Popular DNNs and Datasets

ISCA Tutorial (2019)

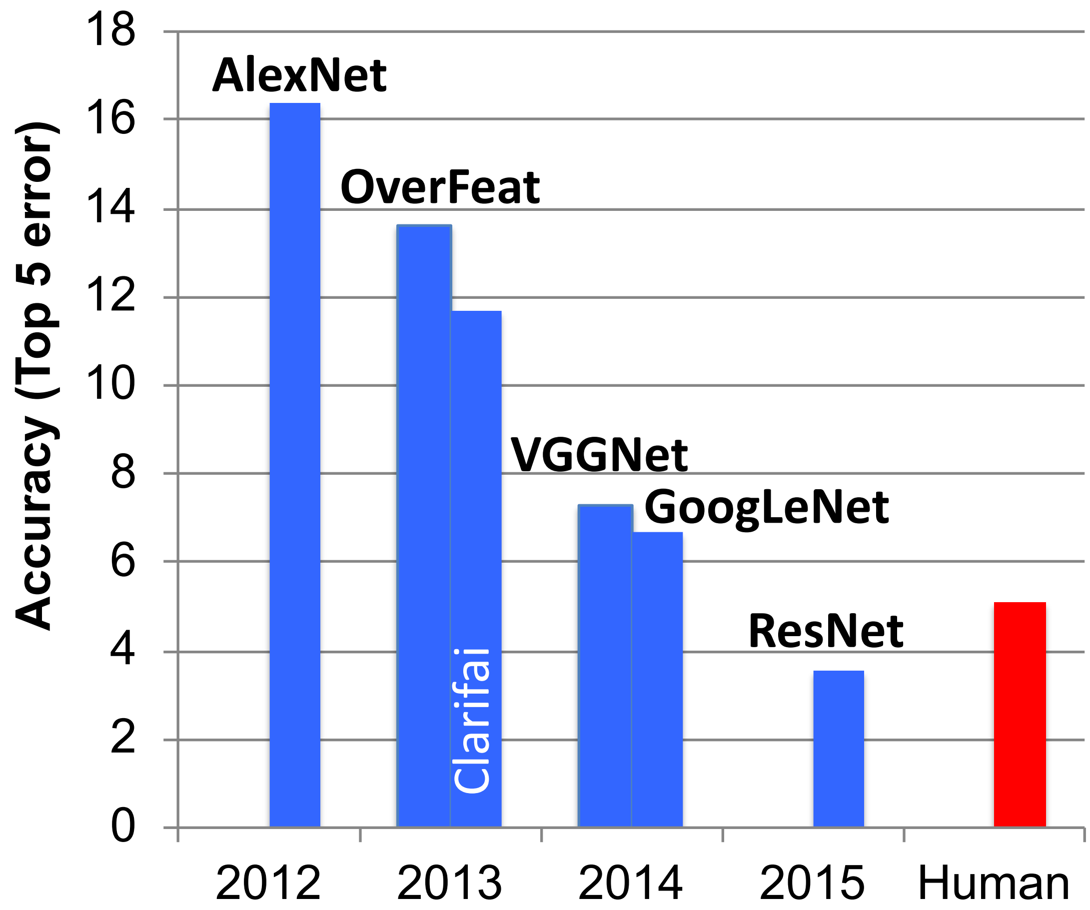
Website: http://eyeriss.mit.edu/tutorial.html

Joel Emer, Vivienne Sze, Yu-Hsin Chen

Popular DNNs

- LeNet (1998)
- AlexNet (2012)
- OverFeat (2013)
- VGGNet (2014)
- GoogleNet (2014)
- ResNet (2015)

ImageNet: Large Scale Visual Recognition Challenge (ILSVRC)



[O. Russakovsky et al., IJCV 2015]

MNIST

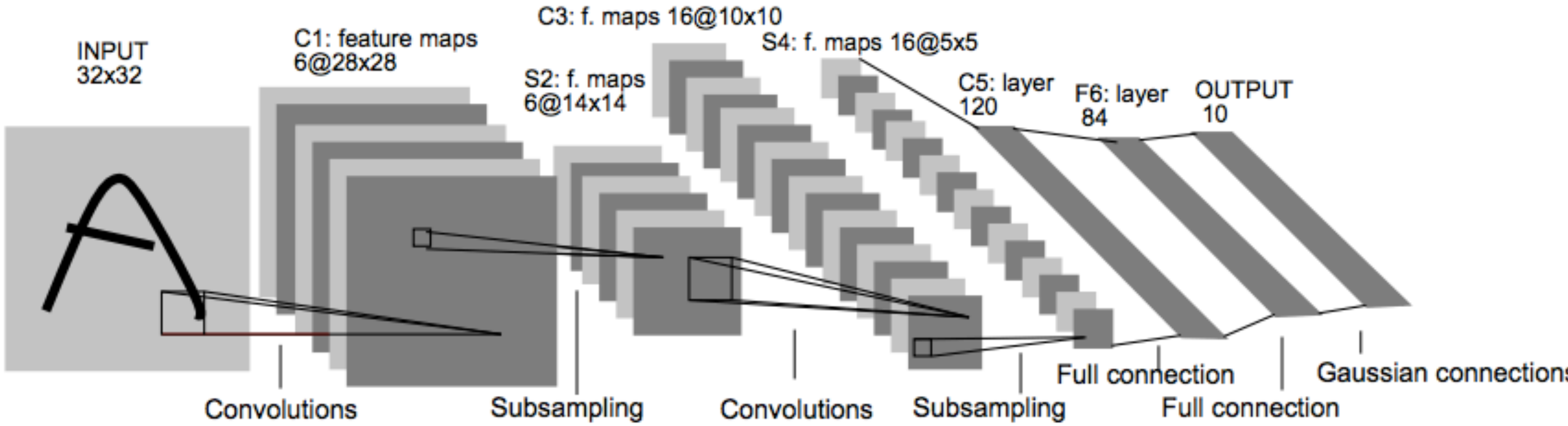
Digit Classification
28x28 pixels (B&W)
10 Classes
60,000 Training
10,000 Testing

http://yann.lecun.com/exdb/mnist/

LeNet-5

CONV Layers: 2
Fully Connected Layers: 2
Weights: 60k
MACs: 341k
Sigmoid used for non-linearity

Digit Classification!
(MNIST Dataset)



six
5x5 filters

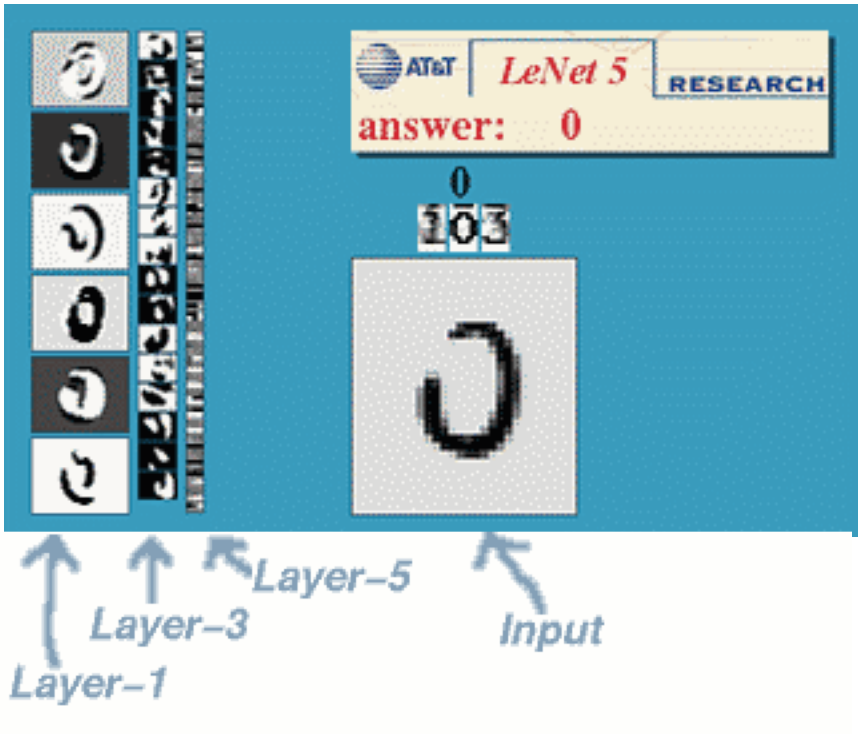
2x2
average
pooling

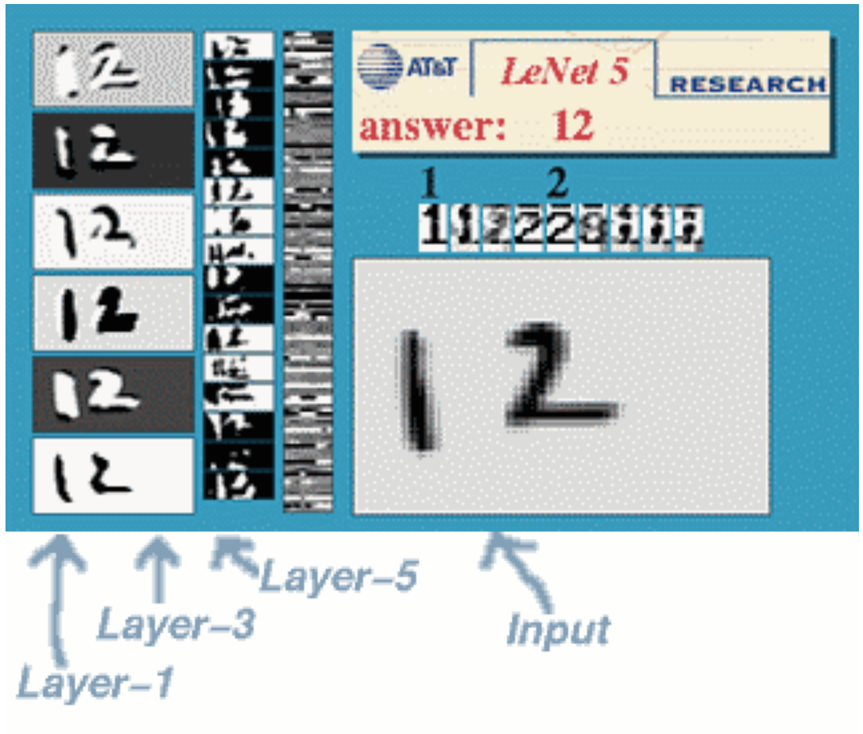
sixteen
5x5 filters

2x2
average
pooling

[Lecun et al., Proceedings of the IEEE, 1998]

LeNet-5





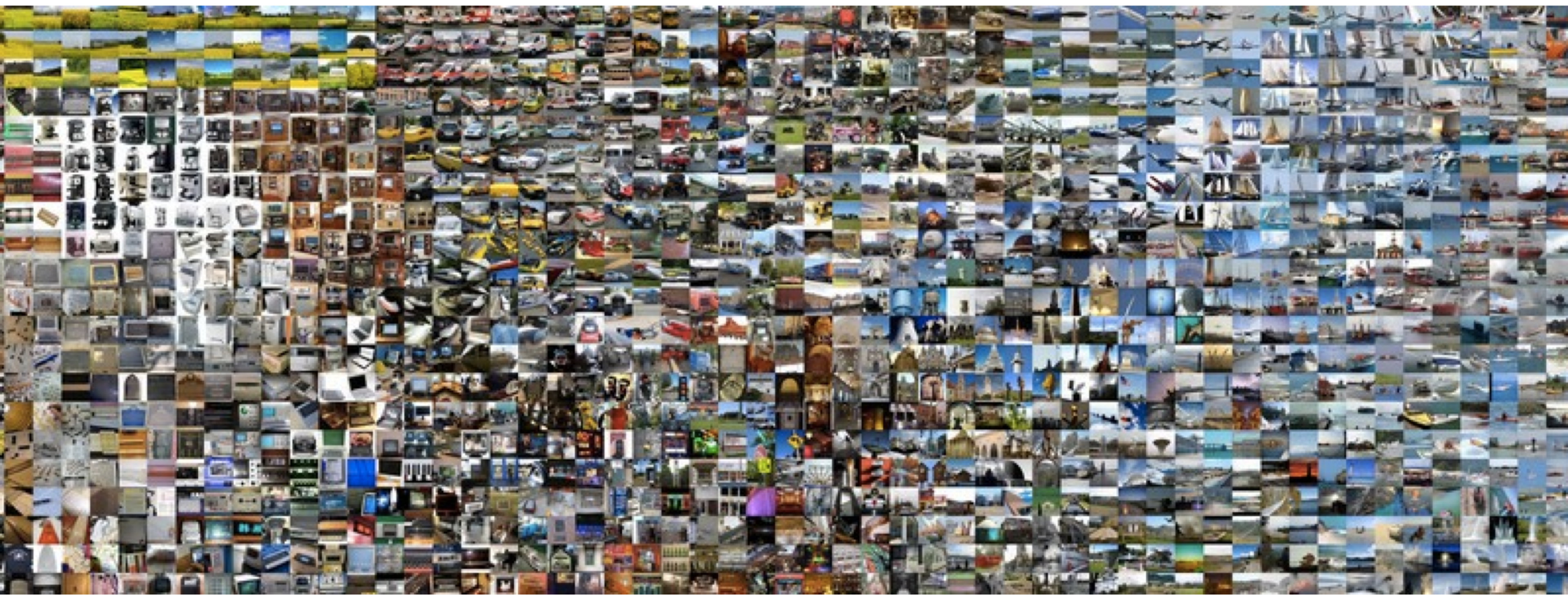
http://yann.lecun.com/exdb/lenet/

IMAGENET

Image Classification
~256x256 pixels (color)
1000 Classes
1.3M Training
100,000 Testing (50,000 Validation)

For **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)** accuracy of classification task reported based on top-1 and top-5 error

Image Source: http://karpathy.github.io/



http://www.image-net.org/challenges/LSVRC/

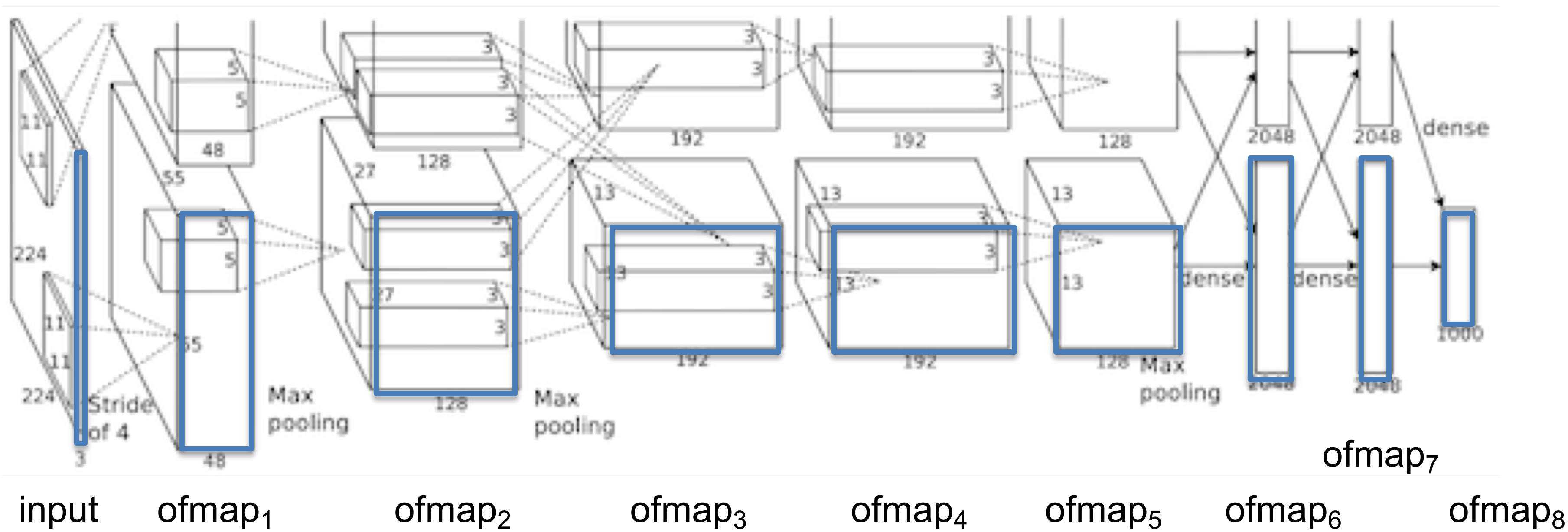
AlexNet

CONV Layers: 5
Fully Connected Layers: 3
Weights: 61M
MACs: 724M
ReLU used for non-linearity

ILSCVR12 Winner

Uses Local Response Normalization (LRN)

[Krizhevsky et al., NeurIPS 2012]



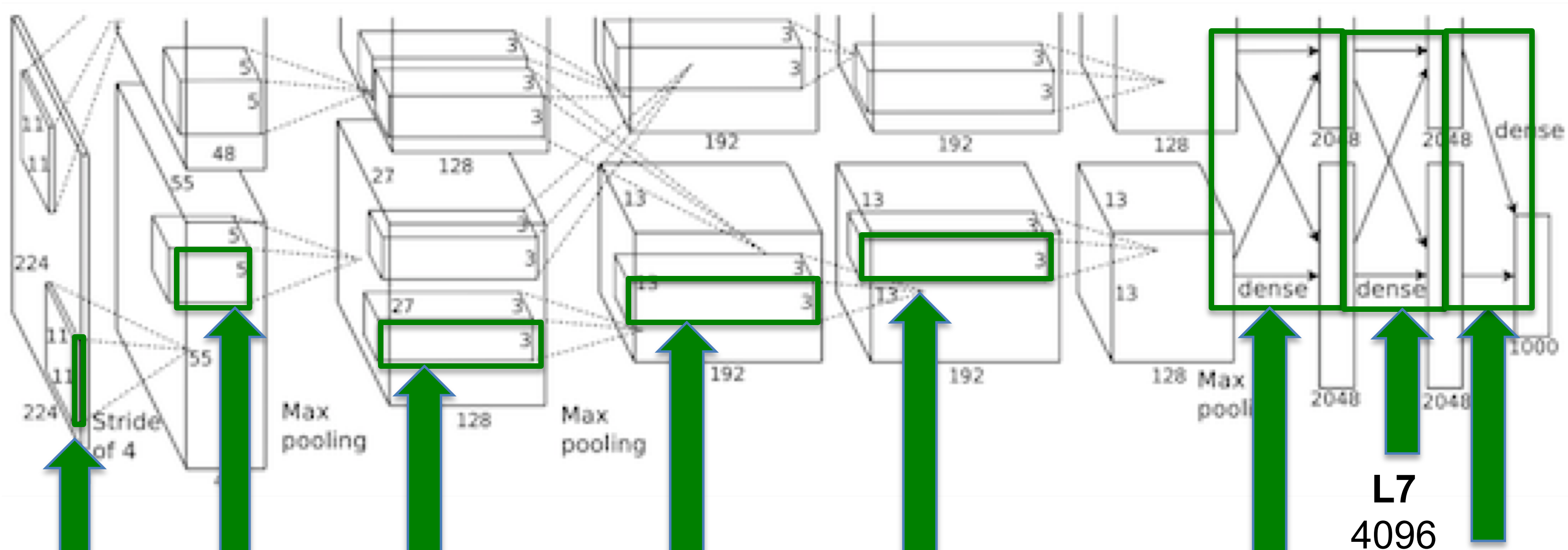
AlexNet

CONV Layers: 5
Fully Connected Layers: 3
Weights: 61M
MACs: 724M
ReLU used for non-linearity

ILSCVR12 Winner

Uses Local Response Normalization (LRN)

[Krizhevsky et al., NeurIPS 2012]



L1 96 filters (11x11x3)

L2 256 filters (5x5x48)

