Course: Deep Learning

Unit 2: Computer Vision

Convolutional Neural Networks Architectures

Luis Baumela

Universidad Politécnica de Madrid

CNN Architectures

1. CNN architectures. In search of depth ...
   - Intro. LeNet.
   - AlexNet
   - VGG
   - GoogLeNet
   - ResNet

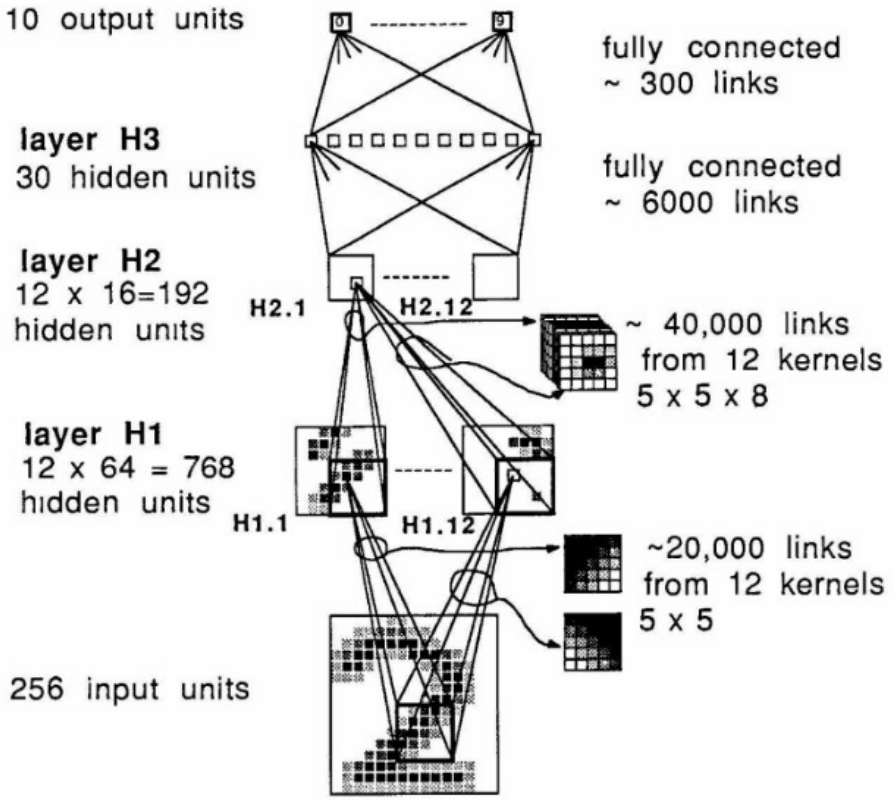
2. Modern architectures. In search of efficiency ...
   - Mobile Nets
   - Efficient Net

3. Encoder-decoder architectures. Applications.
   - FCN
   - UNet
   - Hourglass

CNNs architectures.

- First convolutional neural net

  It all started in 1989 ...



- Input: 16 x 16 grayvalue images
- Active: tanh
- Data set: 7291 training, 2007 test
- Performance: 95% success

