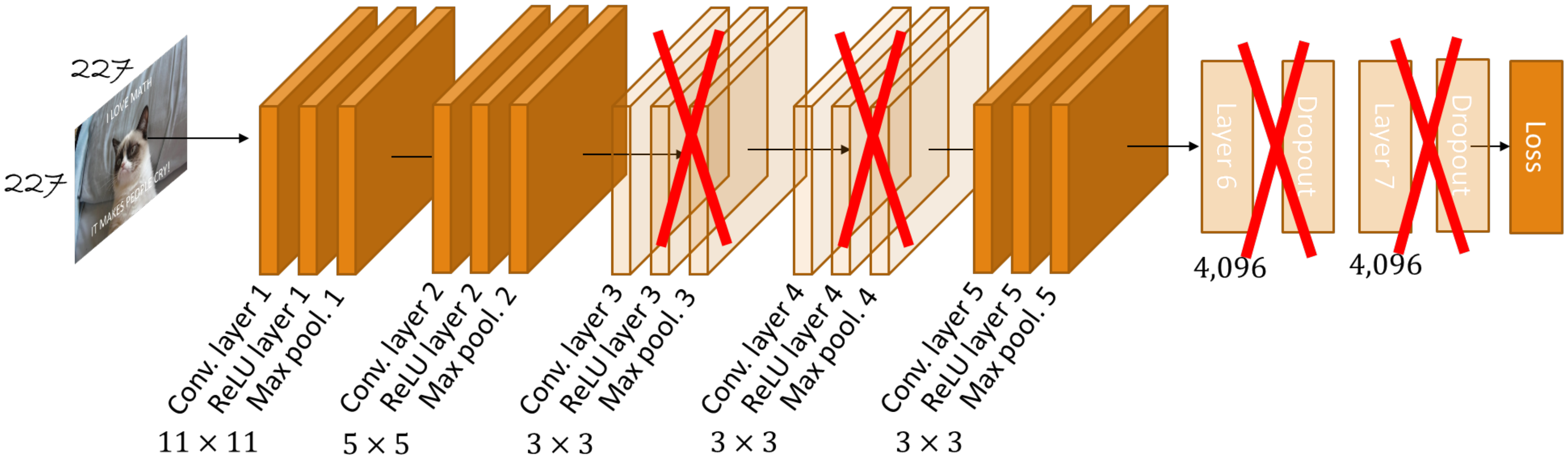
Removing layer 3, 4

3.0% drop in performance, 1 million less parameters. Why?

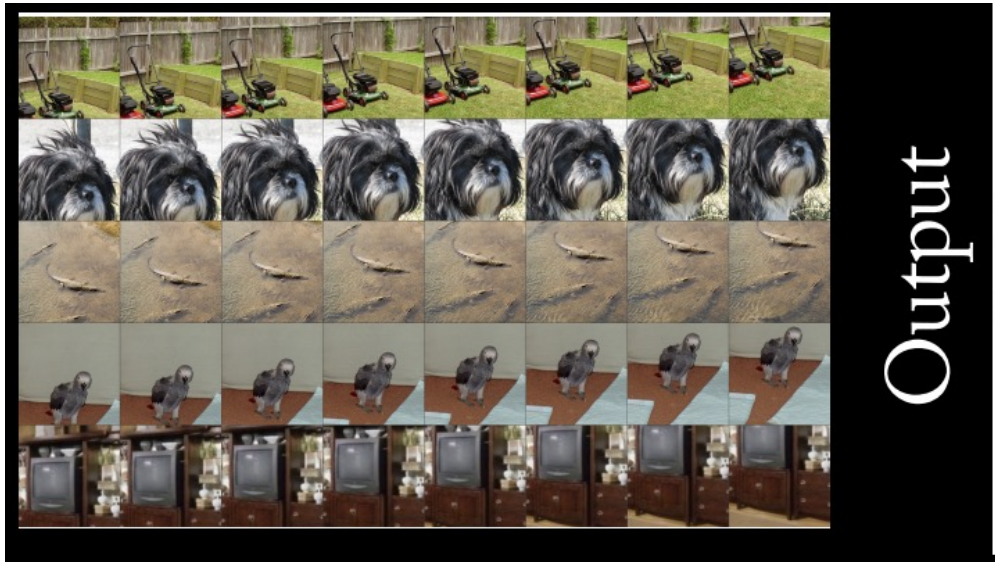


Removing layer 3, 4, 6, 7

33.5% drop in performance. Conclusion? <u>Depth!</u>

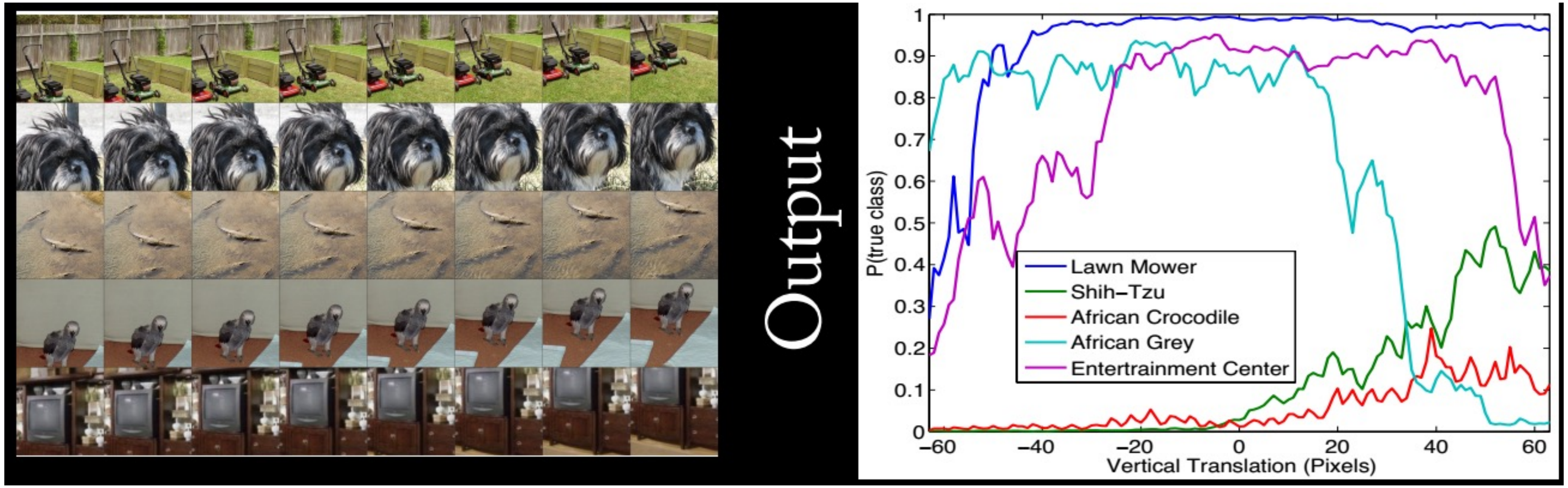


Quiz: Translation invariance?



Credit: R. Fergus slides in Deep Learning Summer School 2016

Translation invariance

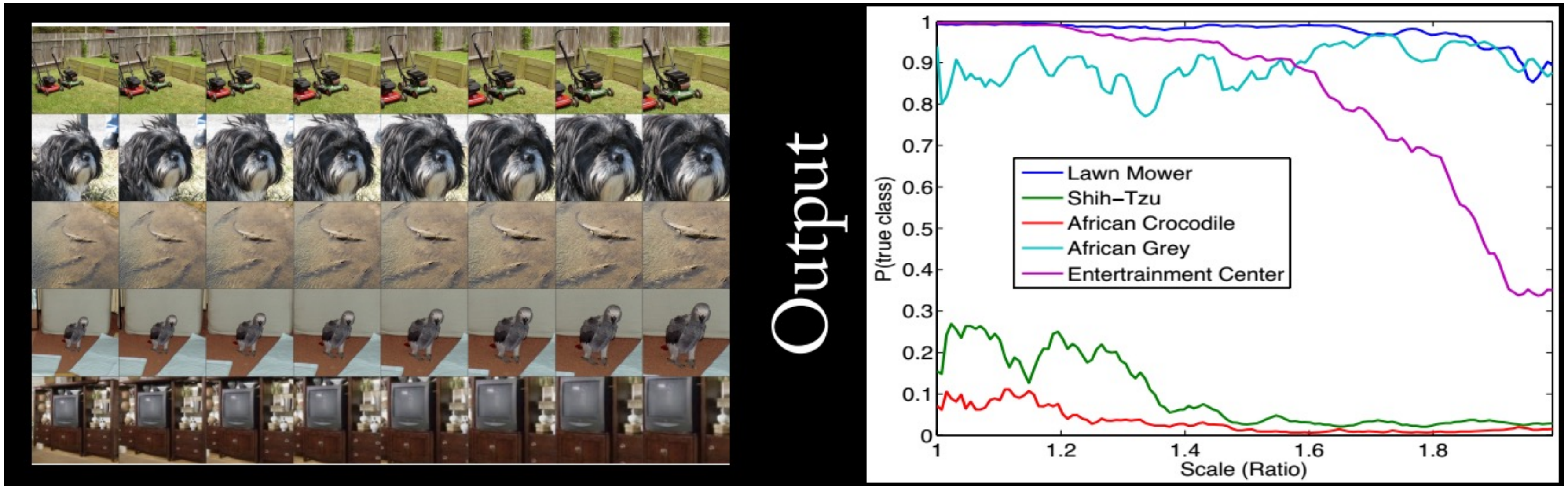


Credit: R. Fergus slides in Deep Learning Summer School 2016

Quiz: Scale invariance?