L3 384 filters (3x3x256)

L4 384 filters (3x3x192)

L5 384 filters (3x3x192)

L6 4096 filters (13x13x256)

L7 4096 (4096x1)

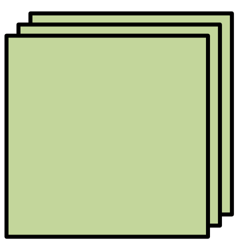
L8 1000 (4096x1)

Large Sizes with Varying Shapes

AlexNet Convolutional Layer Configurations

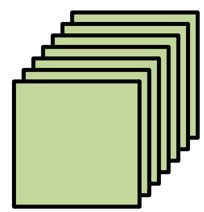
Layer	Filter Size (RxS)	# Filters (M)	# Channels (C)	Stride
1	11x11	96	3	4
2	5x5	256	48	1
3	3x3	384	256	1
4	3x3	384	192	1
5	3x3	256	192	1

Layer 1



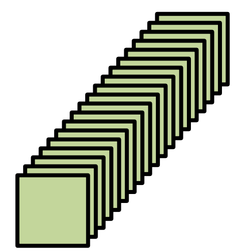
34k Params
105M MACs

Layer 2



307k Params
224M MACs

Layer 3



885k Params
150M MACs

[Krizhevsky et al., NIPS 2012]

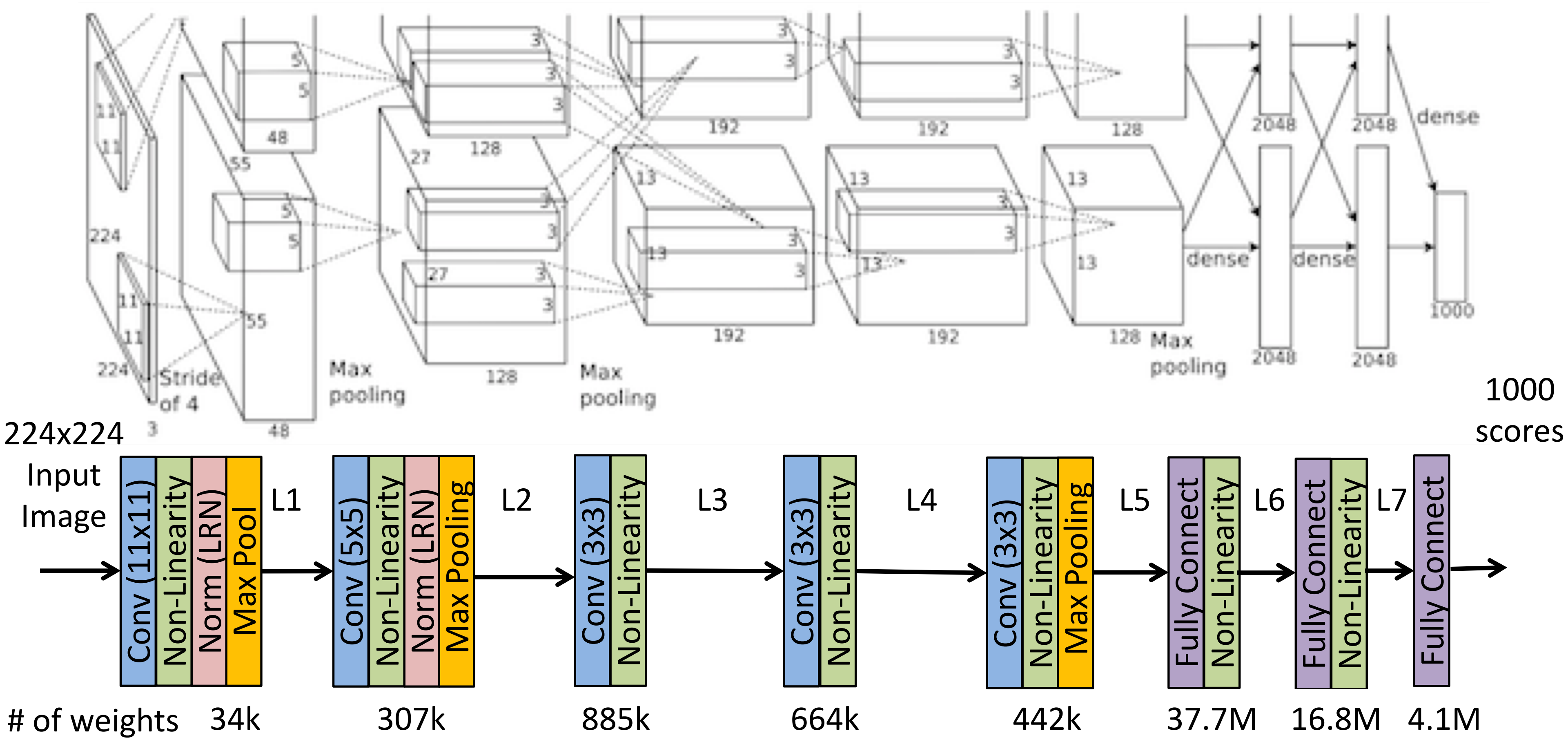
AlexNet

CONV Layers: 5
Fully Connected Layers: 3
Weights: 61M
MACs: 724M
ReLU used for non-linearity

ILSCVR12 Winner

Uses Local Response Normalization (LRN)

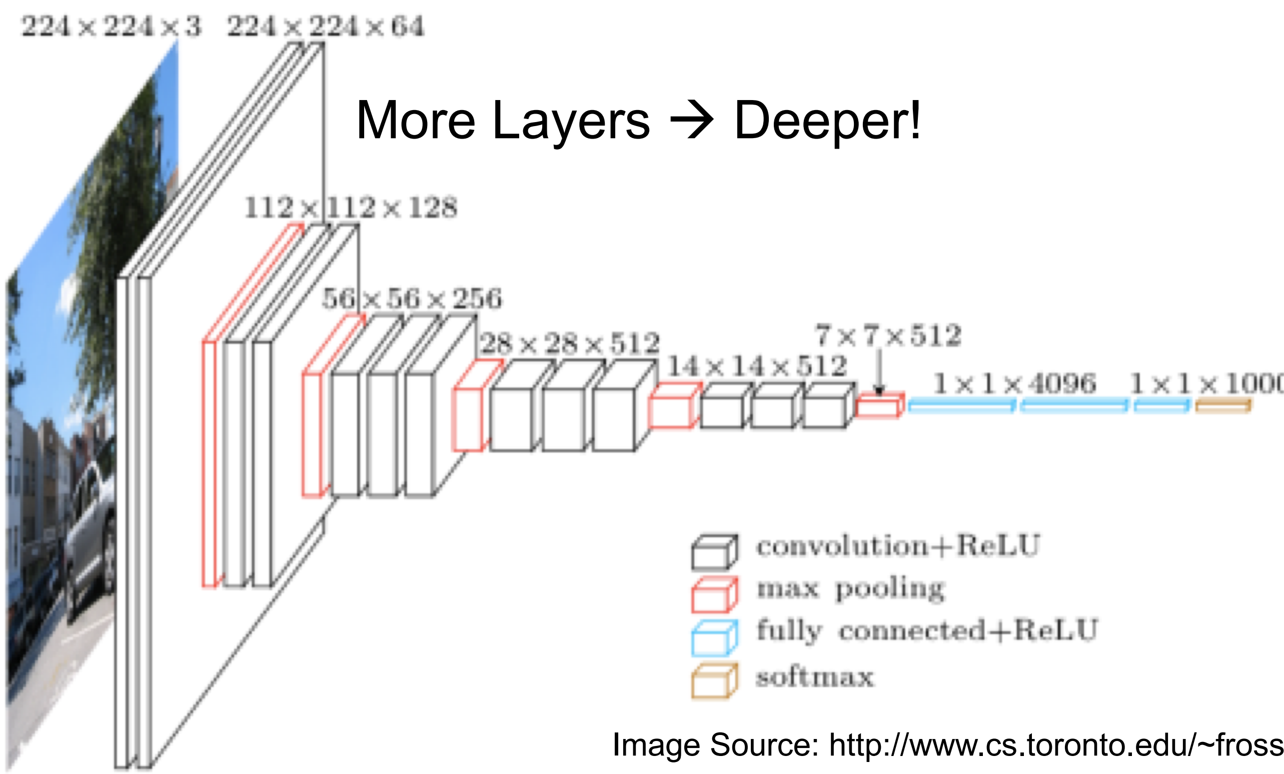
[Krizhevsky et al., NeurIPS 2012]



VGG-16

CONV Layers: 13
Fully Connected Layers: 3
Weights: 138M
MACs: 15.5G

Also, 19 layer version



More Layers → Deeper!

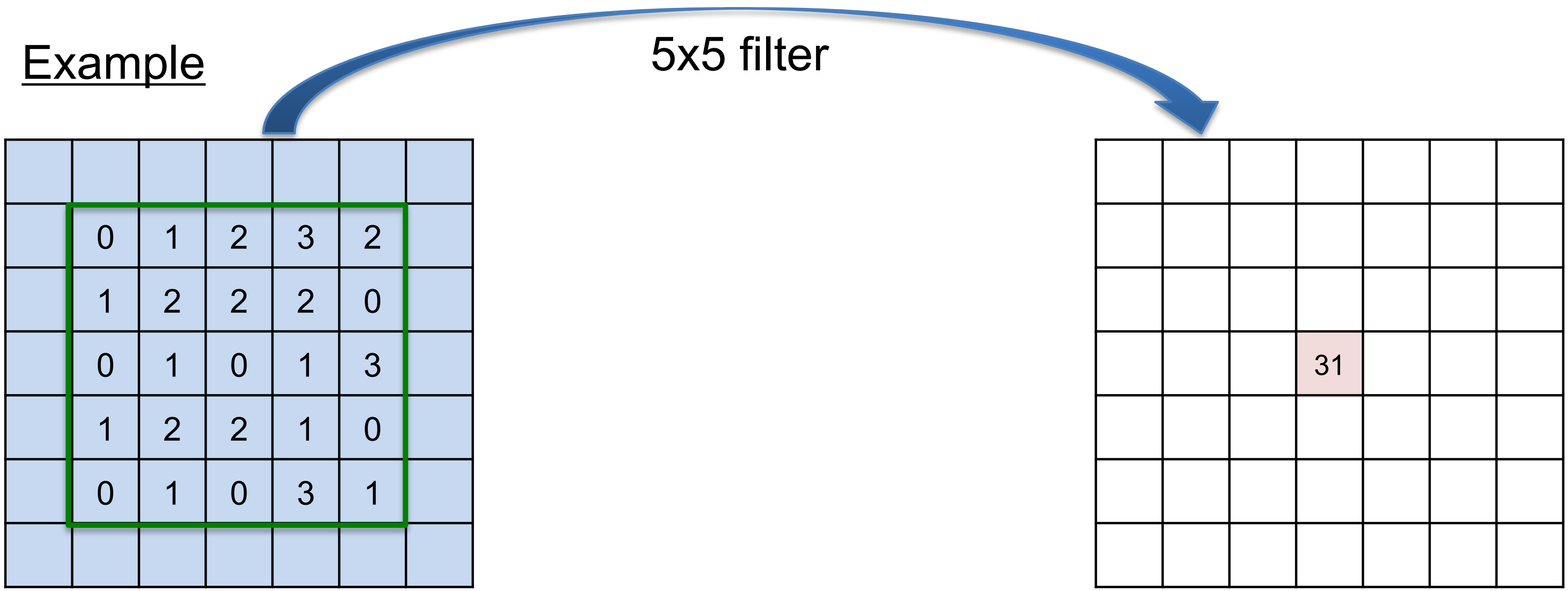
Image Source: http://www.cs.toronto.edu/~frossard/post/vgg16/

[Simonyan et al., arXiv 2014, ICLR 2015]

Stacked Filters

- Deeper network means more weights
- Use stack of smaller filters (3x3) to cover the same receptive field with fewer filter weights

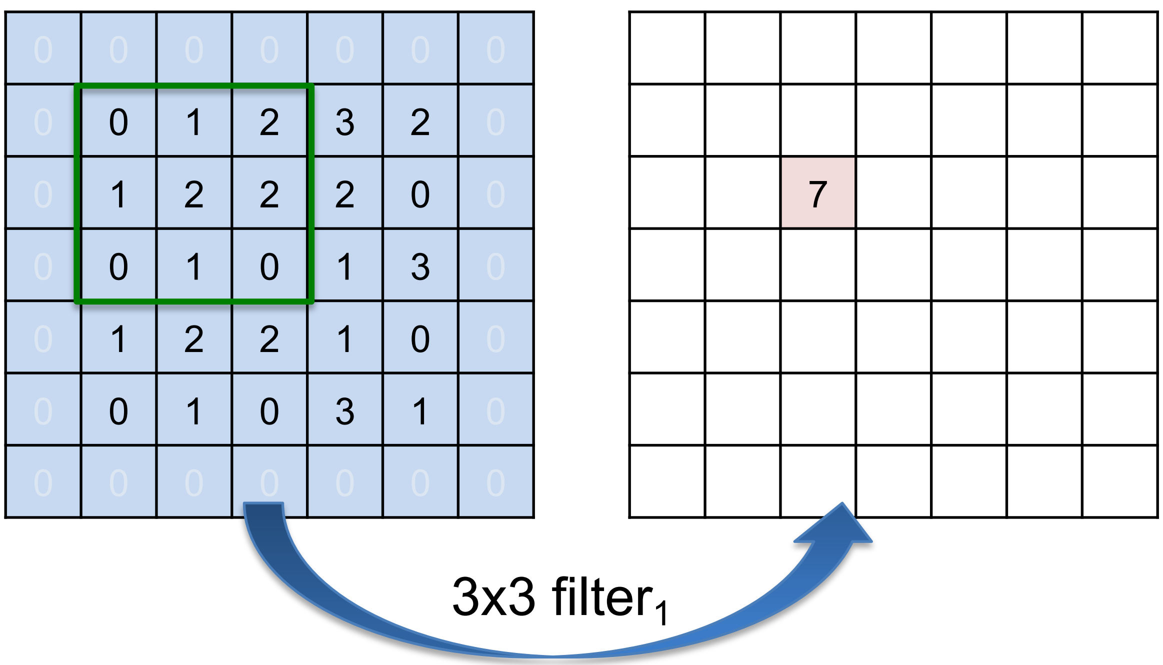
<u>Example</u>



Stacked Filters

- Deeper network means more weights
- Use stack of smaller filters (3x3) to cover the same receptive field with fewer filter weights

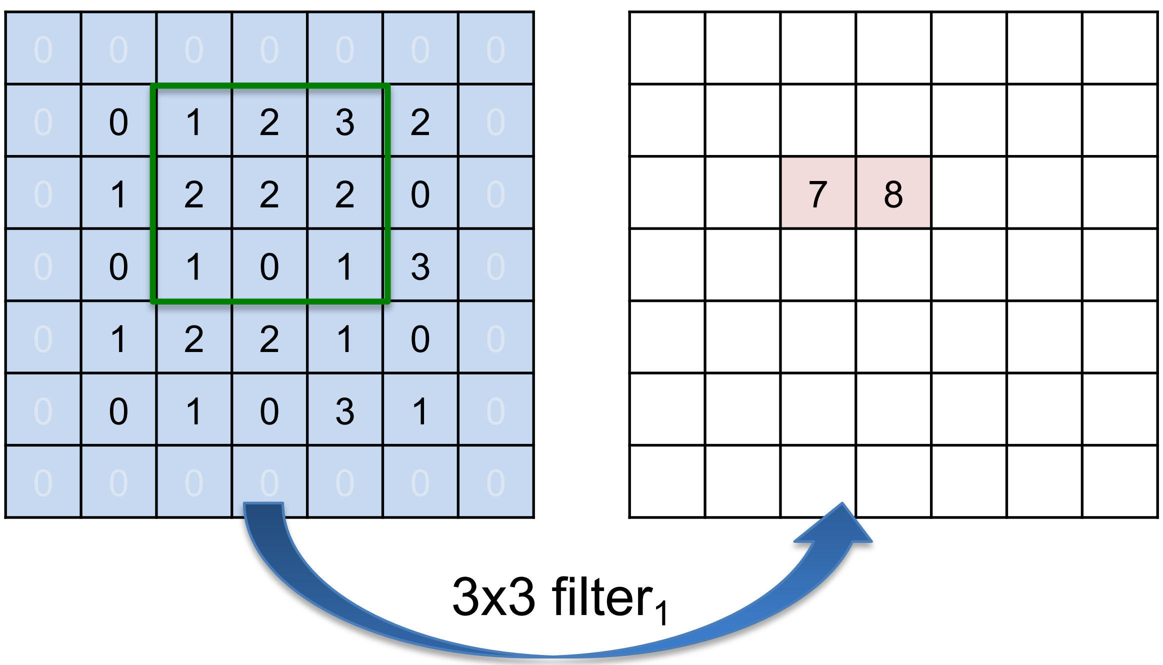
<u>Example</u>



Stacked Filters

- Deeper network means more weights
- Use stack of smaller filters (3x3) to cover the same receptive field with fewer filter weights

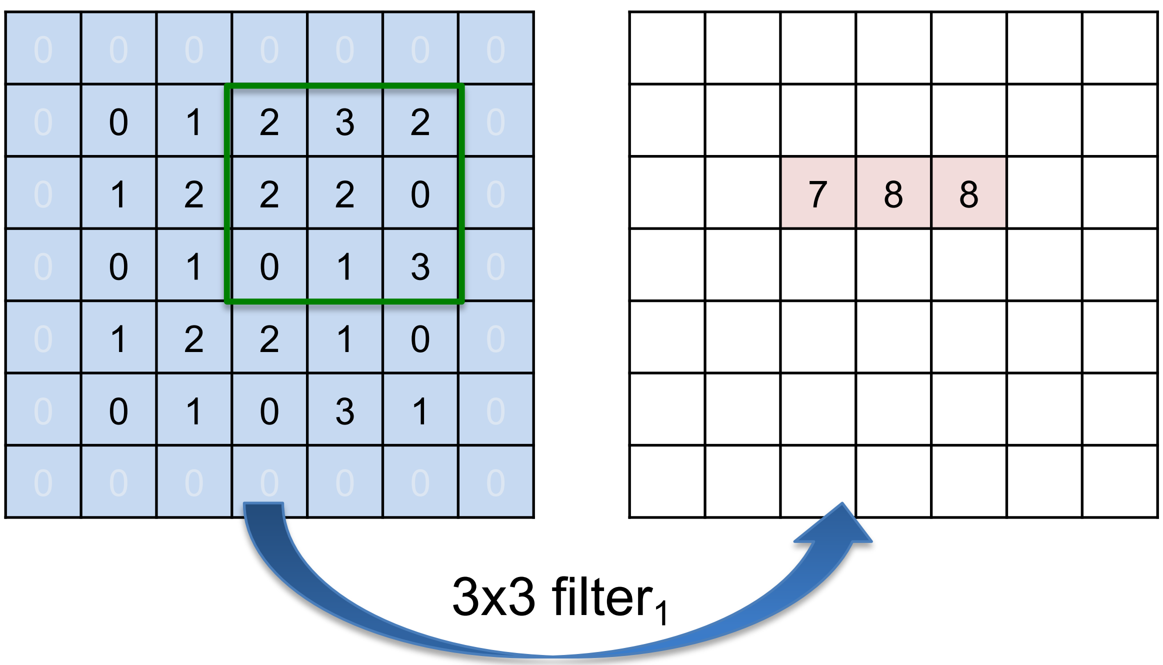
Example



Stacked Filters

- Deeper network means more weights
- Use stack of smaller filters (3x3) to cover the same receptive field with fewer filter weights