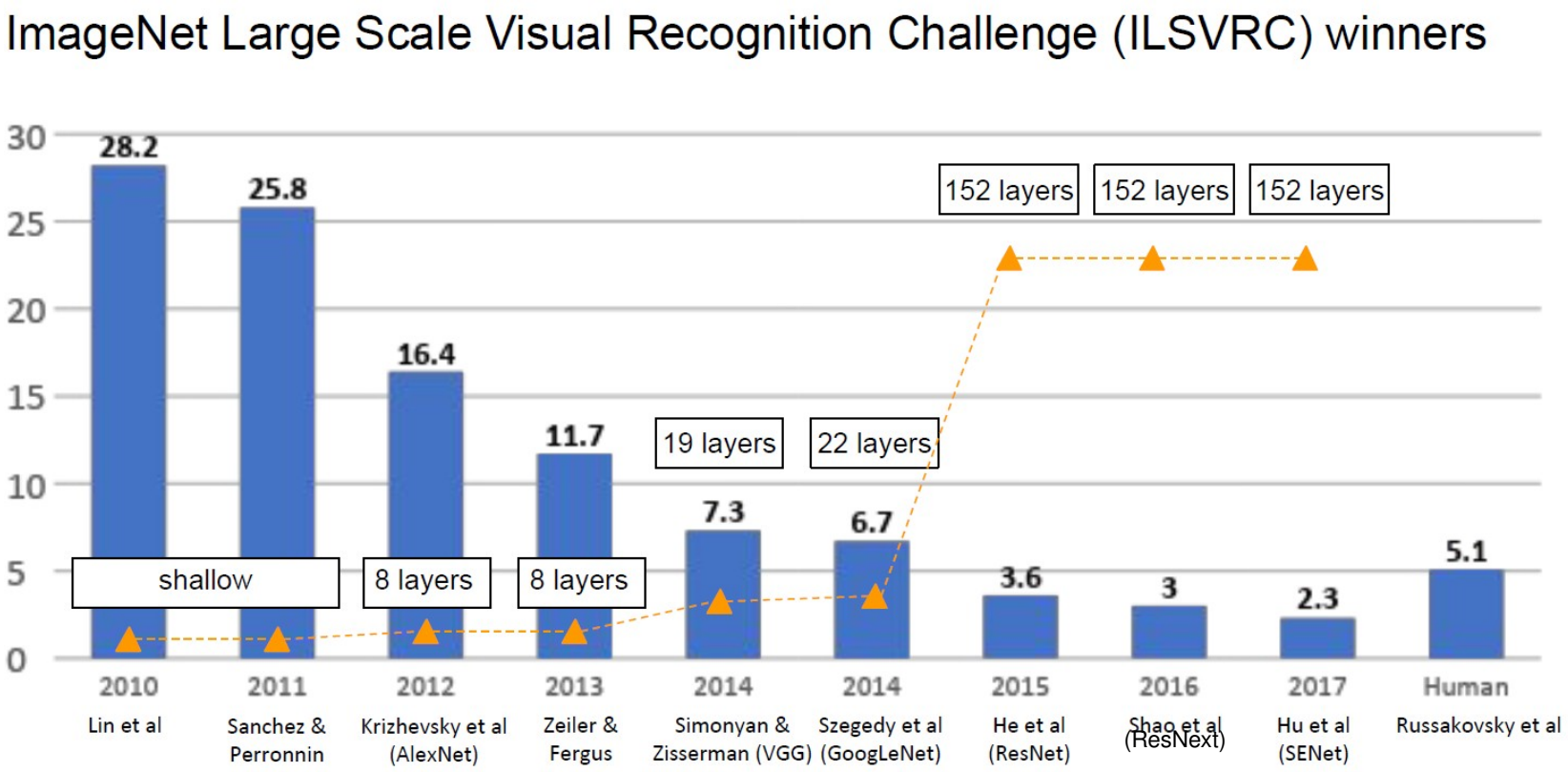
Key ideas:

- Spatial topology
- Weight sharing
- Subsampling (pooling)
- FC end classifier
- Backprop

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. *Backpropagation Applied to Handwritten Zip Code Recognition. Neural Computation*, Vol. 1, No. 4 , Pages: 541-551, 1989.