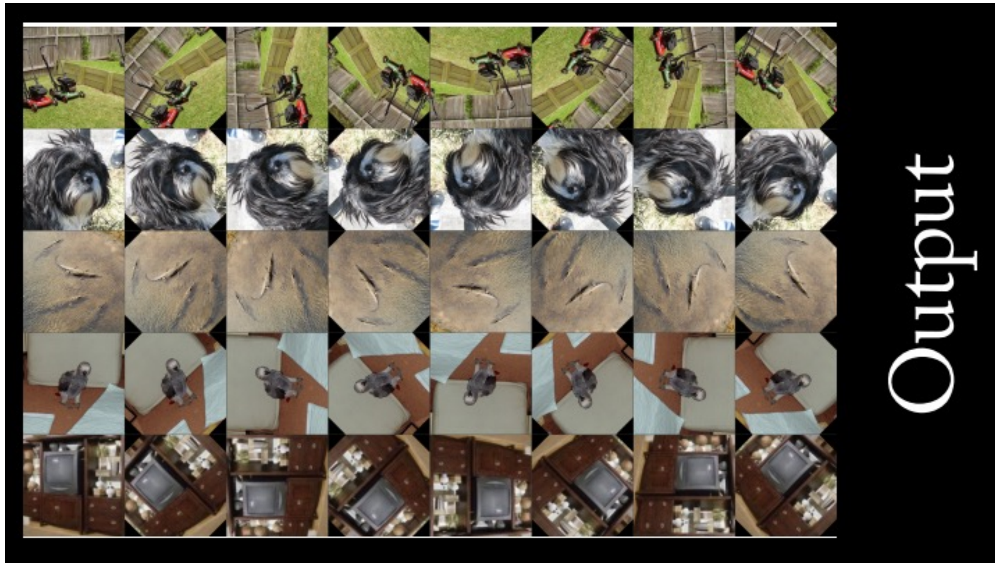
Credit: R. Fergus slides in Deep Learning Summer School 2016

Scale invariance



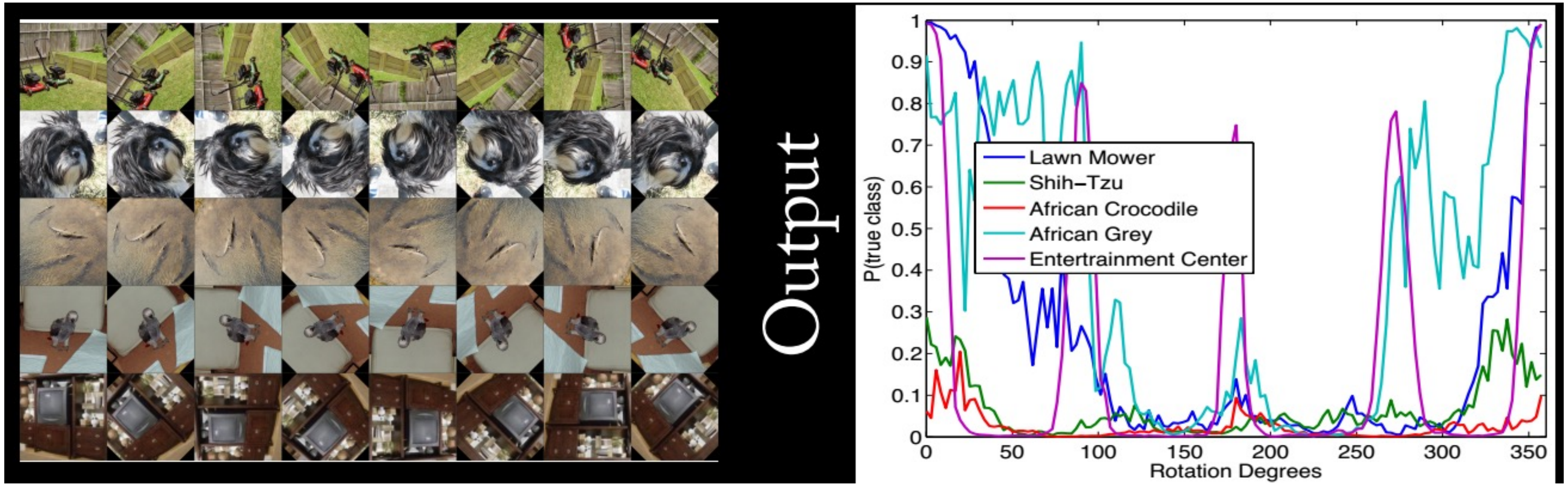
Credit: R. Fergus slides in Deep Learning Summer School 2016

Quiz: Rotation invariance?



Credit: R. Fergus slides in Deep Learning Summer School 2016

Rotation invariance



Credit: R. Fergus slides in Deep Learning Summer School 2016

Modern Deep Nets

- VGG-Net
- ResNet
  - ❑From 14 to 1000 layers
- Google Inception
  - ❑Networks as Direct Acyclic Graphs (DAG)
- ResNext
  - ❑ Factorizing ResNets
- DenseNet
  - ❑ResNets with multiple skip-connections
- Neural Architecture Search

- …and many more

More Depth? VGGnet

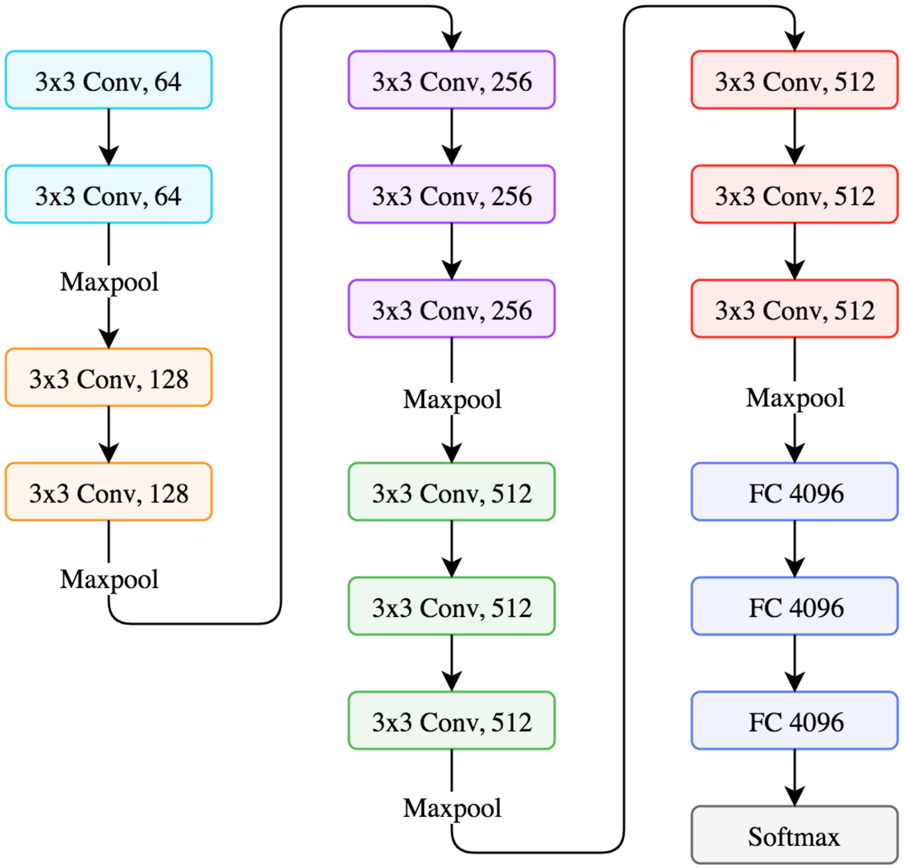
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	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					

Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

VGG16

- 7.3% error rate in ImageNet
- Compared to 18.2% o

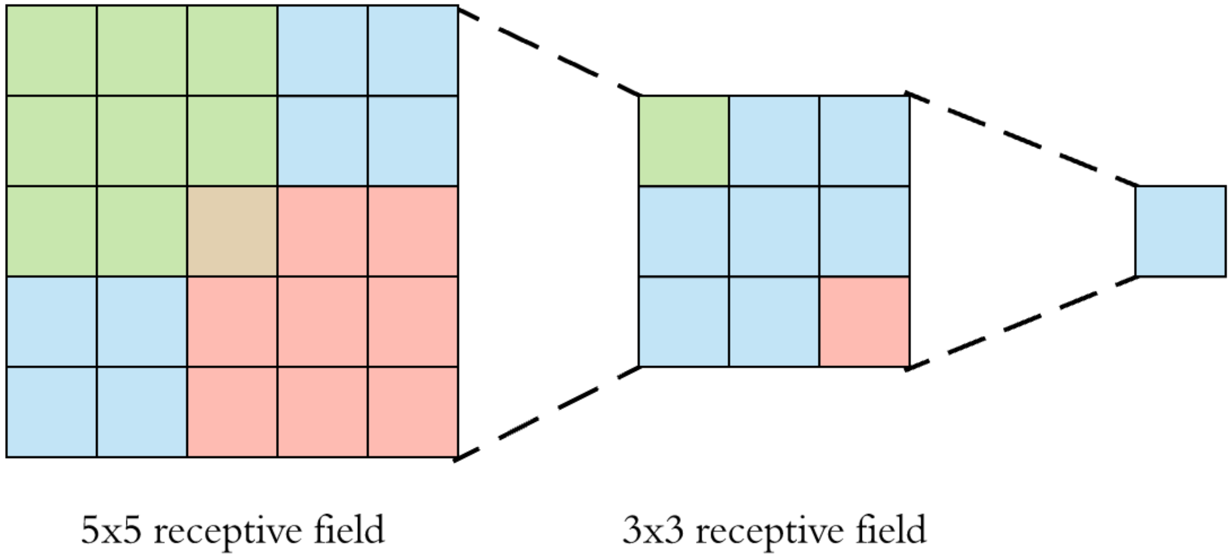


VGG16 characteristics

- Input size: 224×224
- Filter sizes: 3×3
- Convolution stride: 1
  - Spatial resolution preserved
- Padding: 1
- Max pooling: 2×2 with a stride of 2
- ReLU activations
- No fancy input normalizations
  - No Local Response Normalizations
- Although deeper, number of weights is not exploding

Why 3x3 filters?

- The <u>smallest</u> possible filter to captures the "up", "down", "left", "right"
- Two 3×3 filters have the receptive field of one 5×5
- Three 3×3 filters have the receptive field of …



Picture credit: [Arden Dertat]

Why 3x3 filters?

- The smallest possible filter to captures the “up”, “down”, “left”, “right”
- Two 3×3 filters have the receptive field of one 5×5
- Three 3×3 filters have the receptive field of one 7×7
- 1 large filter can be replaced by a deeper stack of successive smaller filters
- Benefit?

Why 3x3 filters?

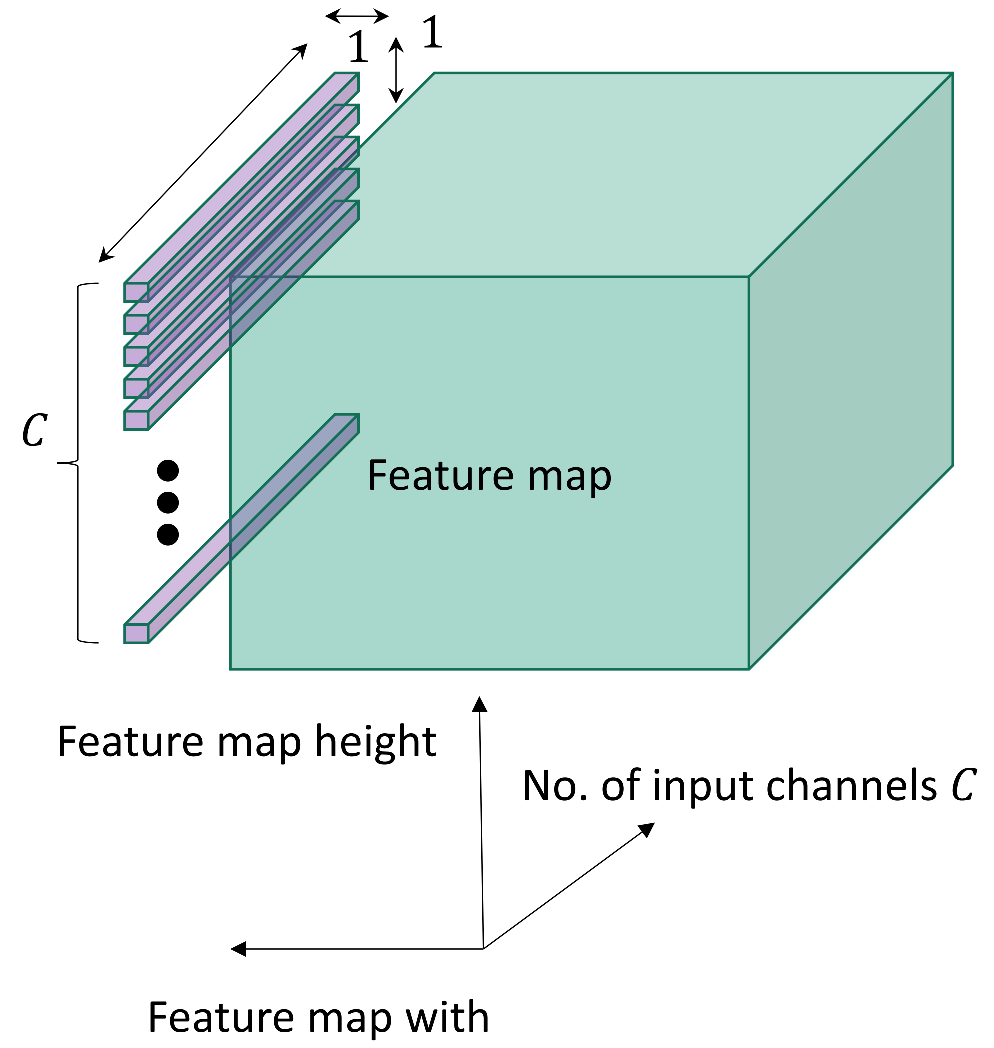
- The <u>smallest</u> possible filter to captures the "up", "down", "left", "right"
- Two 3×3 filters have the receptive field of one 5×5
- Three 3×3 filters have the receptive field of one 7×7
- 1 large filter can be replaced by a deeper stack of successive smaller filters
- Benefit?
- Three more nonlinearities for the same "size" of pattern learning
- Also fewer parameters and regularization

$$(3\times3\times C)\times3 = 27\cdot C, 7\times7\times C\times1 = 49\cdot C$$

- **A large filter can be replaced by a deeper, potentially more powerful, stack of successive smaller filters**