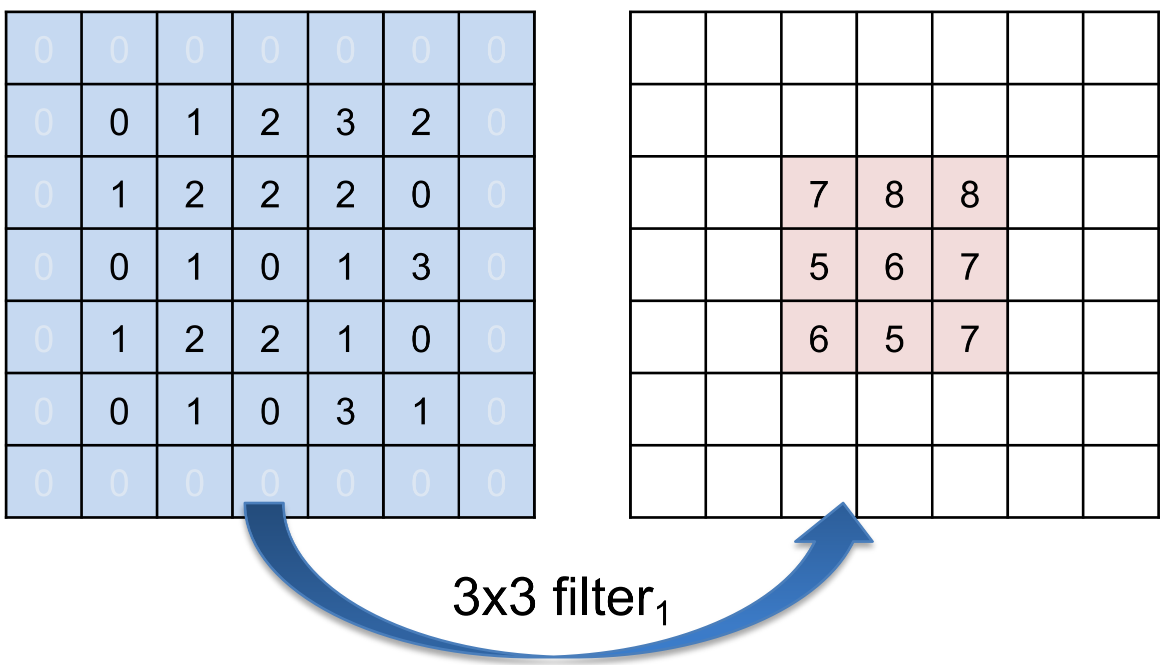
Example



Stacked Filters

- Deeper network means more weights
- Use stack of smaller filters (3x3) to cover the same receptive field with fewer filter weights

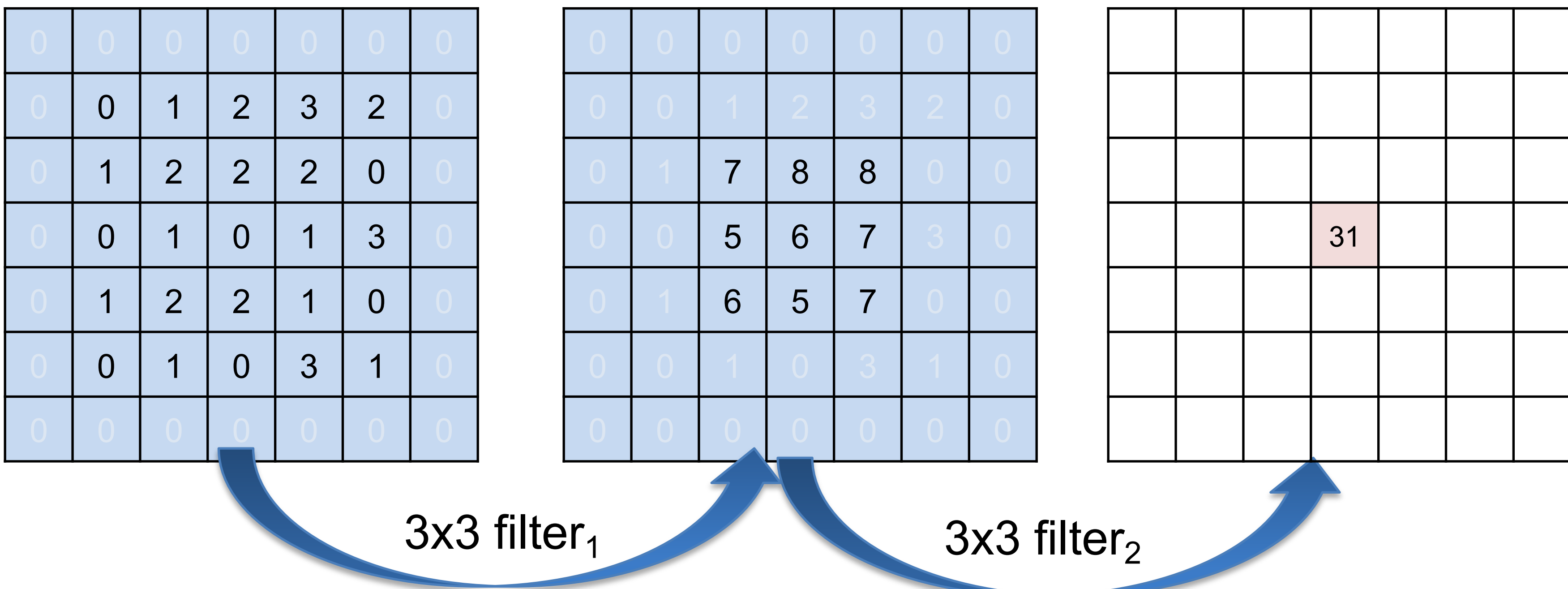
<u>Example</u>



VGGNet: Stacked Filters

- Deeper network means more weights
- Use stack of smaller filters (3x3) to cover the same receptive field with fewer filter weights
- Non-linear activation inserted between each filter

<u>Example</u>: 5x5 filter (25 weights) → two 3x3 filters (18 weights)

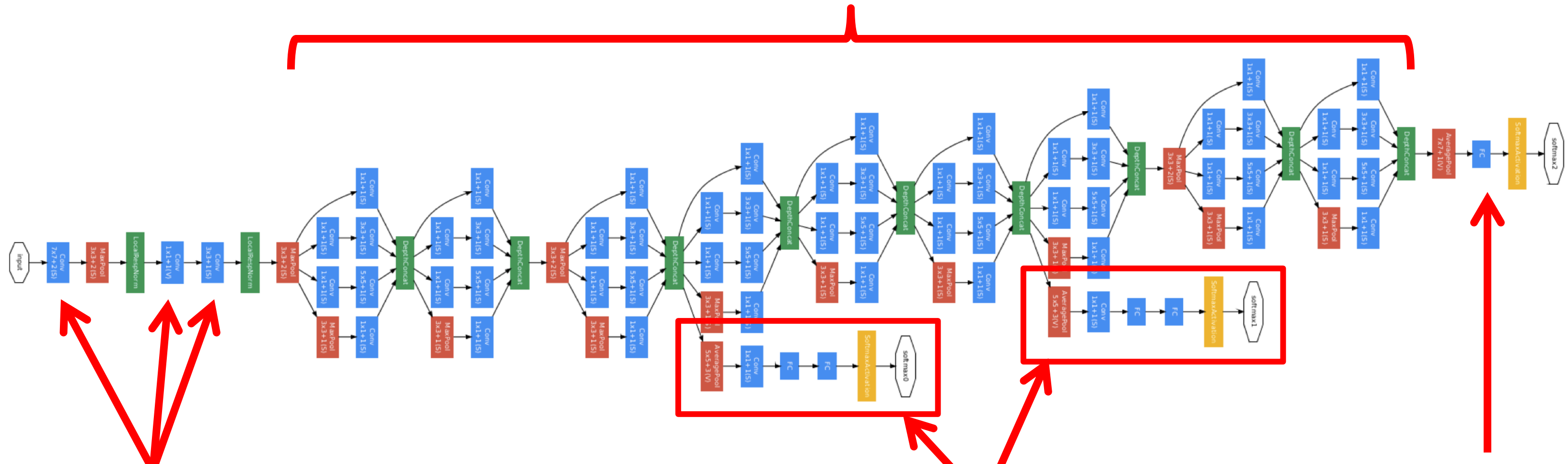


GoogLeNet/Inception (v1)

CONV Layers: 21 (depth), 57 (total)
Fully Connected Layers: 1
Weights: 7.0M
MACs: 1.43G

Also, v2, v3 and v4
ILSVRC14 Winner

9 Inception Layers



3 CONV layers

Auxiliary Classifiers
(helps with training,
not used during inference)

1 FC layer
(reduced from 3)

[Szegedy et al., arXiv 2014, CVPR 2015]

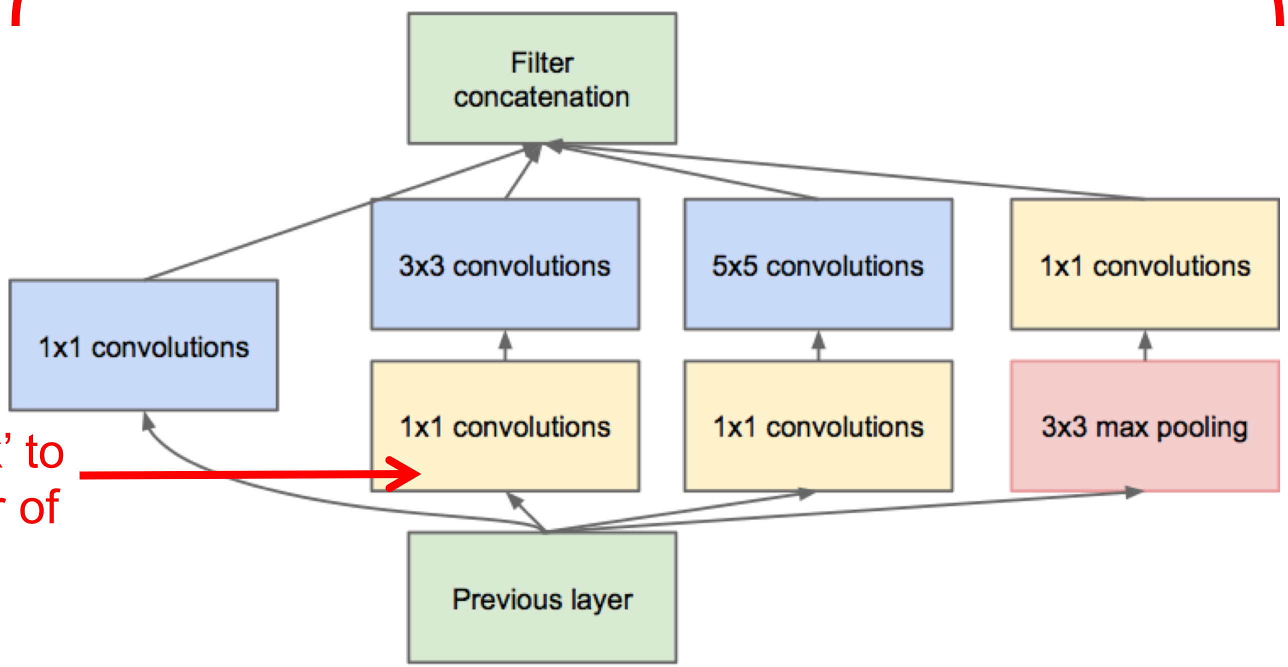
GoogLeNet/Inception (v1)

CONV Layers: 21 (depth), 57 (total)
Fully Connected Layers: 1
Weights: 7.0M
MACs: 1.43G

Also, v2, v3 and v4
ILSVRC14 Winner

parallel filters of different size have the effect of processing image at different scales

Inception Module

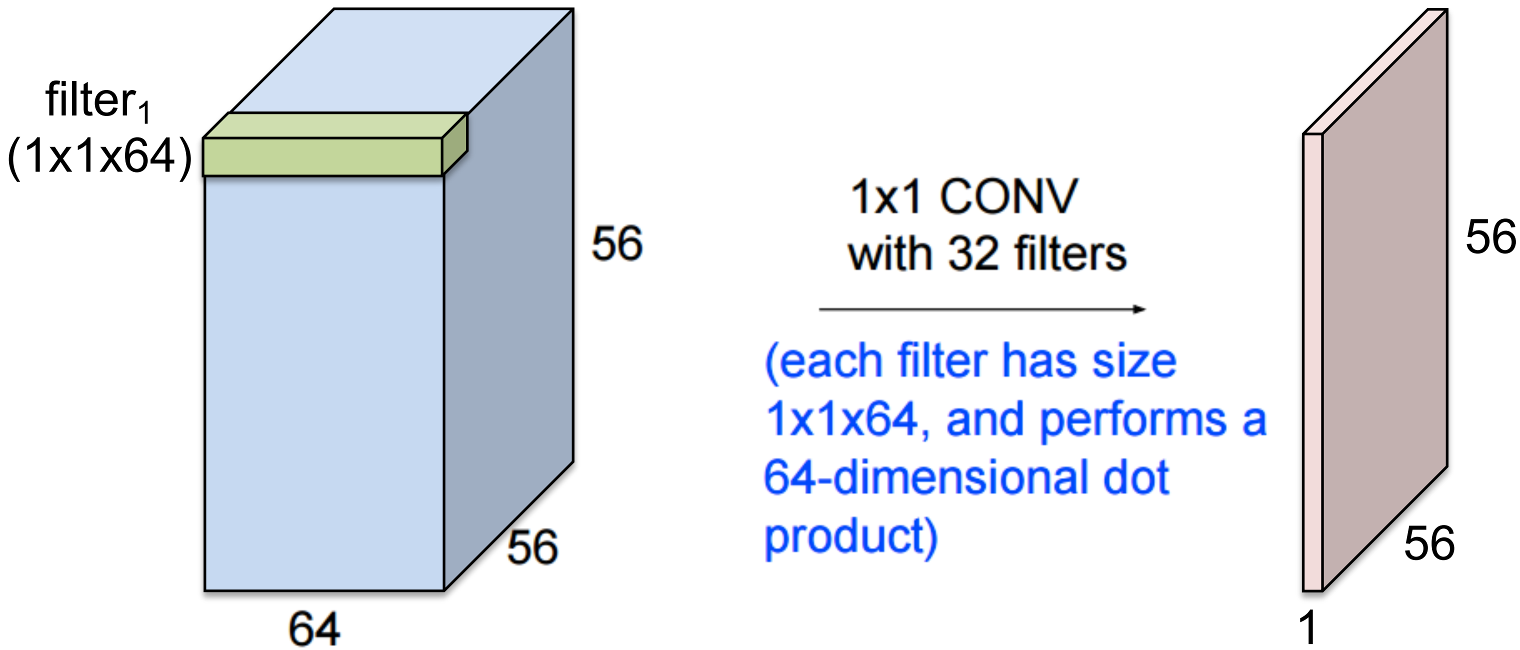


1x1 'bottleneck' to reduce number of weights and multiplications

[Szegedy et al., arXiv 2014, CVPR 2015]

1x1 Bottleneck

Use **1x1 filter** to capture cross-channel correlation, but no spatial correlation.
Can be used to reduce the number of channels in next layer (**bottleneck**)

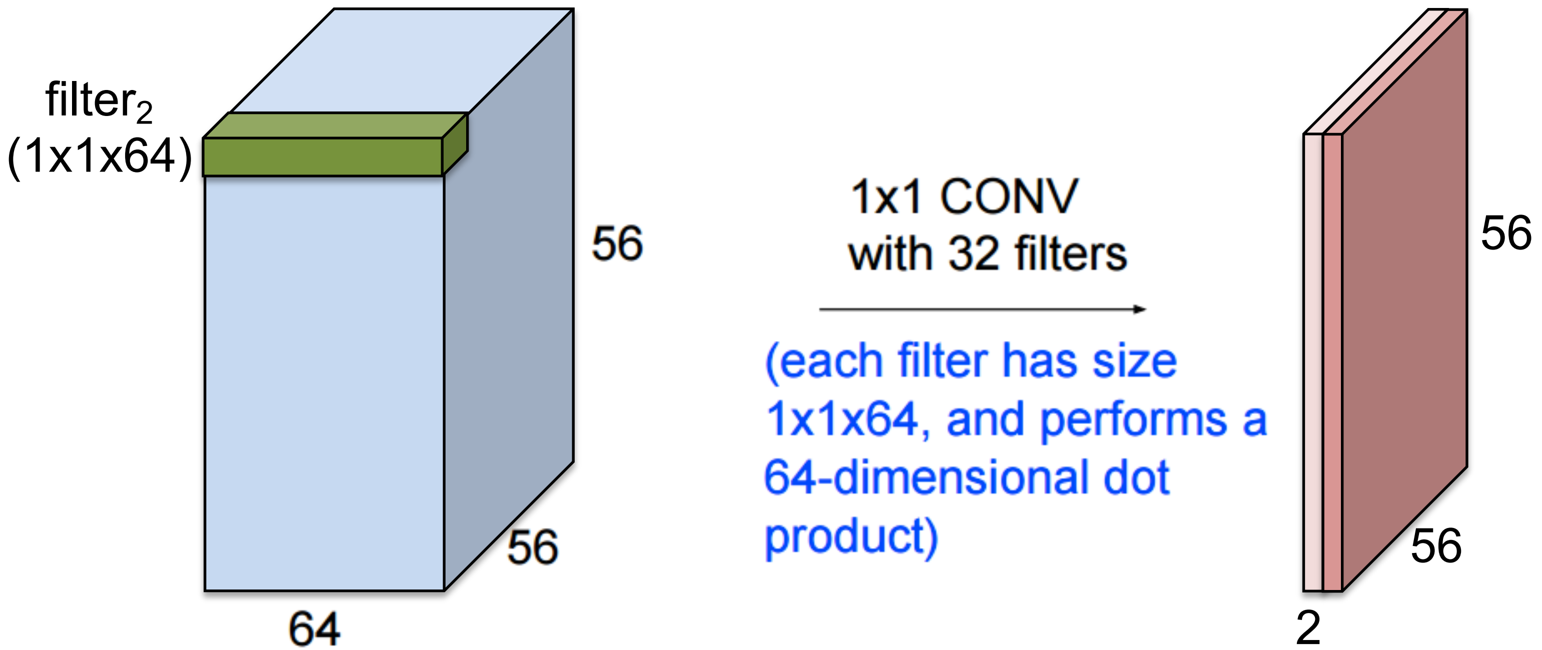


Modified image from source:
Stanford cs231n

[Lin et al., Network in Network, arXiv 2013, ICLR 2014]

1x1 Bottleneck

Use **1x1 filter** to capture cross-channel correlation, but no spatial correlation. Can be used to reduce the number of channels in next layer (**bottleneck**)

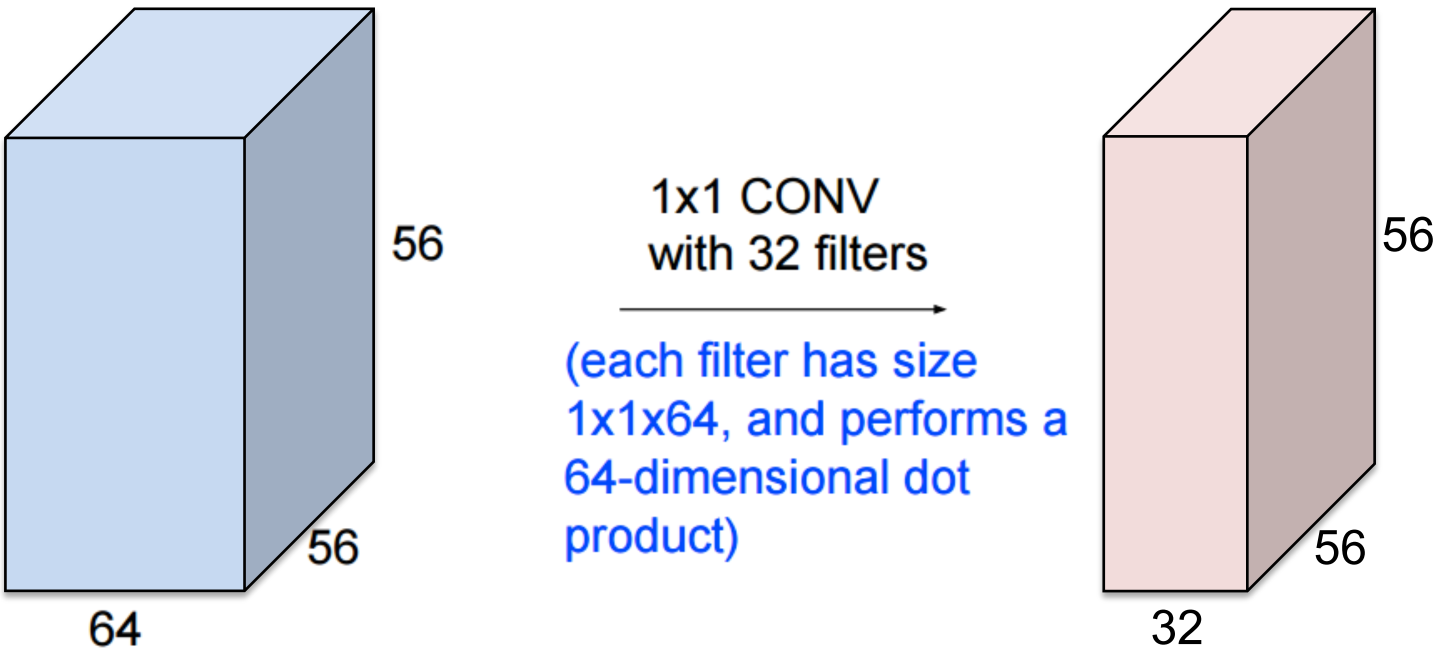


Modified image from source: Stanford cs231n

[Lin et al., Network in Network, arXiv 2013, ICLR 2014]

1x1 Bottleneck

Use **1x1 filter** to capture cross-channel correlation, but no spatial correlation.
Can be used to reduce the number of channels in next layer (**bottleneck**)



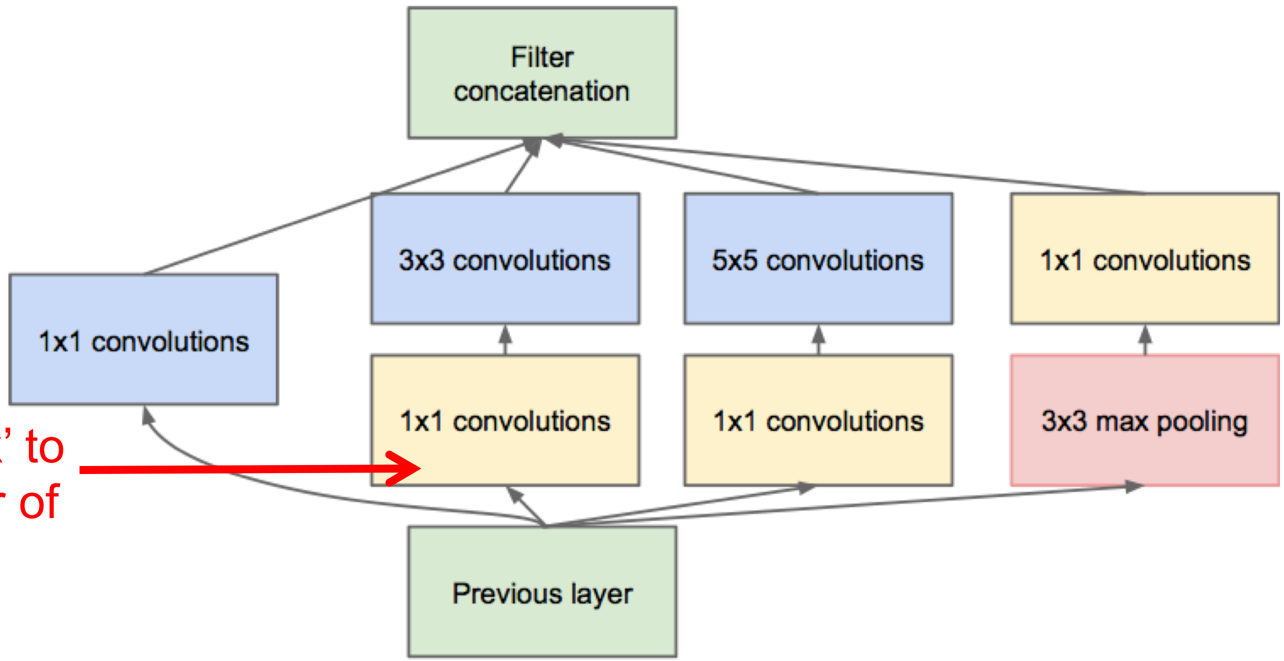
Modified image from source:
Stanford cs231n

[Lin et al., Network in Network, arXiv 2013, ICLR 2014]

GoogLeNet:1x1 Bottleneck

Apply bottleneck before ‘large’ convolution filters.
Reduce weights such that **entire CNN can be trained on one GPU.**
Number of multiplications reduced from 854M → 358M

Inception Module



1x1 ‘bottleneck’ to reduce number of weights and multiplications

[Szegedy et al., arXiv 2014, CVPR 2015]

ResNet

ILSVRC15 Winner
(better than human level accuracy!)

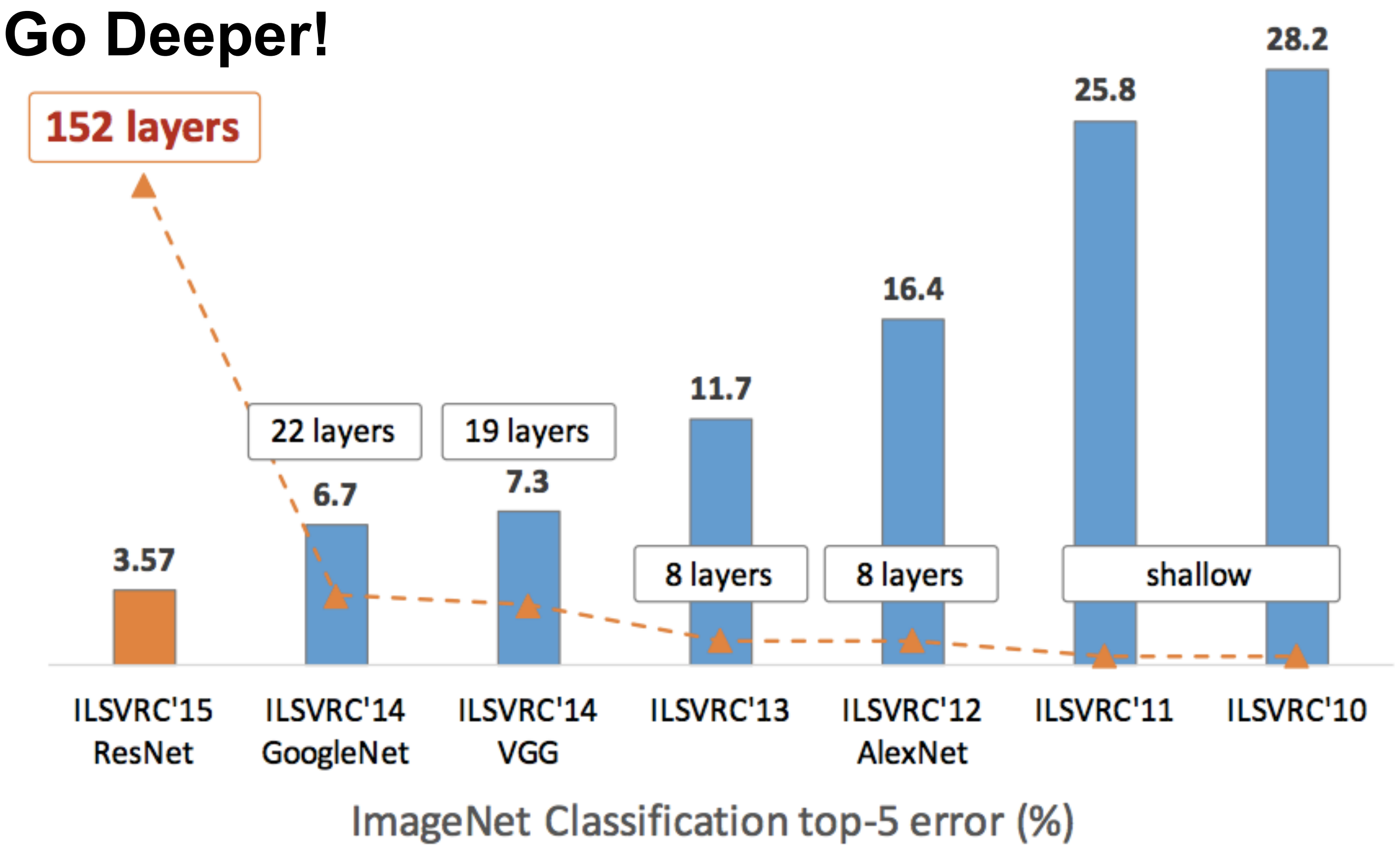
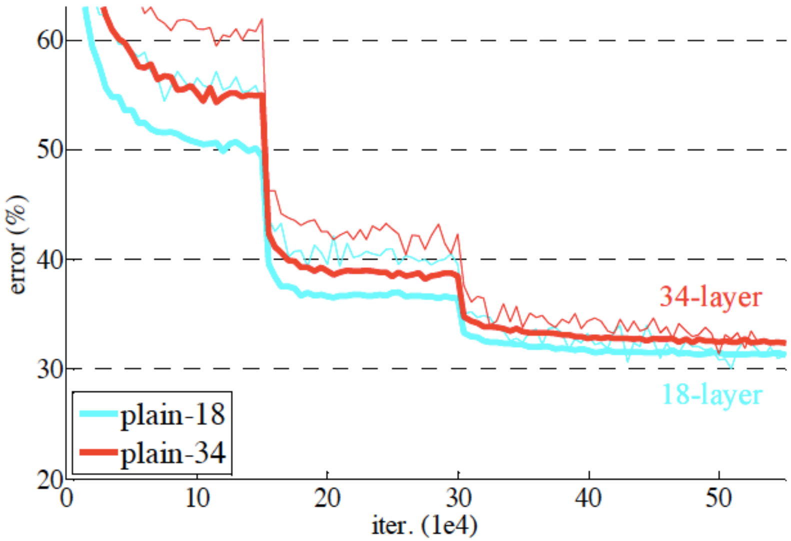


Image Source: http://icml.cc/2016/tutorials/icml2016_tutorial_deep_residual_networks_kaiminghe.pdf

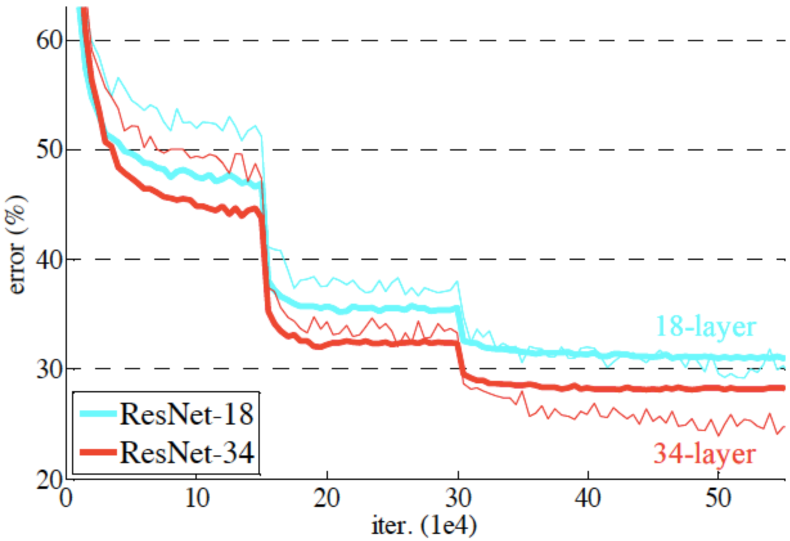
ResNet: Training

Training and validation error **increases** with more layers;
this is due to vanishing gradient, no overfitting.
Introduce **short cut module** to address this!

Without shortcut



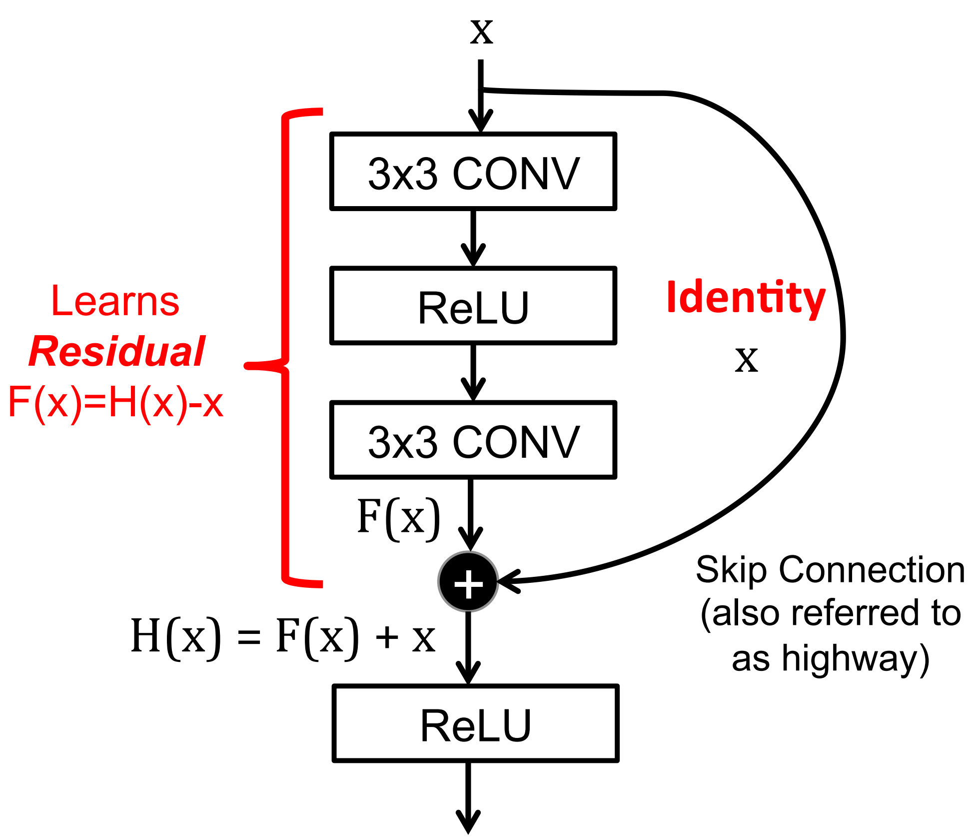
With shortcut

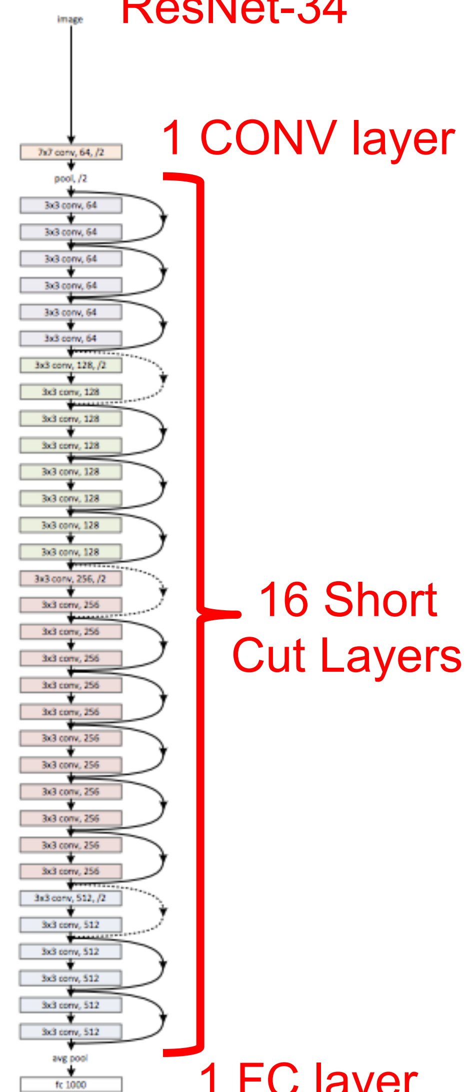


Thin curves denote training error, and bold curves denote validation error.

[He et al., arXiv 2015, CVPR 2016]

ResNet: Short Cut Module



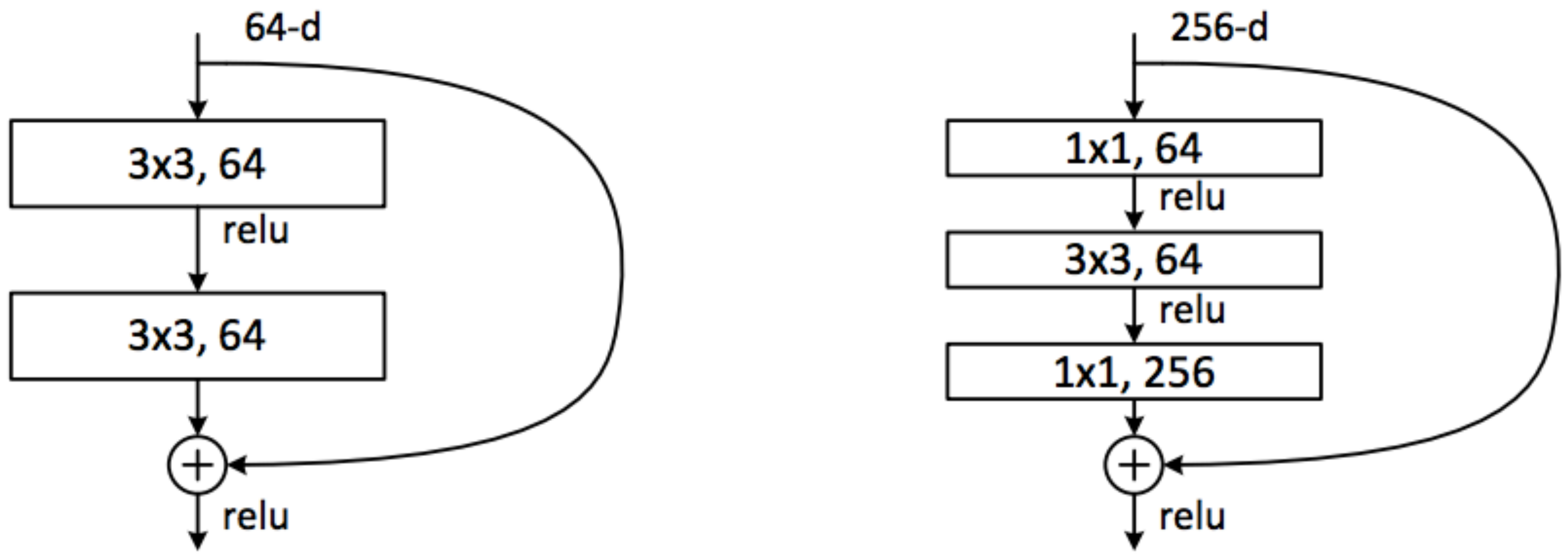


Helps address the vanishing gradient challenge for training very deep networks

[He et al., arXiv 2015, CVPR 2016]

ResNet: Bottleneck

Apply 1x1 bottleneck to reduce computation and size
Also makes network deeper (ResNet-34 → ResNet-50)