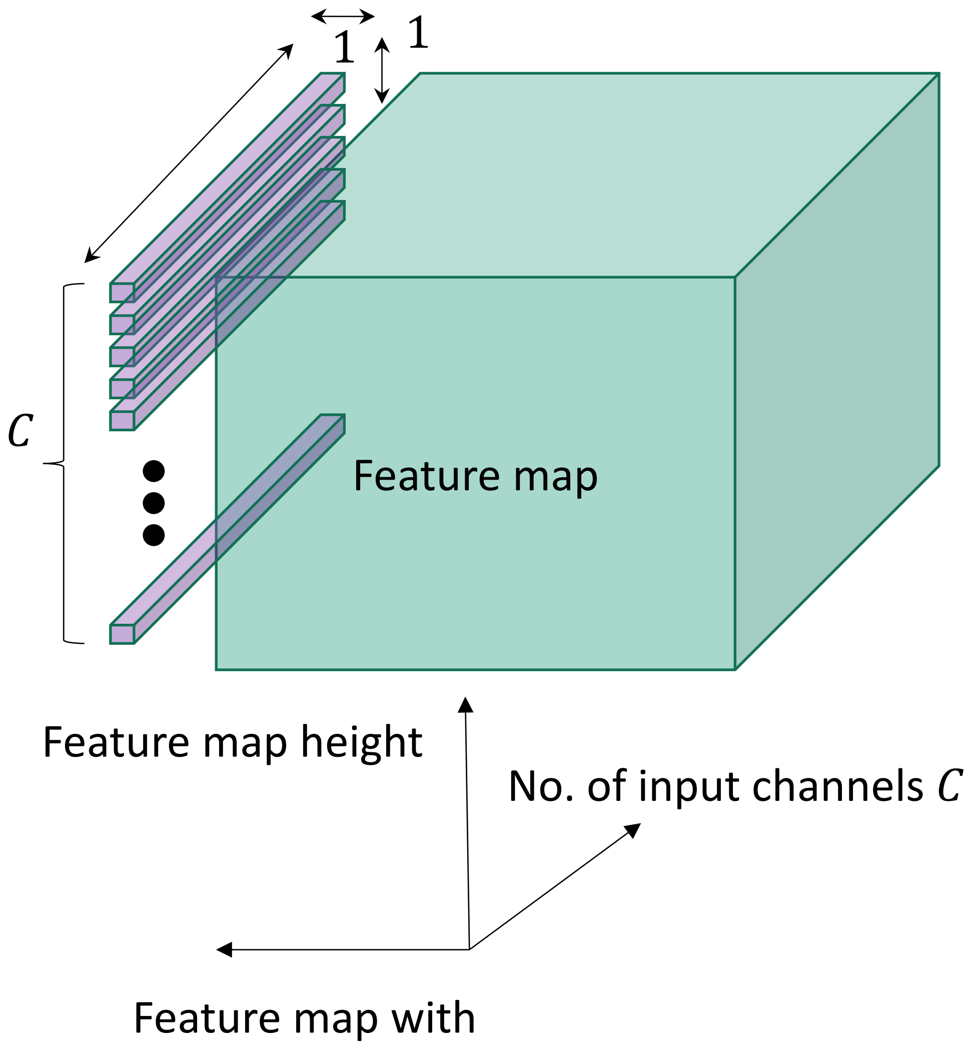
Smaller filters 1x1

- Also $1x1$ filters are possible
- Followed by a nonlinearity
- Why?

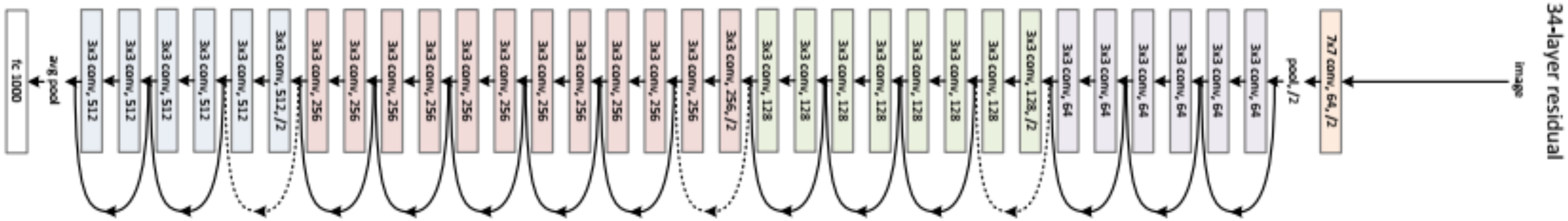


Smaller filters 1x1

- Also 1x1 filters are possible
- Followed by a nonlinearity
- Why?
- Increasing nonlinearities without affecting receptive field sizes
  - Linear transformation of the input channels
- Also, compression



ResNet



34-layer residual
image
7x7 conv, 64, /2
pool, /2
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 64
3x3 conv, 128, /2
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 128
3x3 conv, 256, /2
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 256
3x3 conv, 512, /2
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
3x3 conv, 512
avg pool
fc 1000

Some facts

- The first truly Deep Network, going deeper than 1,000 layers

- More importantly, the first Deep Architecture that proposed a novel concept on how to gracefully go deeper than a few dozen layers
  - ❑Not simply getting more GPUs, more training time, etc

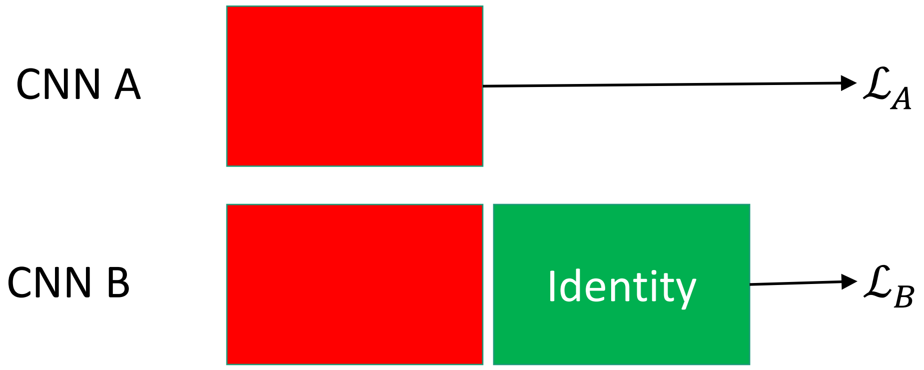
- Smashed Imagenet, with a 3.57% error (in ensembles)

- Won all object classification, detection, segmentation, etc. challenges

Hypothesis

- **Hypothesis:** Is it possible to have a very deep network at least as accurate as averagely deep networks?

- **Thought experiment:** Let's assume two Convnets A, B. They are almost identical, in that B is the same as A, with extra "identity" layers. Since identity layers pass the information unchanged, the errors of the two networks should be similar. Thus, there is a Convnet B, which is at least as good as Convnet A w.r.t. training error



Quiz: What looks weird?

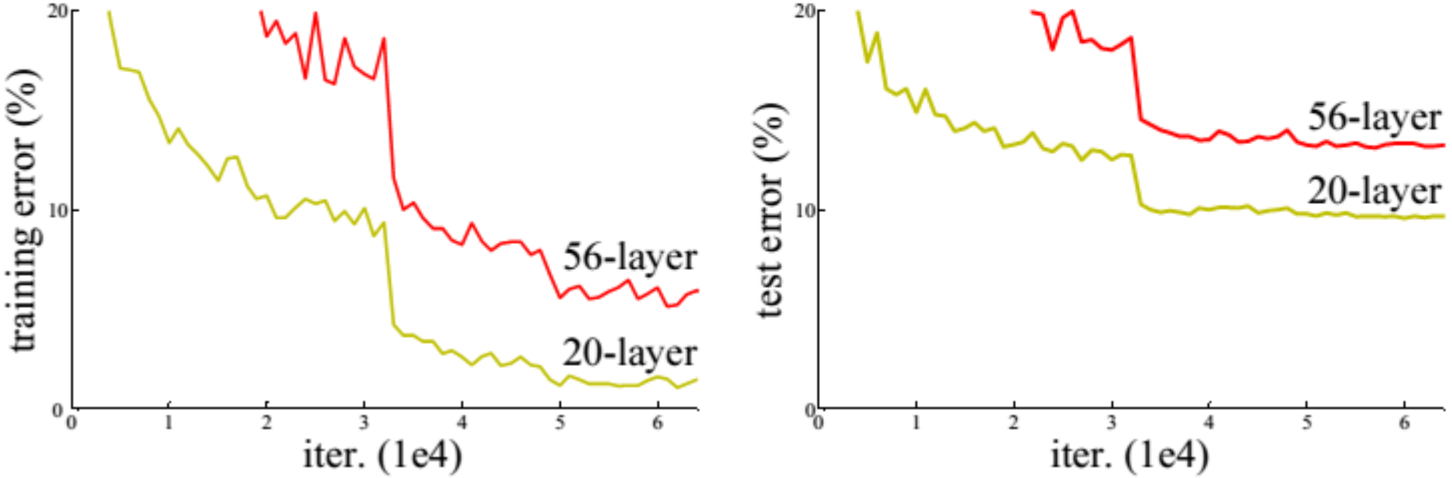


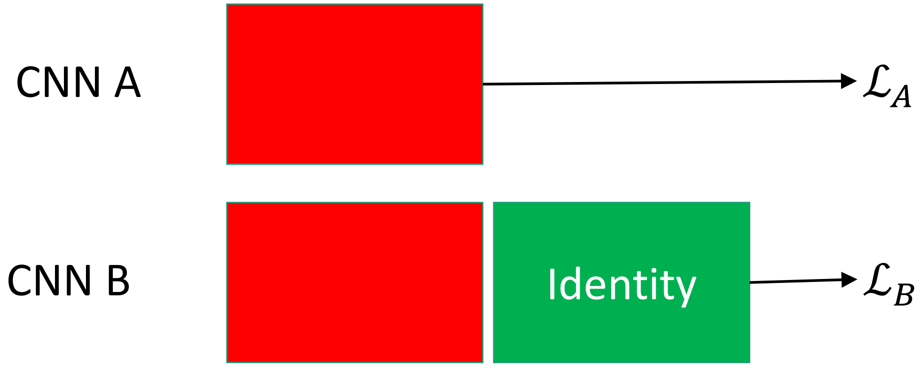
Figure 1. Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer "plain" networks. The deeper network has higher training error, and thus test error. Similar phenomena on ImageNet is presented in Fig. 4.