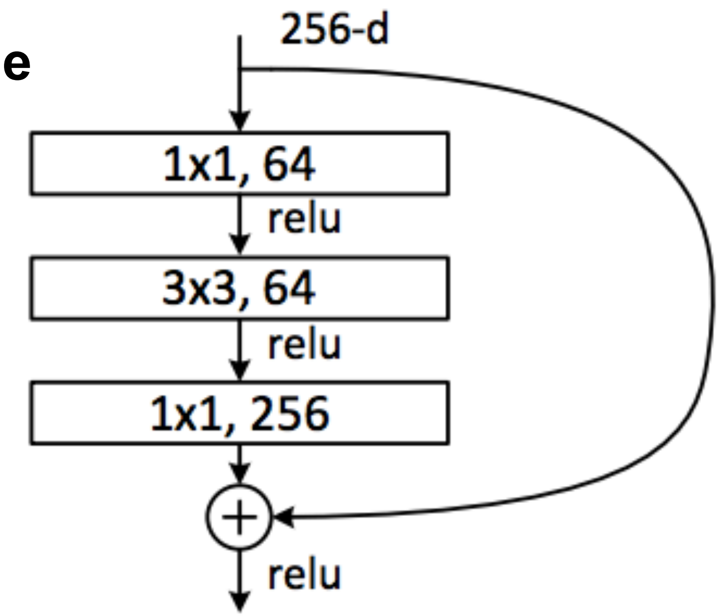
[He et al., arXiv 2015, CVPR 2016]

ResNet-50

CONV Layers: 49
Fully Connected Layers: 1
Weights: 25.5M
MACs: 3.9G

Also, 34,**152** and 1202 layer versions
ILSVRC15 Winner

Short Cut Module



ResNet-34

1 CONV layer

16 Short Cut Layers

1 FC layer

[He et al., arXiv 2015, CVPR 2016]

Summary of Popular DNNs

Metrics	LeNet-5	AlexNet	VGG-16	GoogLeNet (v1)	ResNet-50
Top-5 error	n/a	16.4	7.4	6.7	5.3
Input Size	28x28	227x227	224x224	224x224	224x224
# of CONV Layers	**2**	**5**	**16**	**21 (depth)**	**49**
Filter Sizes	5	3, 5,11	3	1, 3 , 5, 7	1, 3, 7
# of Channels	1, 6	3 - 256	3 - 512	3 - 1024	3 - 2048
# of Filters	6, 16	96 - 384	64 - 512	64 - 384	64 - 2048
Stride	1	1, 4	1	1, 2	1, 2
# of Weights	2.6k	2.3M	14.7M	6.0M	23.5M
# of MACs	283k	666M	15.3G	1.43G	3.86G
# of FC layers	**2**	**3**	**3**	**1**	**1**
# of Weights	58k	58.6M	124M	1M	2M
# of MACs	58k	58.6M	124M	1M	2M
Total Weights	**60k**	**61M**	**138M**	**7M**	**25.5M**
Total MACs	**341k**	**724M**	**15.5G**	**1.43G**	**3.9G**

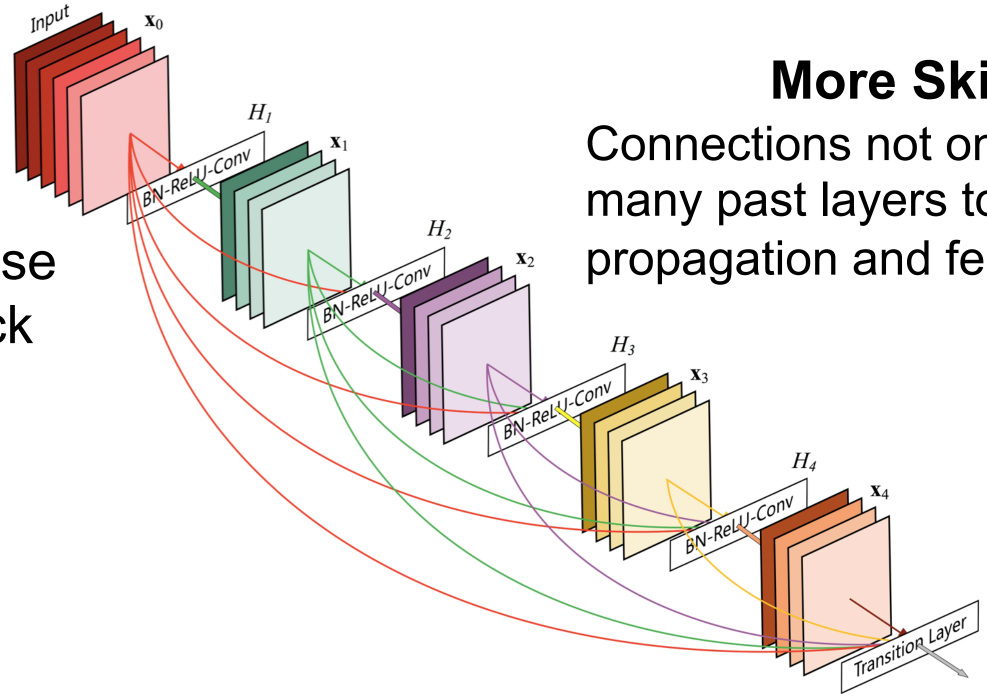
CONV Layers increasingly important!

Summary of Popular DNNs

- **AlexNet**
  - First CNN Winner of ILSVRC
  - Uses LRN (deprecated after this)
- **VGG-16**
  - Goes Deeper (16+ layers)
  - Uses only 3x3 filters (stack for larger filters)
- **GoogLeNet (v1)**
  - Reduces weights with Inception and only one FC layer
  - Inception: 1x1 and DAG (parallel connections)
  - Batch Normalization
- **ResNet**
  - Goes Deeper (24+ layers)
  - Shortcut connections

DenseNet

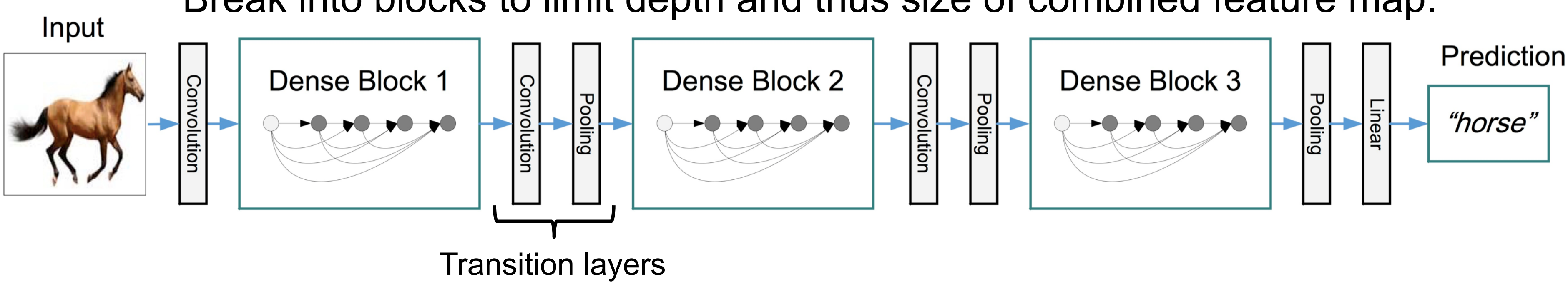
Dense Block



More Skip Connections!

Connections not only from previous layer, but many past layers to strengthen feature map propagation and feature reuse.

Feature maps are concatenated rather than added.
Break into blocks to limit depth and thus size of combined feature map.

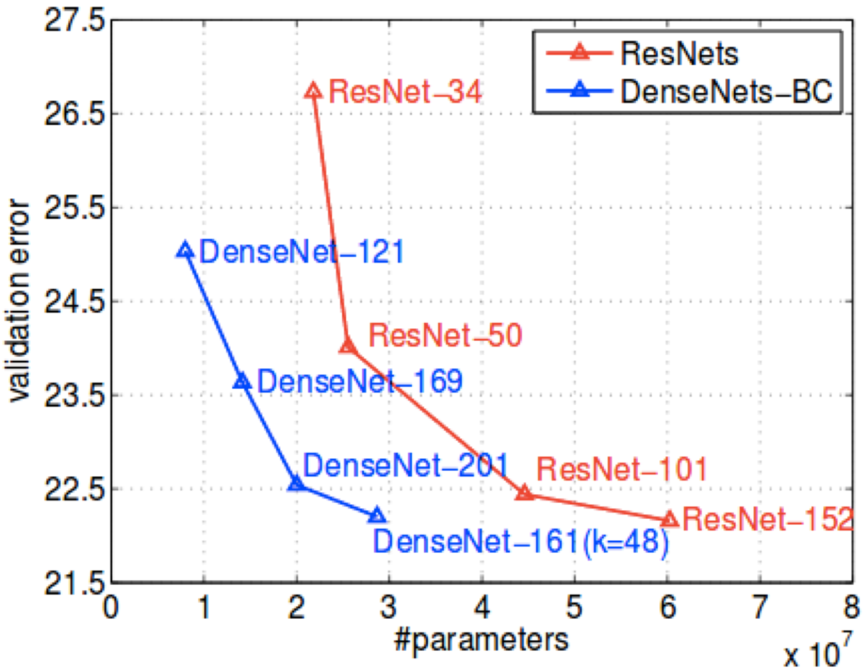


[Huang et al., CVPR 2017]

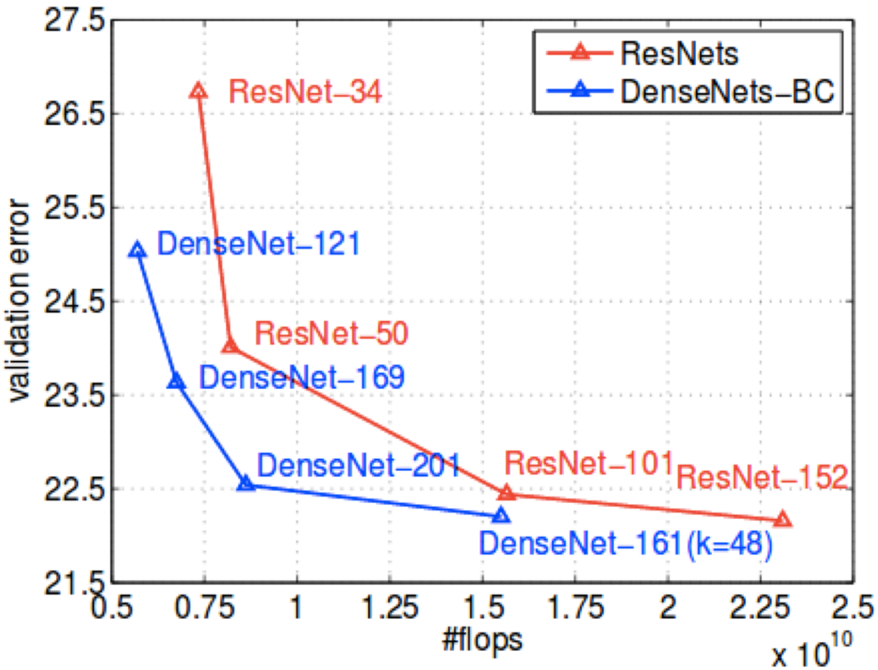
DenseNet

Higher accuracy than ResNet with fewer weights and multiplications

Top-1 error



Top-1 error



Note: 1 MAC = 2 FLOPS

[Huang et al., CVPR 2017]

Wide ResNet

Increase width (# of filters) rather than depth of network

- 50-layer wide ResNet outperforms 152-layer original ResNet
- Increasing width instead of depth is also more parallel-friendly

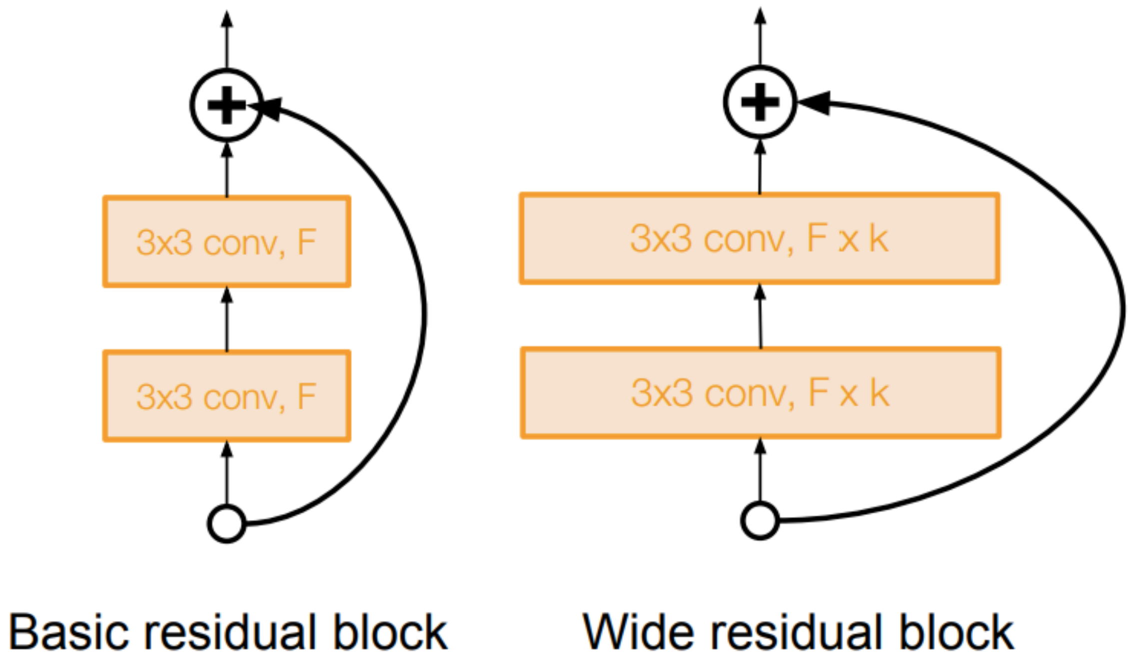
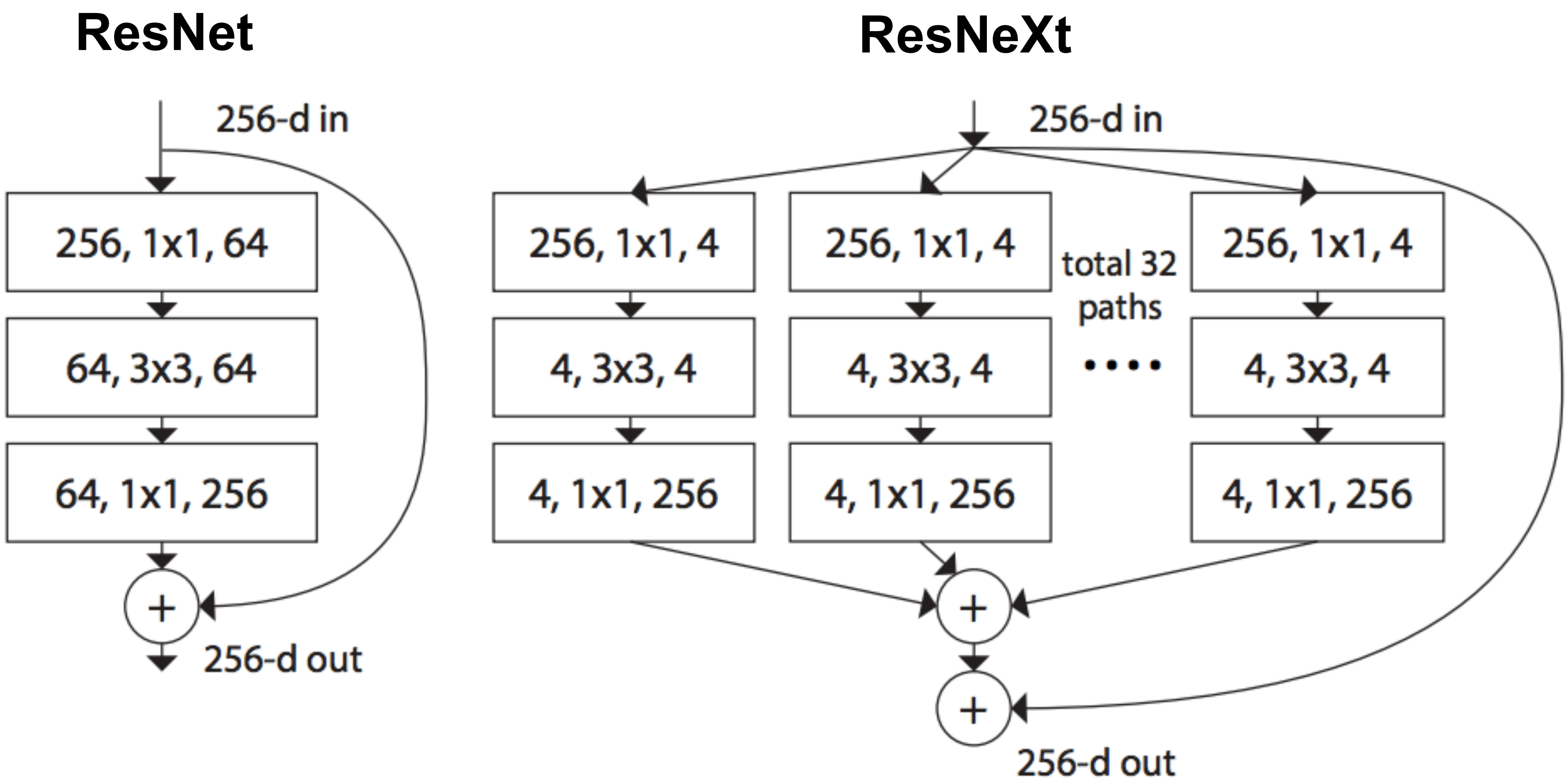


Image Source: Stanford cs231n

[Zagoruyko et al., BMVC 2016]

ResNeXt

Increase number of **convolution groups** (referred to as *cardinality*) instead of depth and width of network

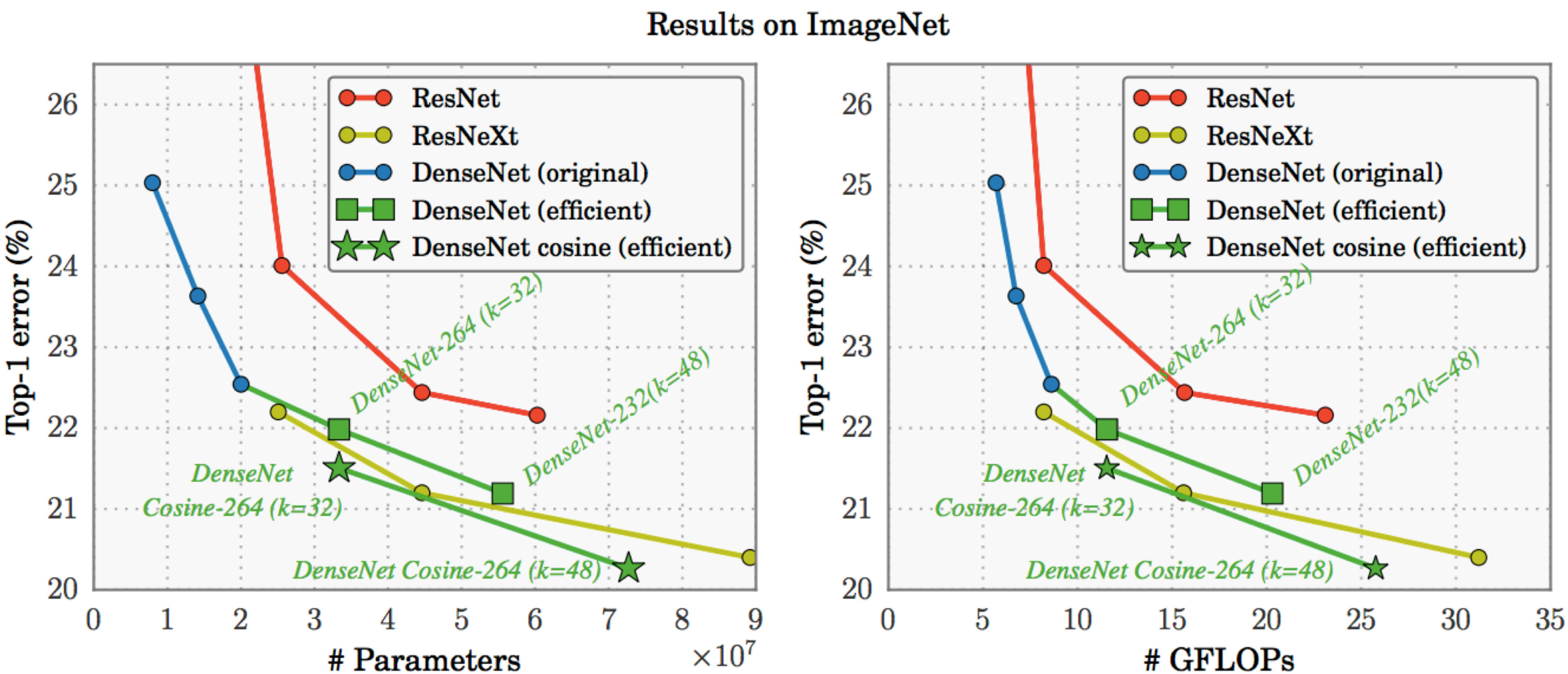


Used by ILSVRC 2017 Winner WMW

[Xie et al., CVPR 2017]