Testing hypothesis

- Adding identity layers increases **training error**!!
  - Training error, not testing error

- **Performance degradation** not caused by overfitting
  - Just the optimization task is harder

- Assuming optimizers are doing their job fine, it appears that not all networks are the same as easy to optimize



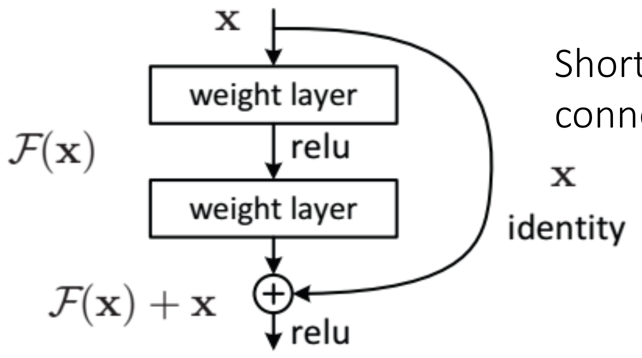
$\mathcal{L}_A^{train} < \mathcal{L}_B^{train}$!!!

ResNet: Main idea

- Layer models residual $F(x) = H(x) - x$ instead of $H(x)$

- If anything, the optimizer can simply set the weights to 0
  - This assumes that the identity mapping is indeed the optimal one

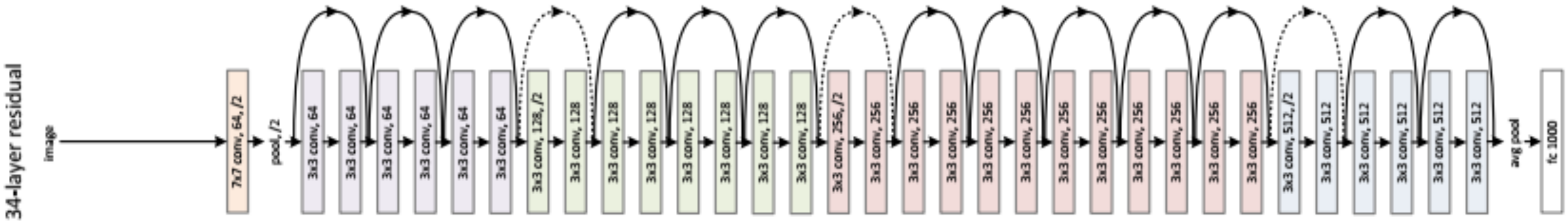
- Adding identity layers should lead to larger networks that have <u>at least</u> lower training error



Shortcut/skip connections

Figure 2. Residual learning: a building block.

Smooth propagation



$$x_{l+1} = x_l + \mathrm{F}(\mathrm{x}_l) \qquad x_{l+2} = x_{l+1} + \mathrm{F}(\mathrm{x}_{l+1}) \qquad \ldots \quad x_L = x_l + \sum_{i=l}^{L-1} F(x_i)$$

- Additive relation between x_l, x_L
  - Traditional NNs have multiplicative: $x_L = \prod_{i=l}^{L-1} W_i x_l$

- Smooth backprop: $\frac{\partial \mathcal{L}}{\partial x_l} = \frac{\partial \mathcal{L}}{\partial x_L} \cdot \frac{\partial x_L}{\partial x_l} = \frac{\partial \mathcal{L}}{\partial x_L}\left(1 + \frac{\partial}{\partial x_L} \sum_{i=l}^{L-1} F(x_i)\right)$
  - The loss closest to the output $\frac{\partial \mathcal{L}}{\partial x_L}$ is always there in the gradients

ResNet block

- $H(x) = F(x) + x$

- If dimensions don't match
  - Either zero padding
  - Or a projection layer to match dimensions

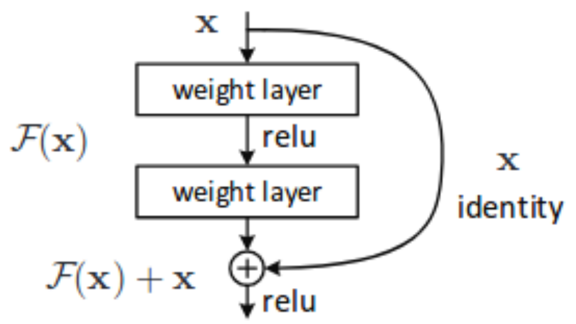
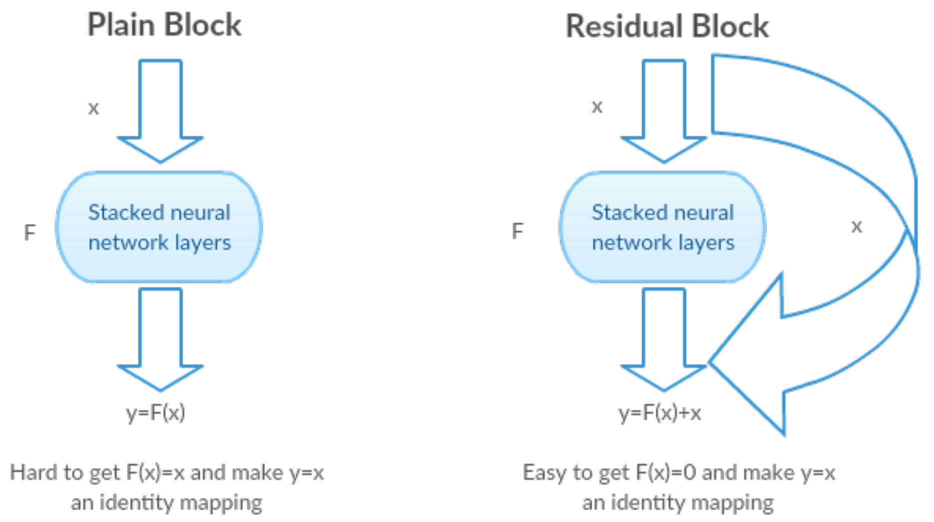


Figure 2. Residual learning: a building block.



No degradation anymore

- Without residual connections deeper networks are untrainable

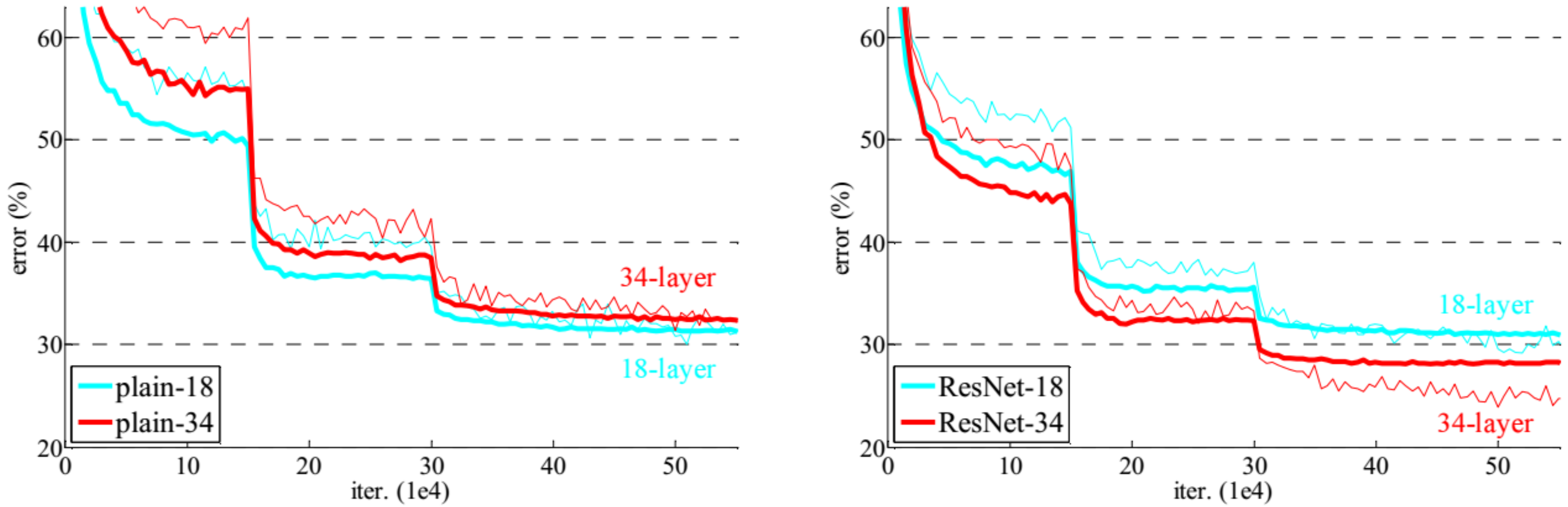


Figure 4. Training on **ImageNet**. Thin curves denote training error, and bold curves denote validation error of the center crops. Left: plain networks of 18 and 34 layers. Right: ResNets of 18 and 34 layers. In this plot, the residual networks have no extra parameter compared to their plain counterparts.

ResNet vs Highway Nets

- ResNet: $y = H(x) - x$

- Highway Nets: $y = H(x) \cdot T_x - x \cdot (1 - T_x)$

- ResNet ⊆ Highway Nets
  - ❑ResNet ≡ Highway Nets: $T_x \sim Binomial$ with $E[T_x] = 0.5$

- ResNet data independent
  - ❑Curse or blessing, depending on point of view
  - ❑Definitely simpler

ResNet breaks records

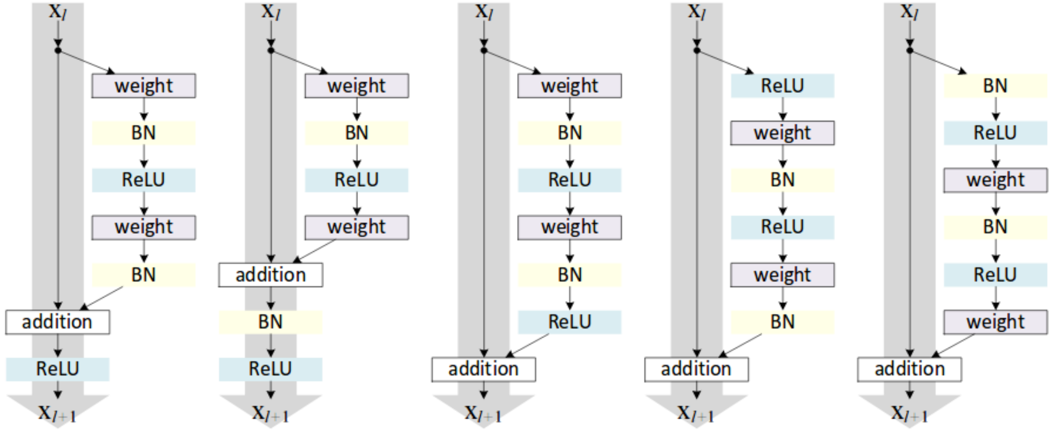
- Ridiculously low error in ImageNet

- Up to 1000 layers ResNets trained
  - ❑Previous deepest network ~30-40 layers on simple datasets

method	top-5 err. (**test**)
VGG [41] (ILSVRC'14)	7.32
GoogLeNet [44] (ILSVRC'14)	6.66
VGG [41] (v5)	6.8
PReLU-net [13]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

Table 5. Error rates (%) of **ensembles**. The top-5 error is on the test set of ImageNet and reported by the test server.

ResNext



(a) original (b) BN after addition (c) ReLU before addition (d) ReLU-only pre-activation (e) **full pre-activation**

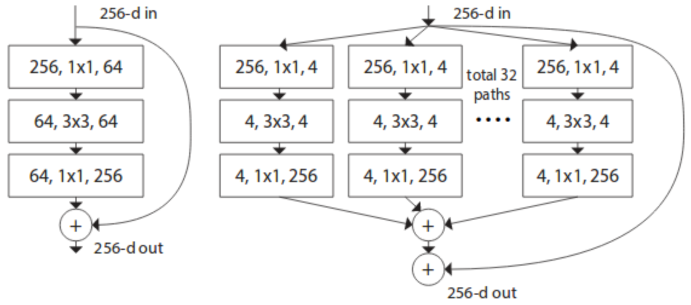


Figure 1. **Left**: A block of ResNet [14]. **Right**: A block of ResNeXt with cardinality = 32, with roughly the same complexity. A layer is shown as (# in channels, filter size, # out channels).

case	Fig.	ResNet-110	ResNet-164
original Residual Unit [1]	Fig. 4(a)	6.61	5.93
BN after addition	Fig. 4(b)	8.17	6.50
ReLU before addition	Fig. 4(c)	7.84	6.14
ReLU-only pre-activation	Fig. 4(d)	6.71	5.91
full pre-activation	Fig. 4(e)	**6.37**	**5.46**

	setting	top-1 err (%)	top-5 err (%)
1× complexity references:			
ResNet-101	1 × 64d	22.0	6.0
ResNeXt-101	32 × 4d	21.2	5.6
2× complexity models follow:			
ResNet-**200** [15]	1 × 64d	21.7	5.8
ResNet-101, wider	1 × **100**d	21.3	5.7
ResNeXt-101	**2** × 64d	20.7	5.5
ResNeXt-101	**64** × 4d	**20.4**	**5.3**

Table 4. Comparisons on ImageNet-1K when the number of FLOPs is increased to 2× of ResNet-101's. The error rate is evaluated on the single crop of 224×224 pixels. The highlighted factors are the factors that increase complexity.

Some observations

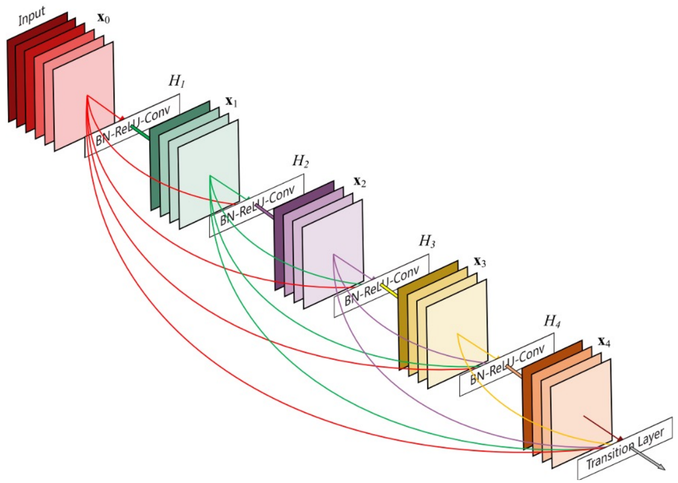
- BatchNorms absolutely necessary because of vanishing gradients
- Networks with skip connections (like ResNets) converge faster than the same network without skip connections
- Identity shortcuts cheaper and almost equal to project shortcuts

DenseNets

- Add skip connections to multiple forward layers

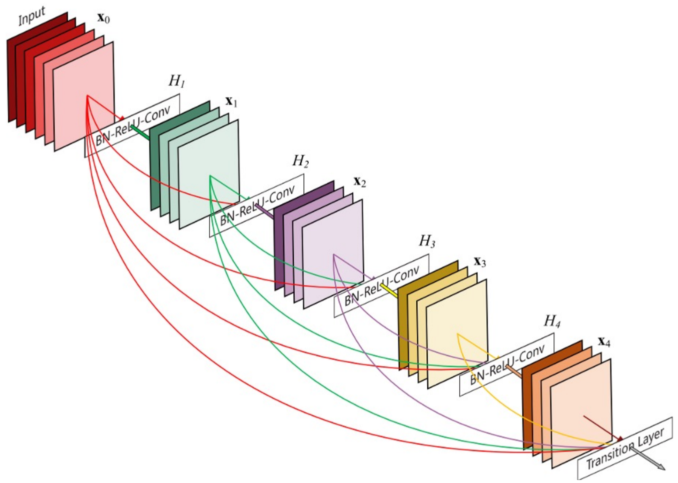
$$y = h(x_l, x_{l-1}, \ldots, x_{l-n})$$

- Why?



DenseNets

- Add skip connections to multiple forward layers
$$y = h(x_l, x_{l-1}, \dots, x_{l-n})$$
- Assume layer 1 captures edges, while layer 5 captures faces (and other stuff)
- Why not have a layer that combines both faces and edges (e.g. to model scarred faces)
- Standard ConvNets do not allow for this
  - Layer 6 combines only layer 5 patterns, not lower



Inception

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

Basic idea

- Problem ?