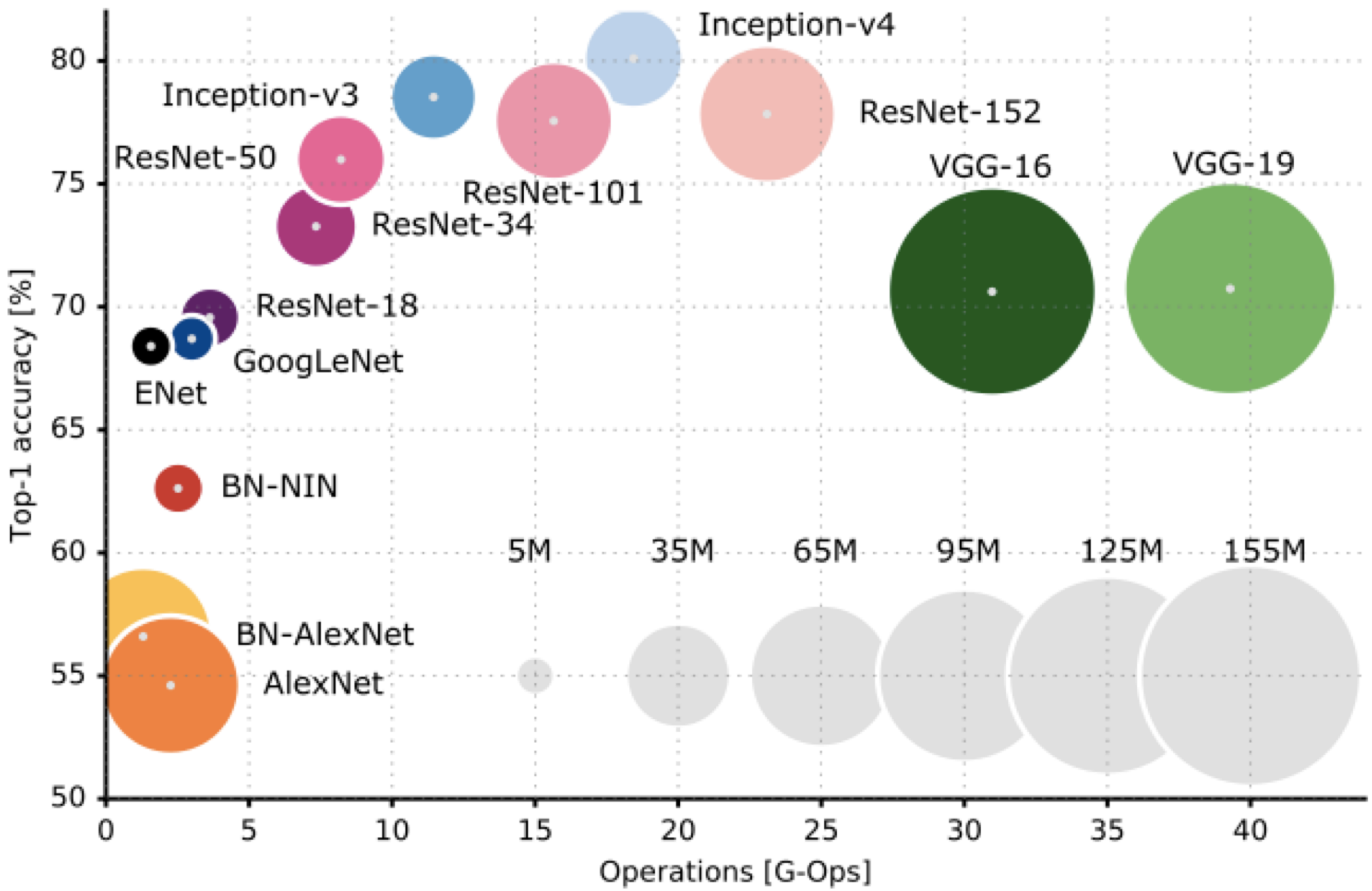
ResNeXt

Improved accuracy vs. 'complexity' tradeoff compared to other ResNet based models



Efficient DNN Models

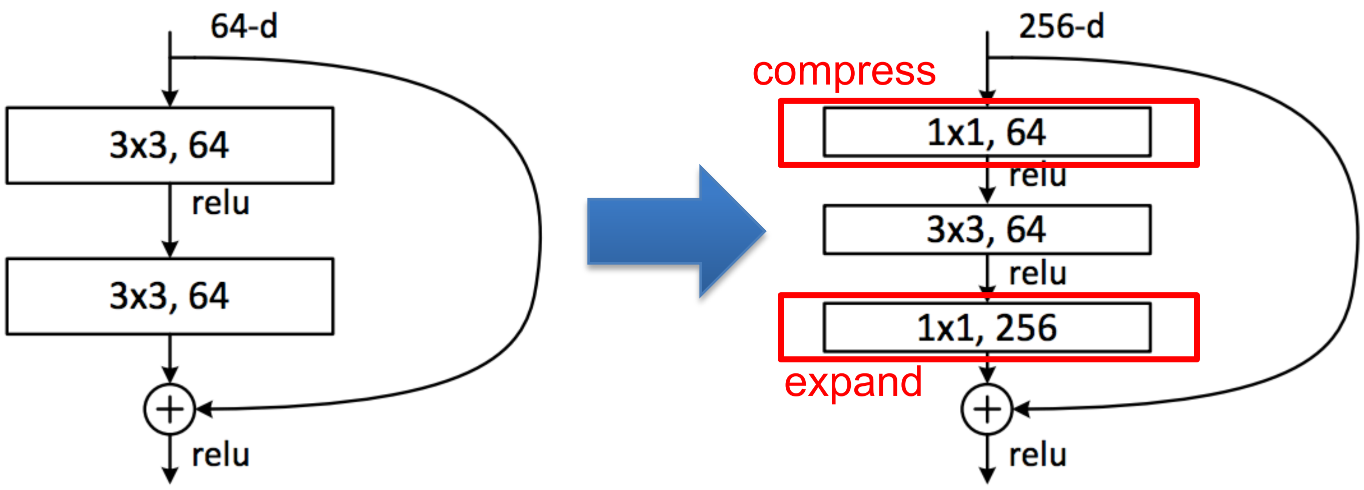
Accuracy vs. Weight & OPs



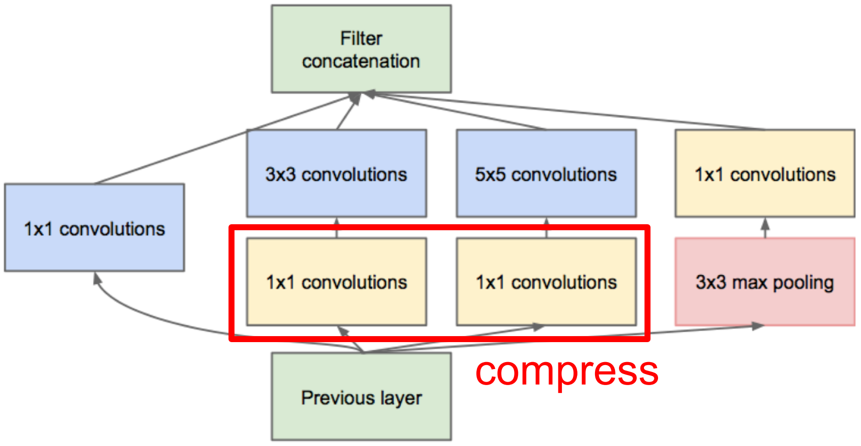
[Alfredo et al., arXiv, 2017]

Bottleneck in Popular DNN Models

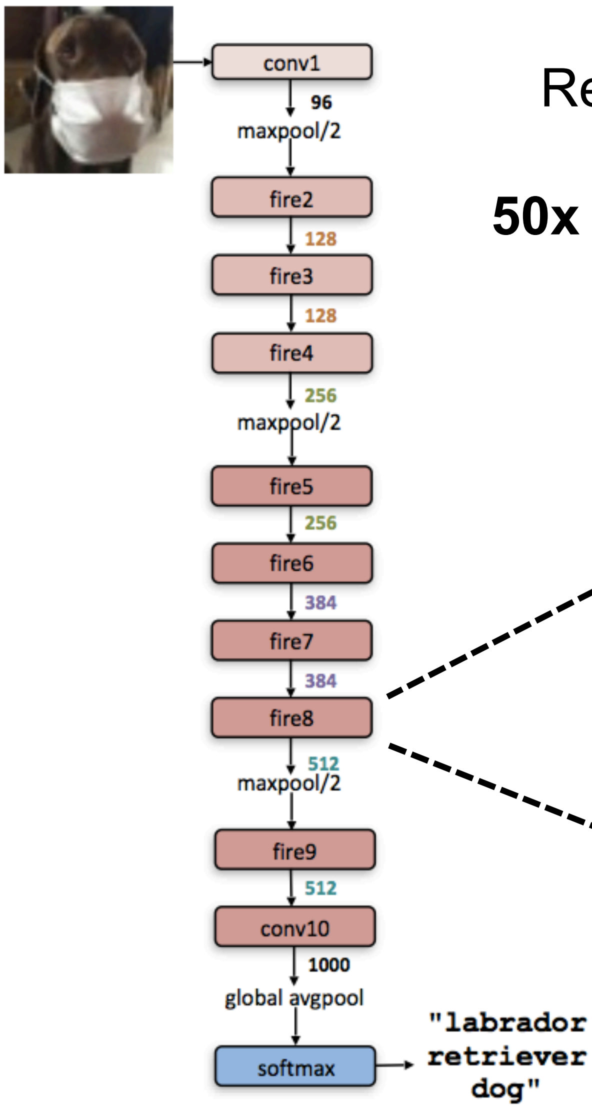
ResNet



GoogleNet



Example: SqueezeNet

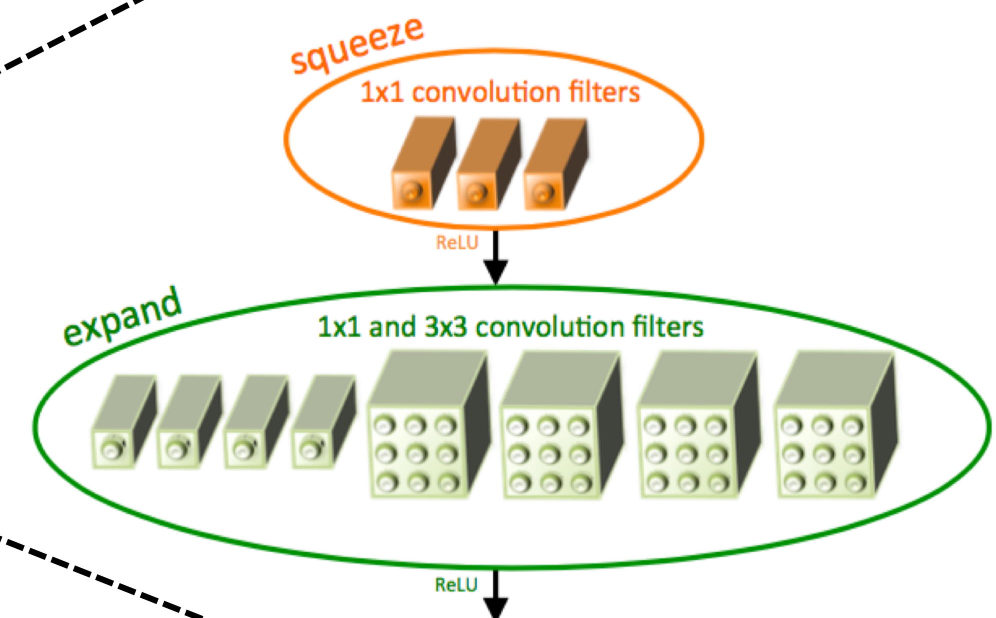


Reduce number of weights by reducing number of input channels by “squeezing” with 1x1

50x fewer weights than AlexNet (no accuracy loss)

However, 2.4x more operations than AlexNet*

Fire Module



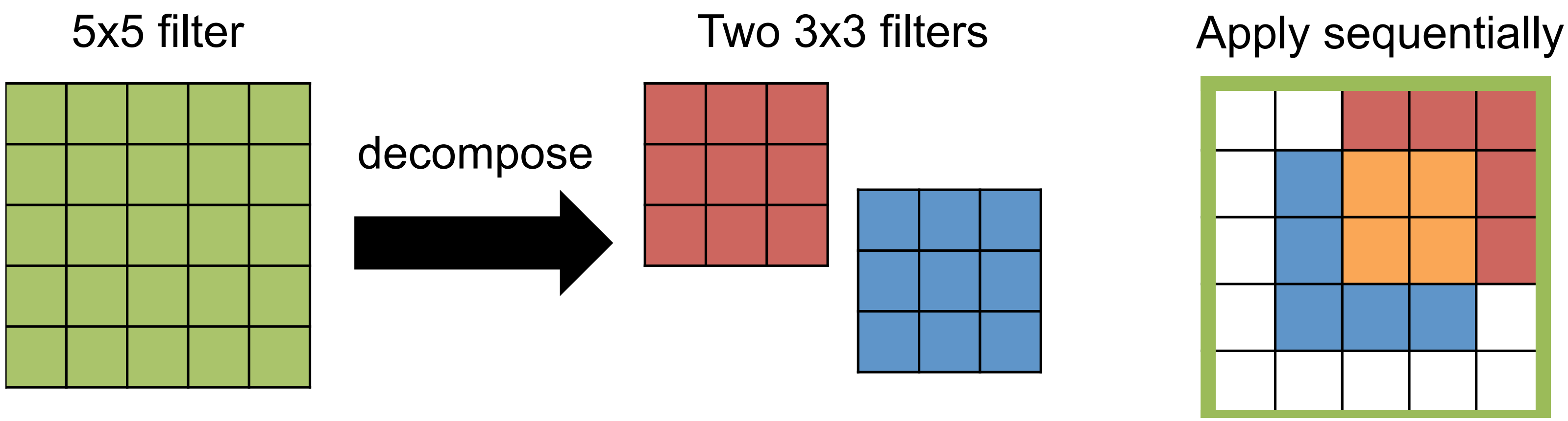
[Iandola et al., arXiv 2016, ICLR 2017]

*SqueezeNetv1.0

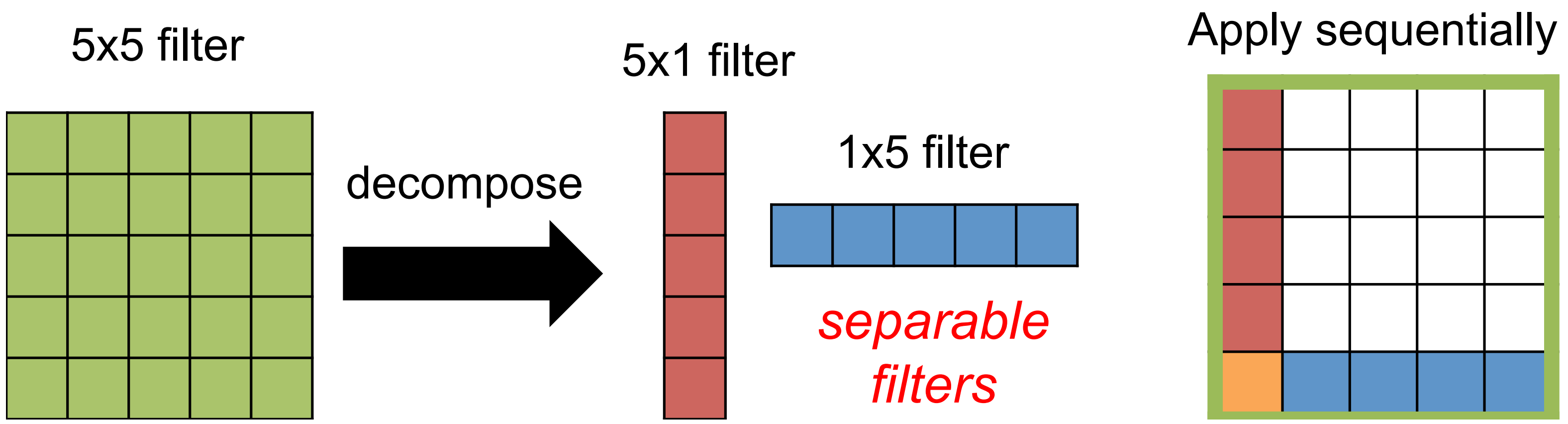
Stacking Small Filters

Build network with a **series of small filters**
(reduces degrees of freedom)

VGG-16



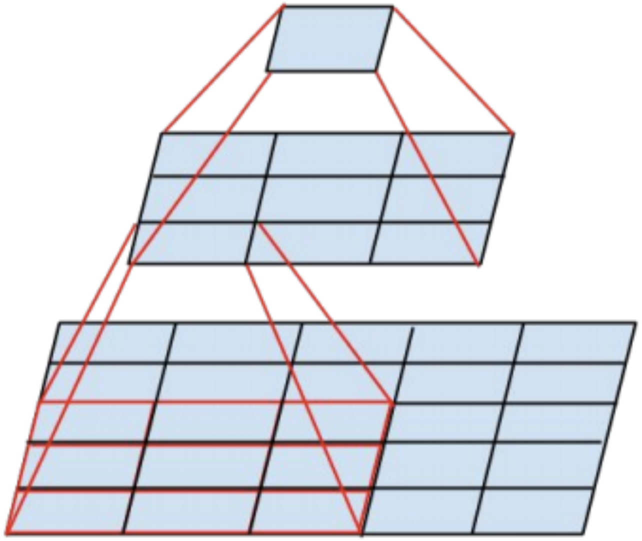
GoogleNet/Inception v3



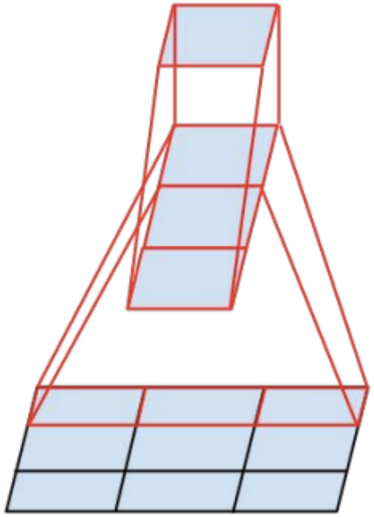
Example: Inception V3

Go deeper **(v1: 22 layers → v3: 40+ layers)** by reducing the number of weights per filter using **filter decomposition**
~3.5% higher accuracy than v1