Picture credit: Bharath Raj

Basic idea

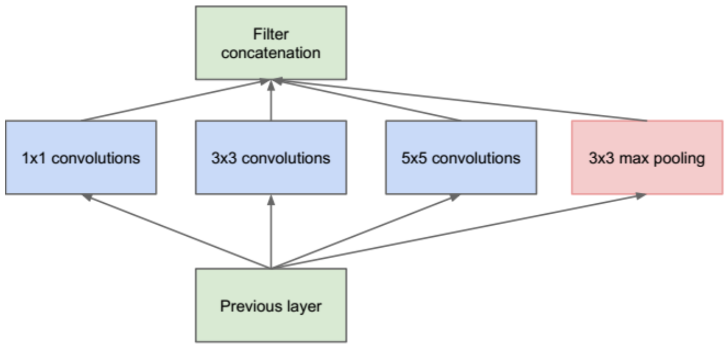
- Problem ?
- Salient parts have great variation in sizes
- Hence, the receptive fields should vary in size accordingly
- Naively stacking convolutional operations is expensive
- Very deep nets are prone to overfitting

Picture credit: Bharath Raj

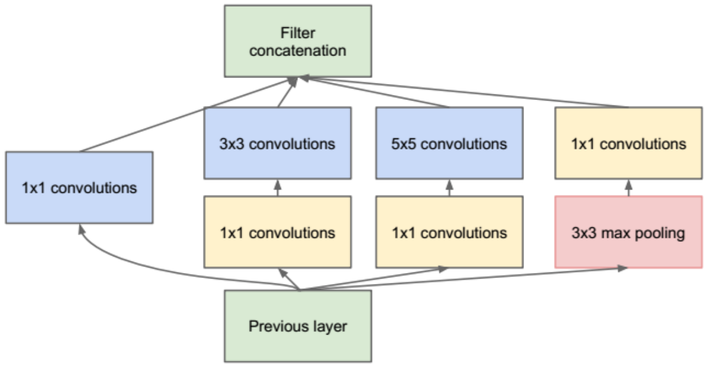
Inception module

- Multiple kernel filters of different sizes ($1\times1, 3\times3, 5\times5$)
  - Naïve version

•Problem?



(a) Inception module, naïve version



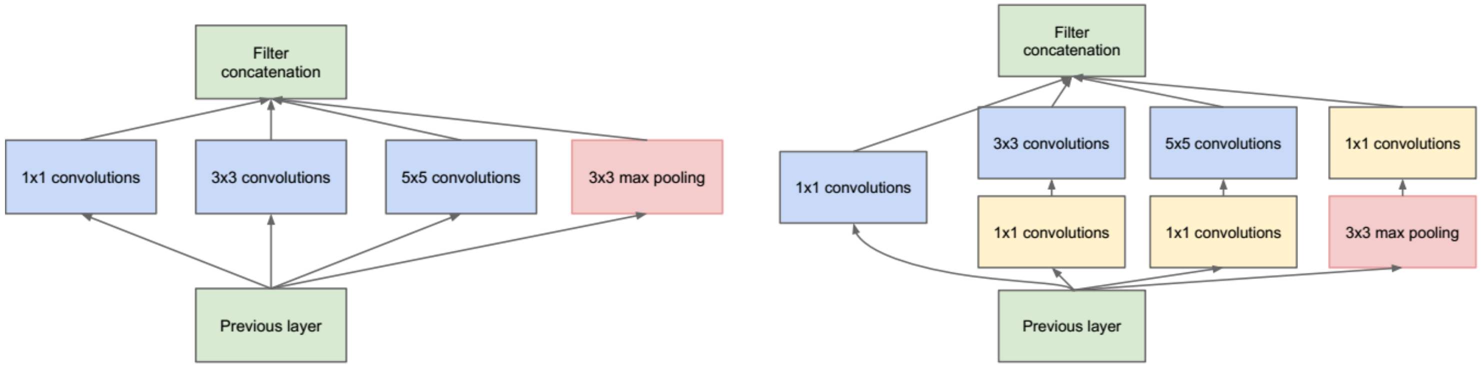
(b) Inception module with dimension reductions

Picture credit: [Bharath Raj]

Inception module

- Multiple kernel filters of different sizes (1×1, 3×3, 5×5)
  - Naïve version

- Problem?
  - Very expensive!
- Add intermediate 1×1 convolutions

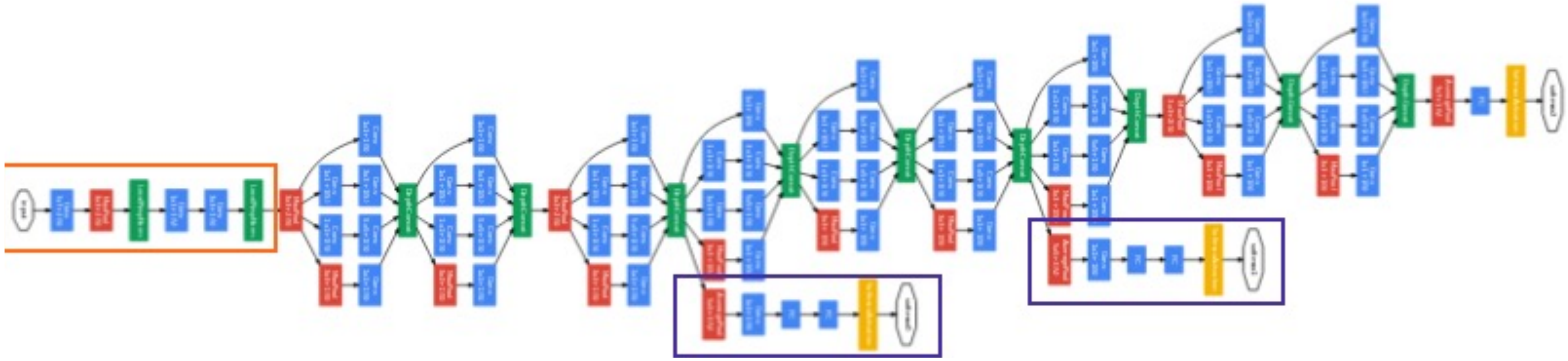


(a) Inception module, naïve version (b) Inception module with dimension reductions

Picture credit: [Bharath Raj]

Architecture

- 9 Inception Modules
- 22 layers deep (27 with the pooling layers)
- Global average pooling at the end of last Inception Module
- 6.67% Imagenet error, compared to 18.2% of Alexnet



Picture credit: [Bharath Raj]

Houston, we have a problem

Problem: Vanishing gradients

- The network was too deep (at the time)

- Roughly speaking, backprop is lots of matrix multiplications

$$\frac{\partial \mathcal{L}}{\partial w^l} = \frac{\partial \mathcal{L}}{\partial a^L} \cdot \frac{\partial a^L}{\partial a^{L-1}} \cdot \frac{\partial a^{L-1}}{\partial a^{L-2}} \cdot \ \dots \cdot \frac{\partial a^l}{\partial w^l}$$

- Many of intermediate terms < 1 → the final $\frac{\partial \mathcal{L}}{\partial w^l}$ gets extremely small
- Extremely small gradient → ?

Problem: Vanishing gradients

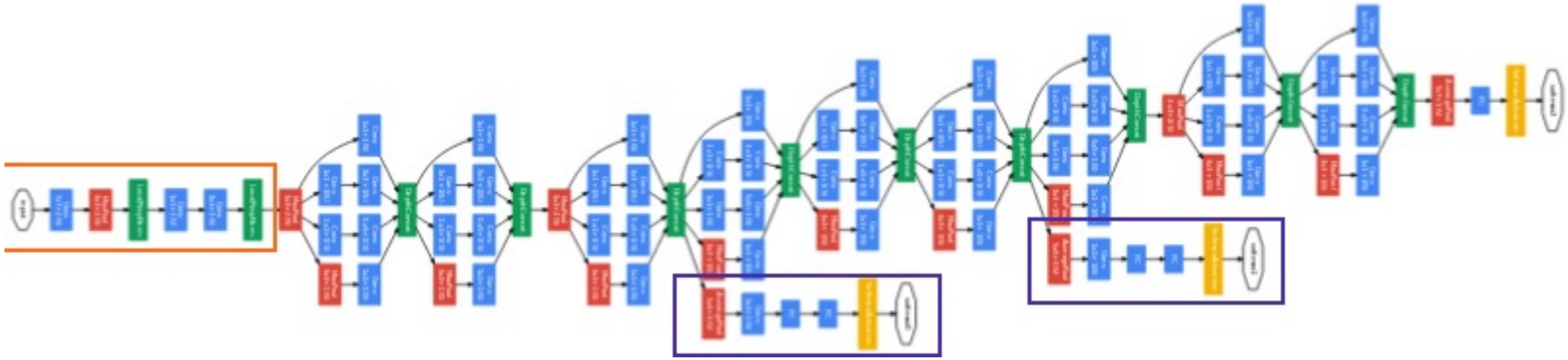
- The network was too deep (at the time)
- Roughly speaking, backprop is lots of matrix multiplications

$$\frac{\partial \mathcal{L}}{\partial w^l} = \frac{\partial \mathcal{L}}{\partial a^L} \cdot \frac{\partial a^L}{\partial a^{L-1}} \cdot \frac{\partial a^{L-1}}{\partial a^{L-2}} \cdot \ \dots \cdot \frac{\partial a^l}{\partial w^l}$$

- Many of intermediate terms < 1 → the final $\frac{\partial \mathcal{L}}{\partial w^l}$ gets extremely small
- Extremely small gradient → ?
- Many of intermediate terms < 1 → the final $\frac{\partial \mathcal{L}}{\partial w^l}$ gets extremely small
- Extremely small gradient → Extremely slow learning