5x5 filter → 3x3 filters



3x3 filter → 3x1 and 1x3 filters

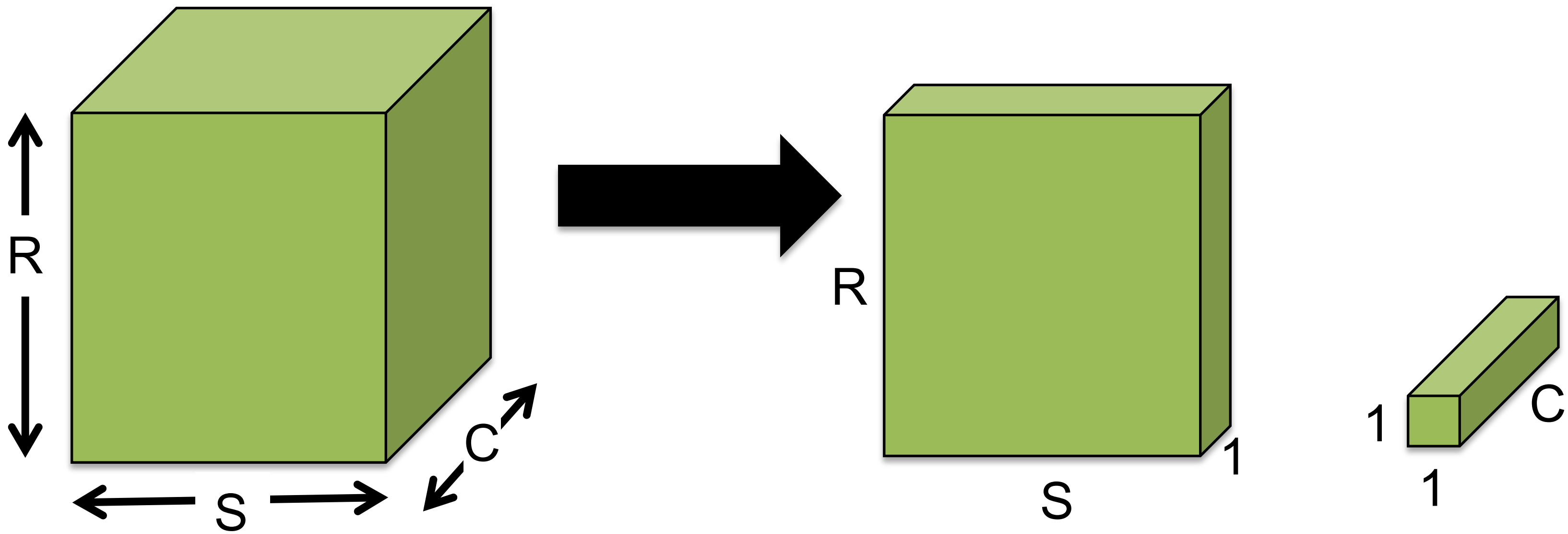


Separable filters

[Szegedy et al., arXiv 2015]

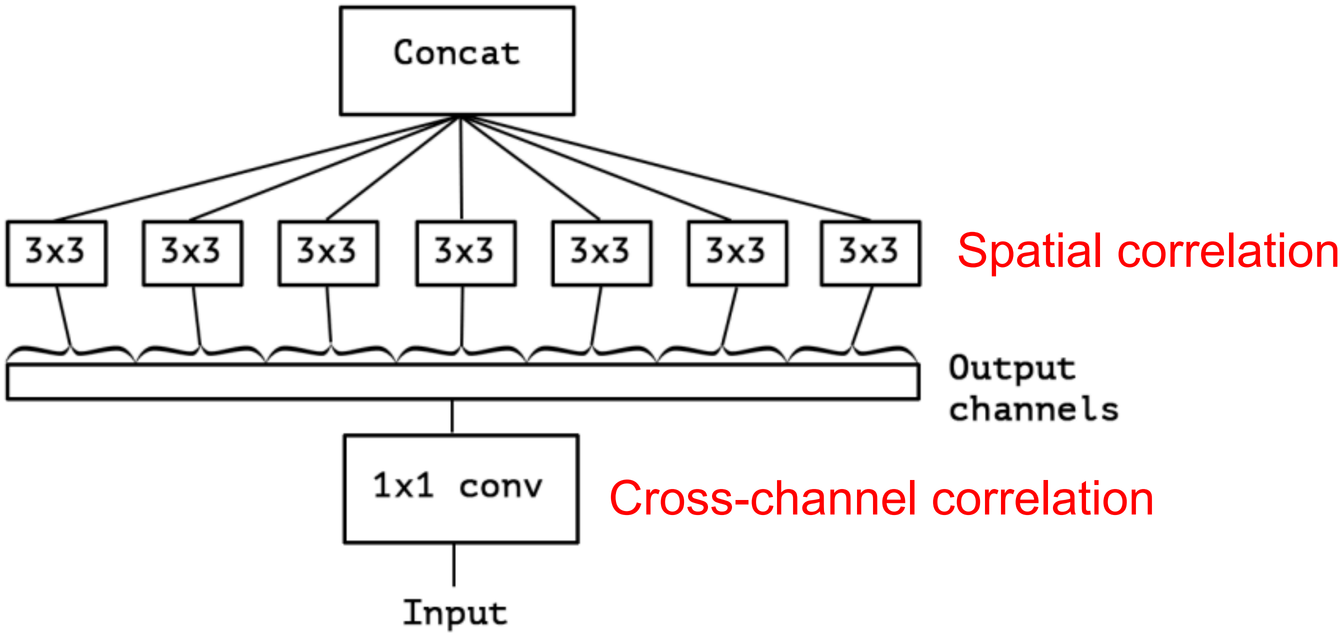
Depth-wise Separable

Decouple the **cross-channels correlations** and **spatial correlations** in the feature maps of the DNN



Example: Xception

- An Inception module based on depth-wise separable convolutions
- Claims to learn richer features with similar number of weights as Inception V3 (i.e. more efficient use of weights)
  - Similar performance on ImageNet; 4.3% better on larger dataset (JFT)
  - However, 1.5x more operations required than Inception V3



[Chollet, CVPR 2017]

Example: MobileNets

Depth-wise filter decomposition

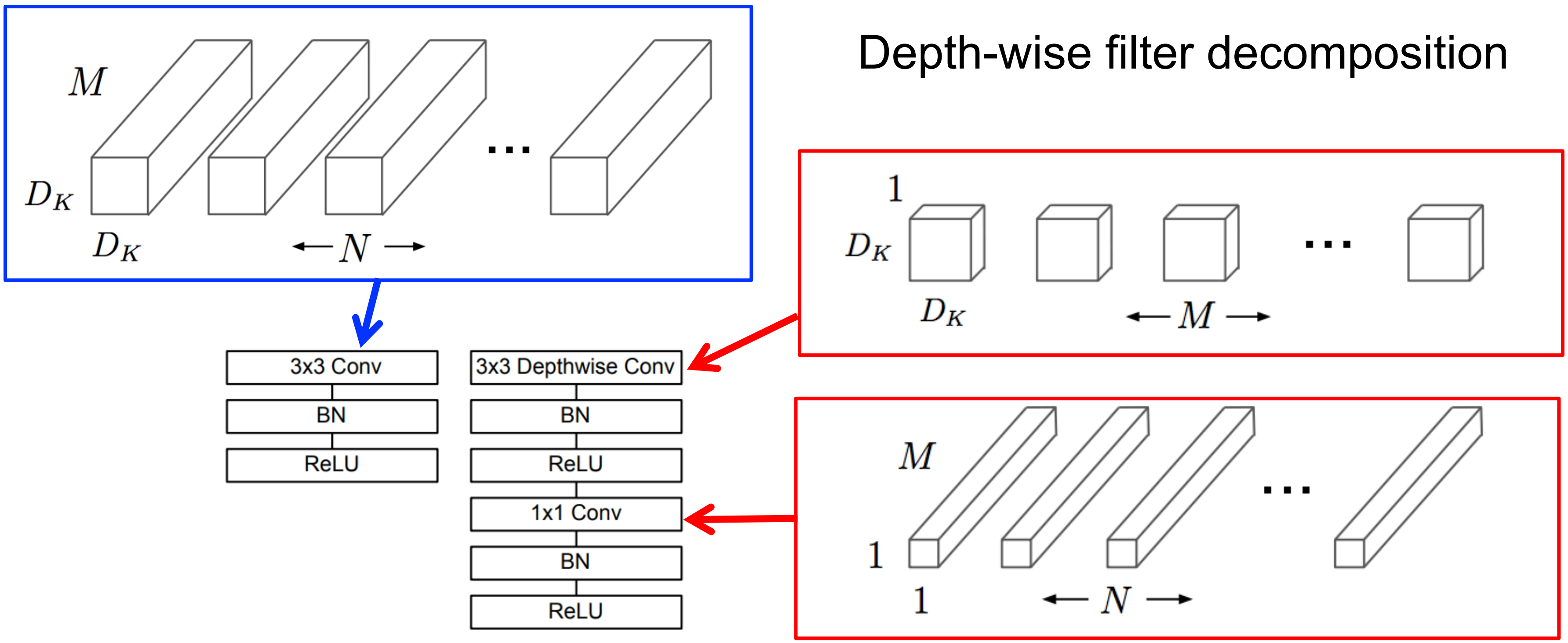


Table 4. Depthwise Separable vs Full Convolution MobileNet

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameters
Conv MobileNet	71.7%	4866	29.3
MobileNet	70.6%	569	4.2

[Howard et al., arXiv, April 2017]

MobileNets: Comparison

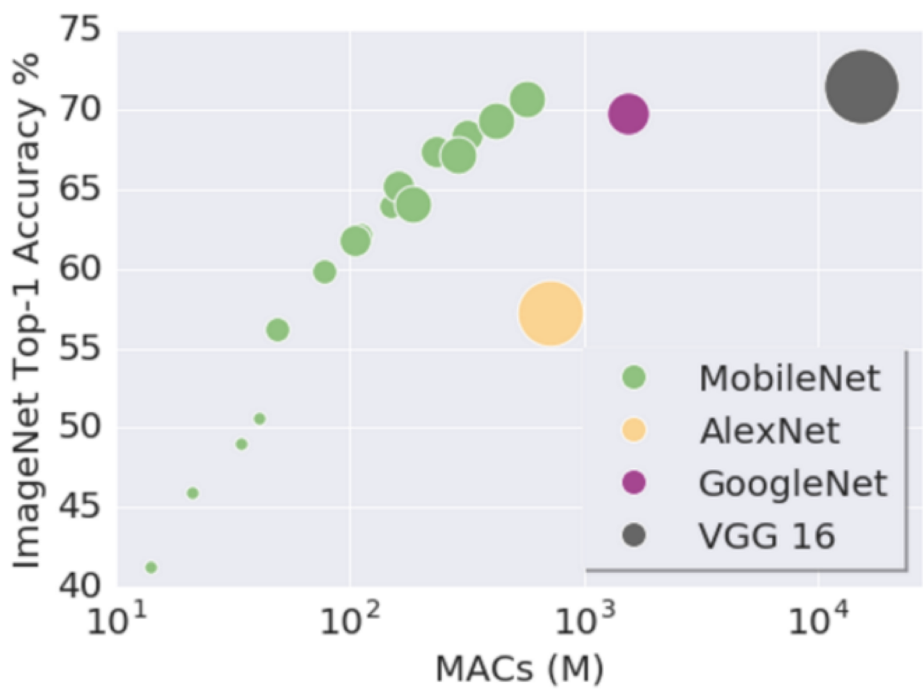
Comparison with other DNN Models

Table 8. MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameter
1.0 MobileNet-224	70.6%	569	4.2
GoogleNet	69.8%	1550	6.8
VGG 16	71.5%	15300	138

Table 9. Smaller MobileNet Comparison to Popular Models

Model	ImageNet Accuracy	Million Mult-Adds	Million Parameter
0.50 MobileNet-160	60.2%	76	1.32
Squeezenet	57.5%	1700	1.25
AlexNet	57.2%	720	60

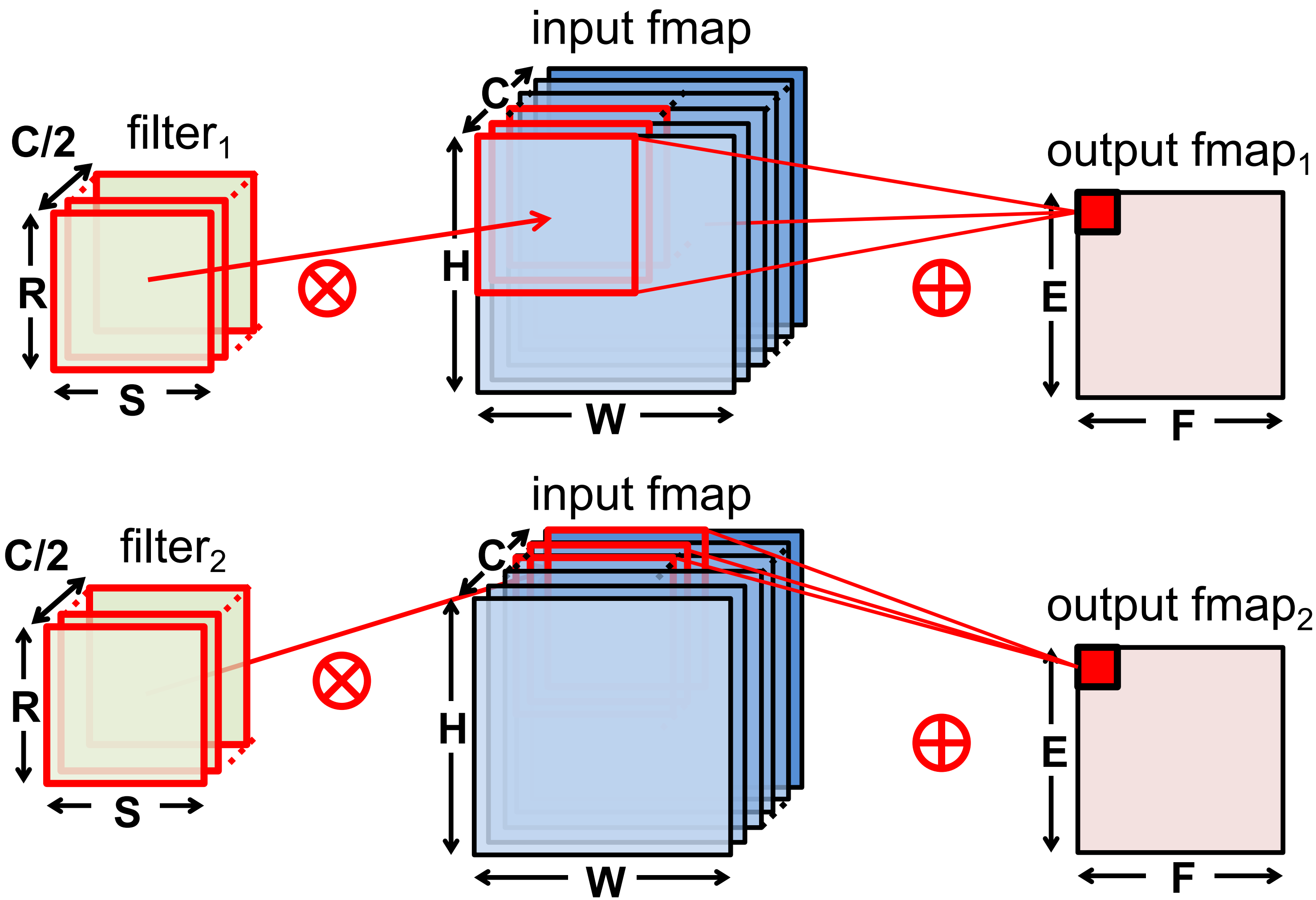


[Image source: Github]

[Howard et al., arXiv, April 2017]

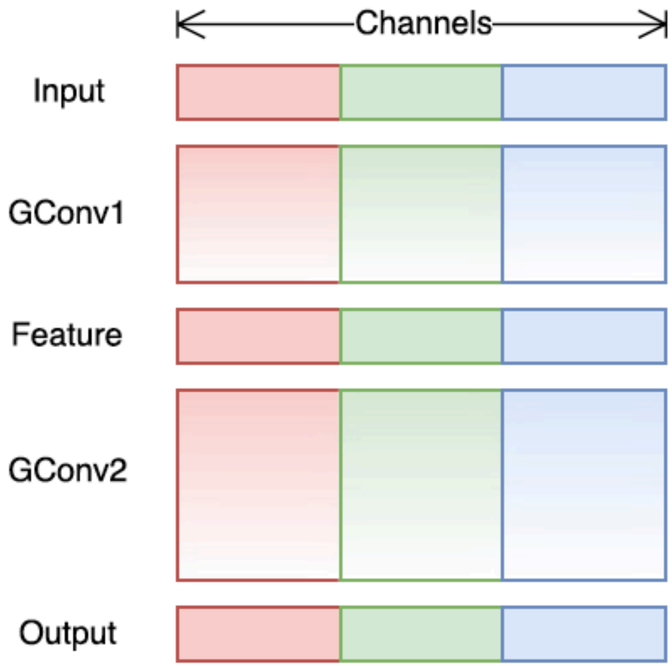
Grouped Convolutions

Grouped convolutions reduce the number of **weights and multiplications** at the cost of not sharing information between **groups**

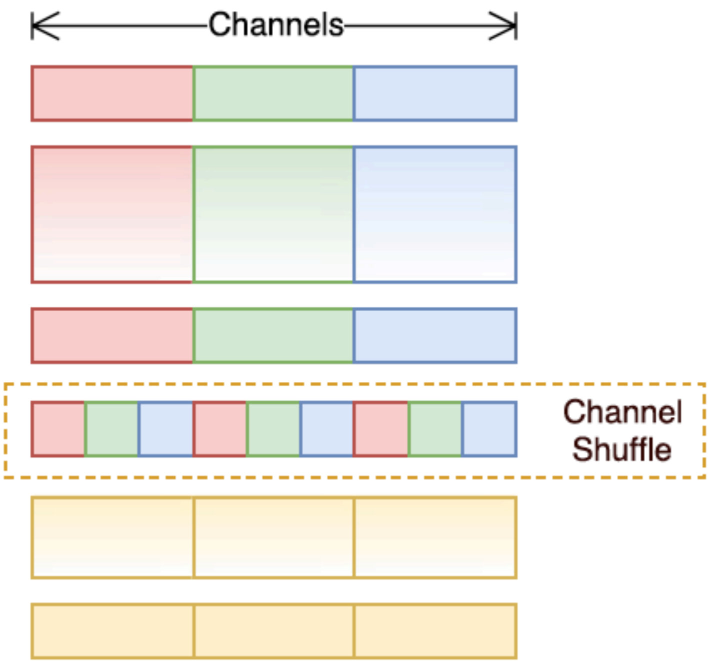


Example: ShuffleNet

Shuffle order such that channels are not isolated across groups
(up to 4% increase in accuracy)



No interaction between
channels from different groups



Shuffling allow interaction between
channels from different groups

[Zhang et al., arXiv, July 2017]