Architecture

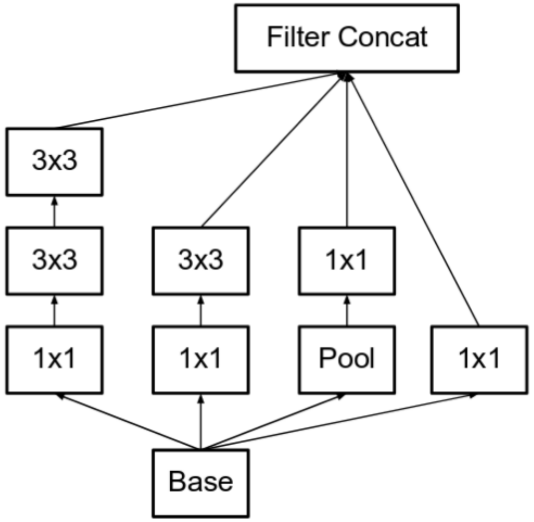
- 9 Inception Modules
- 22 layers deep (27 with the pooling layers)
- Global average pooling at the end of last Inception Module
- Because of the increased depth → **Vanishing gradients**
- Inception solution to vanishing gradients: **intermediate classifiers**
  - Intermediate classifiers removed after training

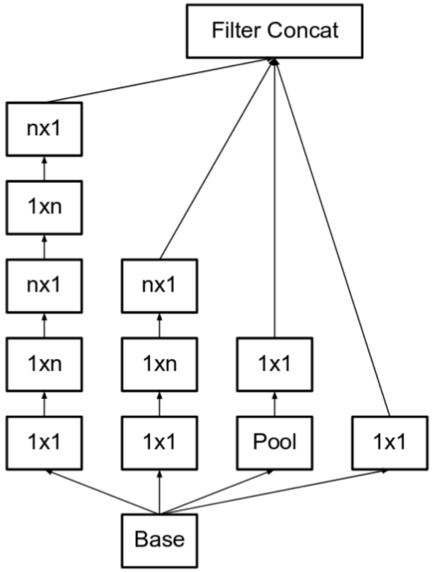


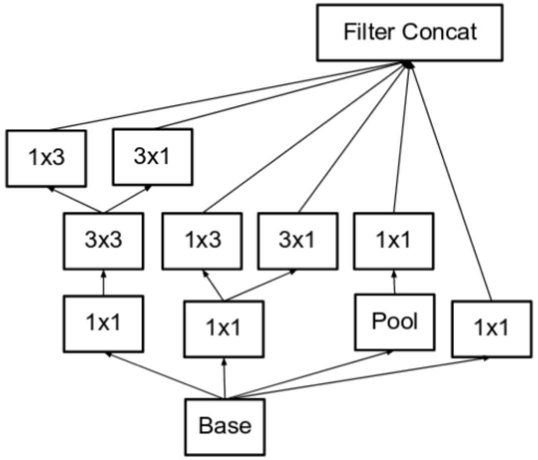
Picture credit: [Bharath Raj]

Inceptions v2, v3, v4

- Factorize 5×5 in two 3×3 filters
- Factorize $n \times n$ in two $n \times 1$ and $1 \times n$ filters (quite a lot cheaper)
- Make nets wider
- RMSprop, BatchNorms, …







Picture credit: [Bharath Raj]

Neural Architecture Search

- It is also possible to learn the neural architecture

- Problem?

Neural Architecture Search

- It is also possible to learn the neural architecture

- Problem?
- Architectures/graphs are discrete structures → Backprop?

- Still, some very interesting workarounds have been proposed in practice

- Will it work for you? If you are Facebook or Google, yes!

Evolutionary Search for NAS

- DARTS: Differentiable Architecture Search, Liu et al., 2018
- Efficient Neural Architecture Search via Parameter Sharing, Pham et al., 2018
- Evolving Space-Time Neural Architectures for Videos, Piergiovanni et al. 2018
- Regularized Evolution for Image Classifier Architecture Search, Real et al., 2019

Algorithm 1 Evolutionary search algorithm

function SEARCH
  Randomly initialize the population, P
  Evaluate each individual in P
  for $i <$ number of evolutionary rounds **do**
    $S =$ random sample of 25 individuals
    $parent =$ the most fit individual in S
    $child = parent$
    for $\max(\lceil d - \frac{i}{r} \rceil, 1)$ **do**
      $child = mutate(child)$
    end for
    evaluate $child$ and add to population
    remove least fit individual from population
  end for
end function

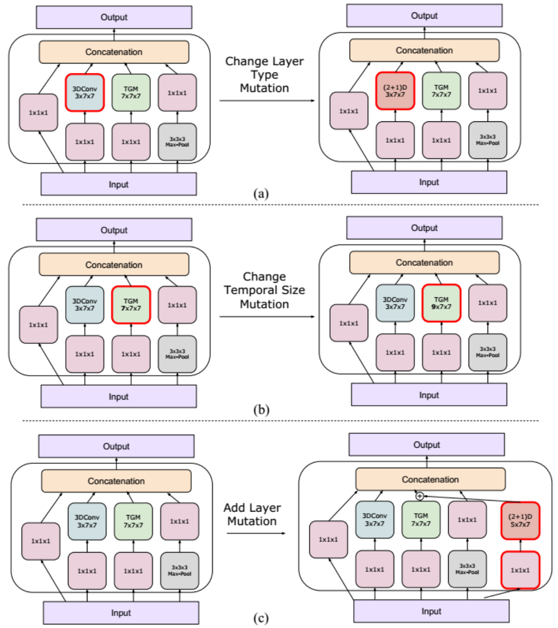
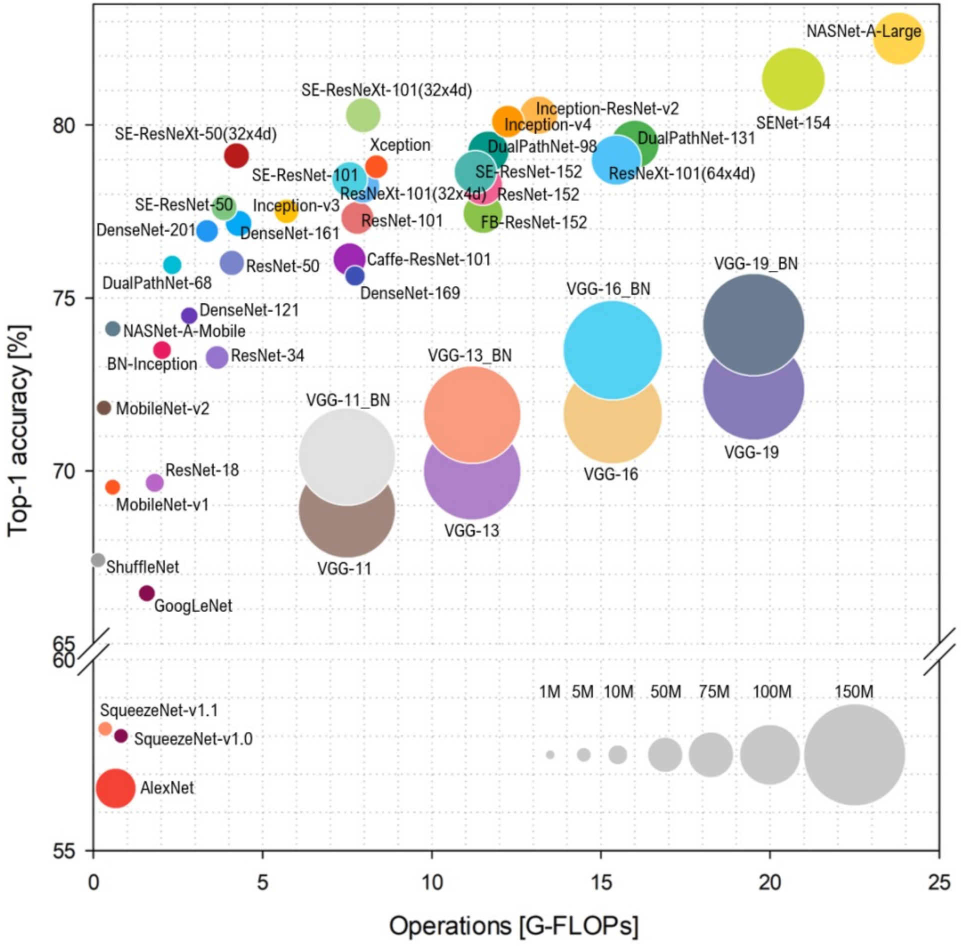


Figure 5. Example mutations applied to a module, including (a) layer type change, (b) filter length change, and (c) layer addition.

State-of-the-art



Bianco et al., Benchmark Analysis of Representative Deep Neural Network Architectures, IEEE Access 2018