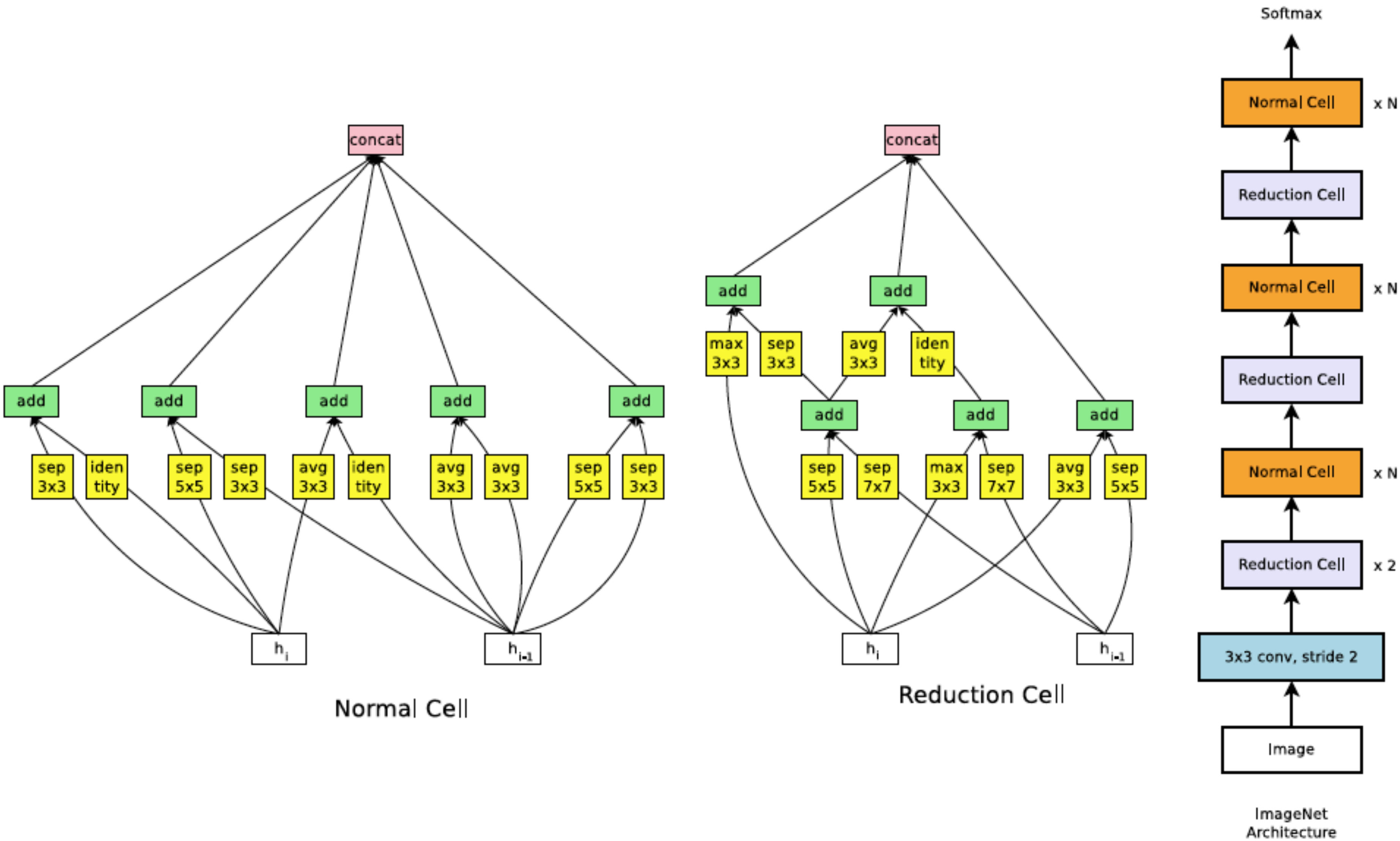
Learn DNN Models

- Rather than handcrafting the model, learn the model
- More recent result uses Neural Architecture Search
- Build model from popular layers

  - Identity
  - 1x3 then 3x1 convolution
  - 1x7 then 7x1 convolution
  - 3x3 dilated convolution
  - 1x1 convolution
  - 3x3 convolution
  - 3x3 separable convolution
  - 5x5 separable convolution
  - 3x3 average pooling
  - 3x3 max pooling
  - 5x5 max pooling
  - 7x7 max pooling

[Zoph et al., arXiv, July 2017]

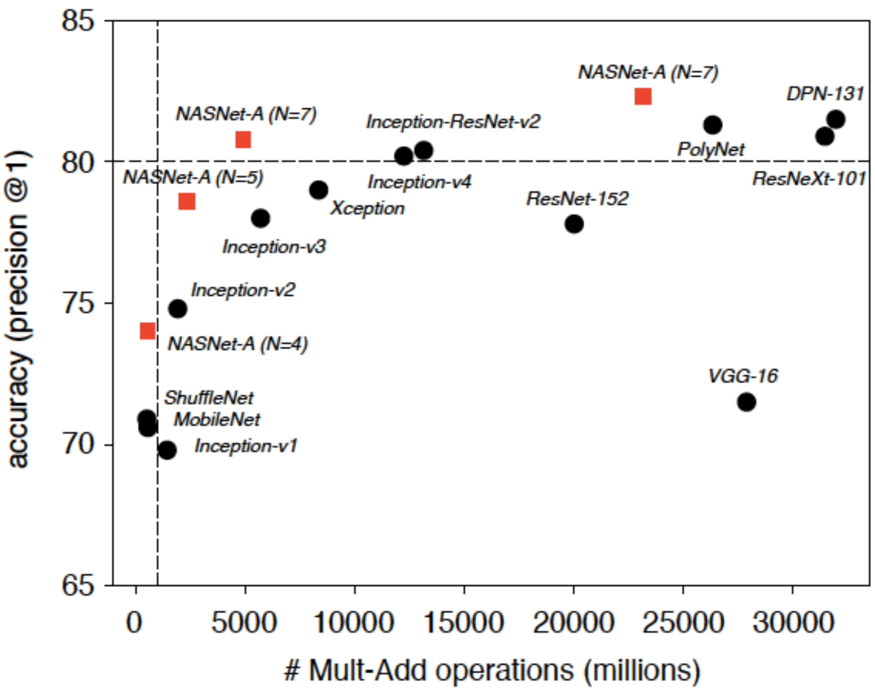
Learned Convolutional Cells

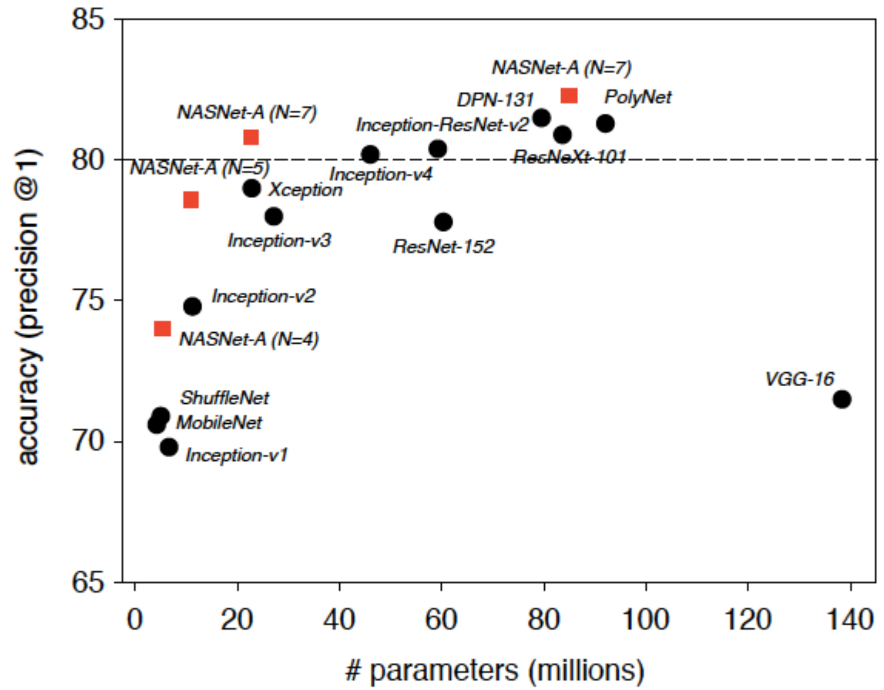


[Zoph et al., arXiv, July 2017]

Comparison with Existing Networks

Learned models have improved accuracy vs. 'complexity' tradeoff compared to handcrafted models





[Zoph et al., arXiv, July 2017]

Comparison with Existing Networks

Model	image size	# parameters	Mult-Adds	Top 1 Acc. (%)	Top 5 Acc. (%)
Inception V2 [27]	224×224	11.2 M	1.94 B	74.8	92.2
NASNet-A (N = 5)	**299×299**	**10.9 M**	**2.35 B**	**78.6**	**94.2**
Inception V3 [51]	299×299	23.8 M	5.72 B	78.0	93.9
Xception [9]	299×299	22.8 M	8.38 B	79.0	94.5
Inception ResNet V2 [50]	299×299	55.8 M	13.2 B	80.4	95.3
NASNet-A (N = 7)	**299×299**	**22.6 M**	**4.93 B**	**80.8**	**95.3**
ResNeXt-101 (64 x 4d) [58]	320×320	83.6 M	31.5 B	80.9	95.6
PolyNet [60]	331×331	92 M	34.7 B	81.3	95.8
DPN-131 [8]	320×320	79.5 M	32.0 B	81.5	95.8
NASNet-A (N = 7)	**331×331**	**84.9 M**	**23.2 B**	**82.3**	**96.0**

Model	# parameters	Mult-Adds	Top 1 Acc. (%)	Top 5 Acc. (%)
Inception V1 [49]	6.6M	1,448 M	69.8	89.9
MobileNet-224 [22]	4.2 M	569 M	70.6	89.5
ShuffleNet (2x) [59]	$\sim$ 5M	524 M	70.9	89.8
NASNet-A (N=4)	**5.3 M**	**564 M**	**74.0**	**91.6**
NASNet-B (N=4)	5.3M	488 M	72.8	91.3
NASNet-C (N=3)	4.9M	558 M	72.5	91.0

[Zoph et al., arXiv, July 2017]

Warning!

- These works use **number of weights and operations** to measure "**complexity**"
- Number of weights provides an indication of **storage cost** for inference
- However later in the course, we will see that
  - Number of operations doesn't directly translate to throughput
  - Number of weights and operations doesn't directly translate to power/energy consumption
- Understanding the underlying hardware is important for evaluating the impact of these "efficient" DNN models

Summary

- Approaches used to improve accuracy by popular DNN models in the ImageNet Challenge
  - Go deeper (i.e. more layers)
  - Stack smaller filters and apply 1x1 bottlenecks to reduce number of weights such that the deeper models can fit into a GPU (faster training)
  - Use multiple connections across layers (e.g. parallel and short cut)
- Efficient models aim to reduce number of weights and number of operations
  - Most use some form of filter decomposition (spatial, depth and channel)
  - Note: Number of weights and operations does not directly map to storage, speed and power/energy. Depends on hardware!
- Filter shapes vary across layers and models
  - Need flexible hardware!

Datasets

Image Classification Datasets

- **Image Classification/Recognition**
  - Given an entire image → Select 1 of N classes
  - No localization (detection)

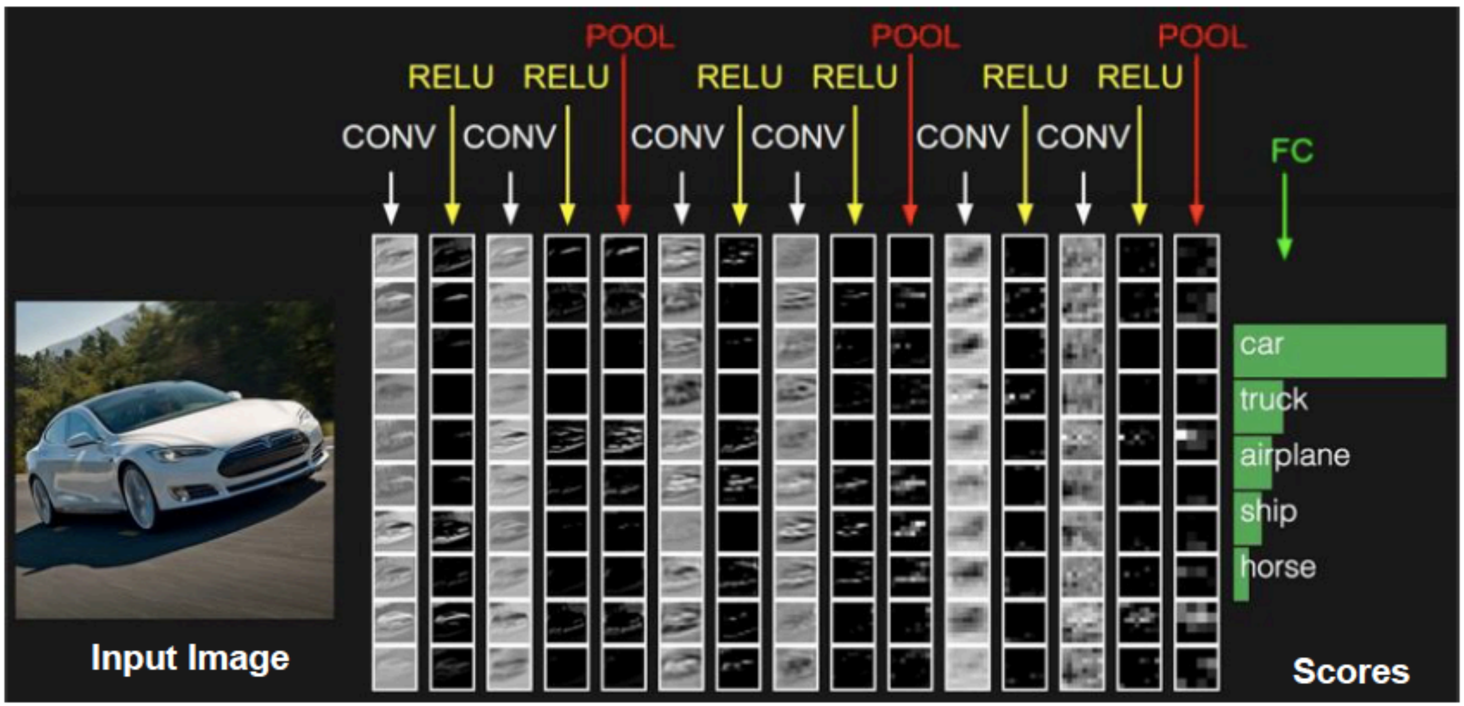


Image Source: Stanford cs231n

Datasets affect difficulty of task

Image Classification Summary

	MNIST	IMAGENET
Year	1998	2012
Resolution	28x28	256x256
Classes	10	1000
Training	60k	1.3M
Testing	10k	100k
Accuracy	0.21% error (ICML 2013)	2.25% top-5 error (2017 winner)

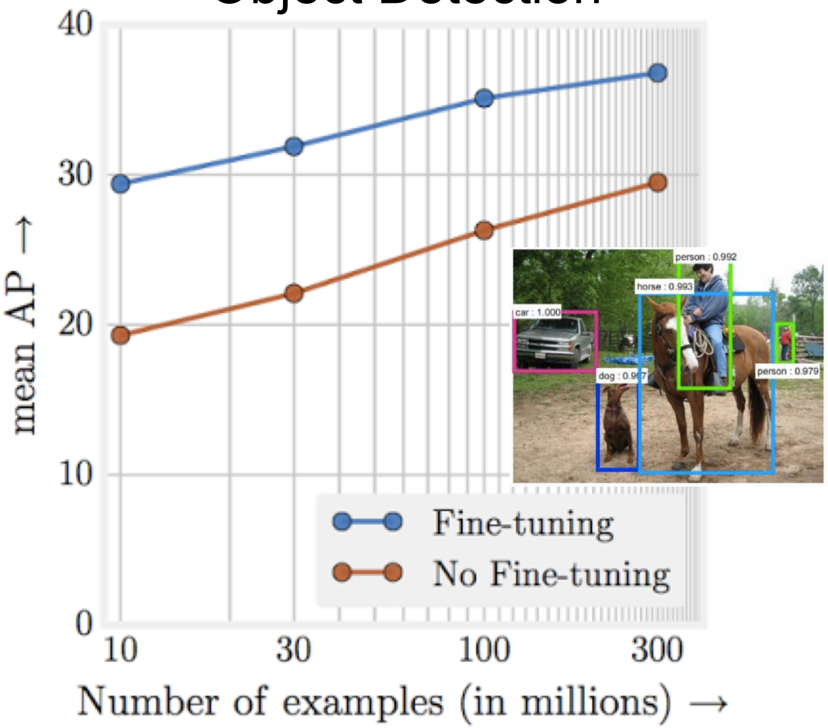
http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html

Effectiveness of More Data

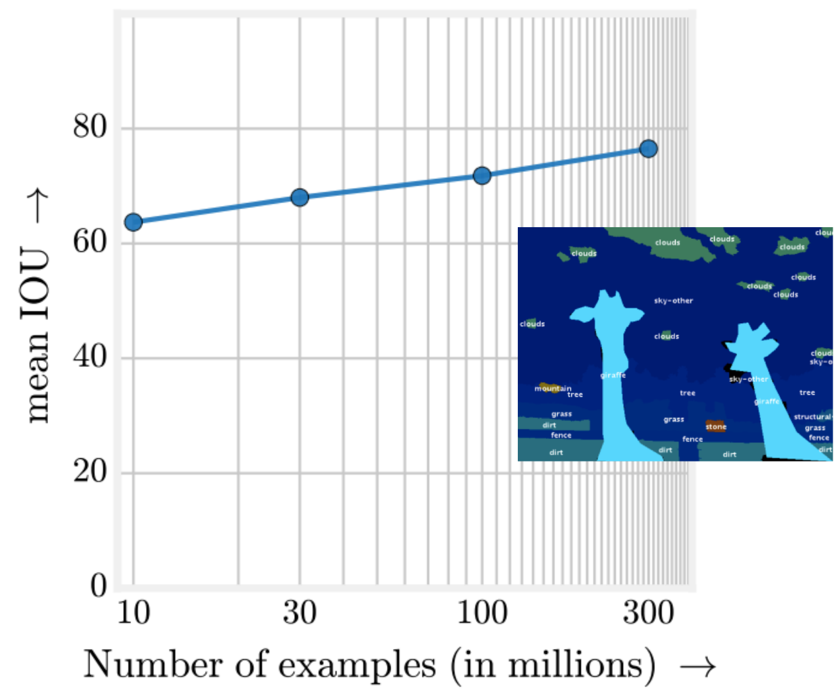
Accuracy increases logarithmically based on amount training data

Results from Google Internal Dataset
JFT-300M (300M images, 18291 categories)
Orders of magnitude larger than ImageNet

Object Detection



Semantic Segmentation



[Sun et al., ICCV 2017]

Recently Introduced Datasets

- Google Open Images (~9M images)
  - https://github.com/openimages/dataset
- Youtube-8M (8M videos)
  - https://research.google.com/youtube8m/
- AudioSet (2M sound clips)
  - https://research.google.com/audioset/index.html

Beyond CNN (CONV and FC Layers)

- **RNN and LSTM**
  - Often used for sequential data (e.g., speech recognition, machine translation, etc.) → 'seq2seq'
    (Note: CNNs can also be used for some of these applications)
  - Key operation is matrix multiplication

    Example 'Vanilla' RNN $h_t = tanh(W \bullet [h_{t-1}, x_t] + b)$

  → **FC layer approaches/optimizations can be applied**
- **Transformer**
  - Also matrix multiplication

Summary

- Development resources presented in this section enable us to evaluate hardware using the appropriate DNN model and dataset
  - Difficult tasks typically require larger models
  - Different datasets for different tasks
  - Number of datasets growing at a rapid pace