CNNs architectures. In search of depth

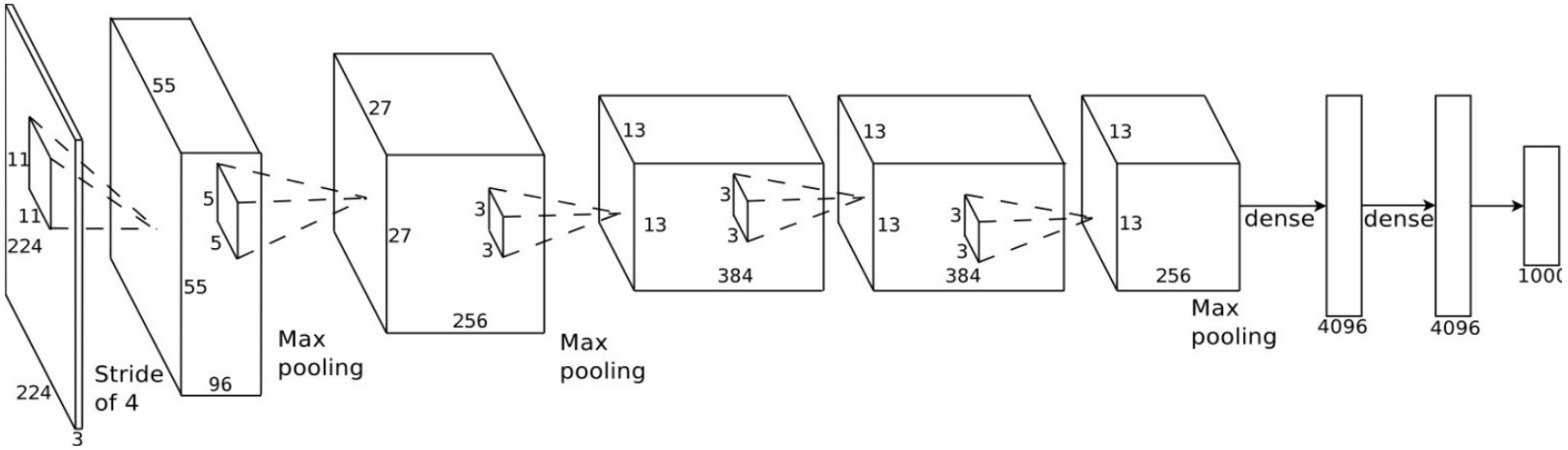
- CNN architectures



https://www.kaggle.com/getting-started/149448

CNNs architectures. In search of depth

- AlexNet (8 layers).



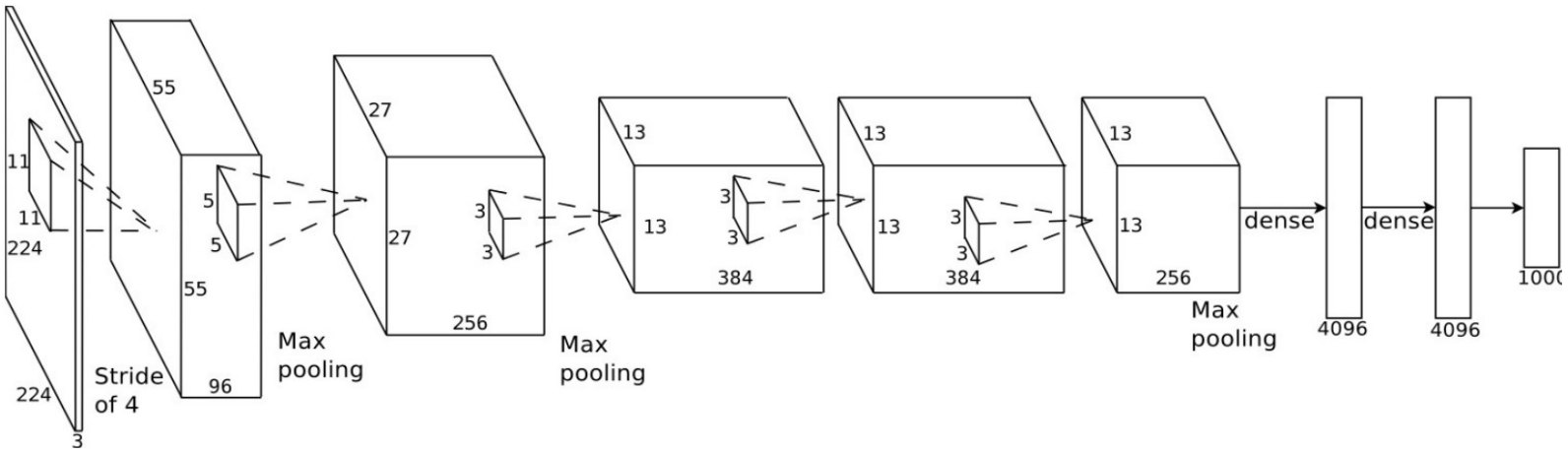
Design parameters:

- 650k neurons, 60M parameters, 630M connections.
- Trained on two NVIDIA GTX 580 GPUs for **a week** (~x30 speedup).
- Faster training with ReLU activation function (first deep model!)
- Data augmentation (training and testing)
- Optimization with SGD, lr0=0.01, manually reduced, momentum=0.9
- Regularization: L2 weight decay, Dropout (p=0.5) on last two fc layers.
- Written in Python/C++/CUDA

A. Krizhevsky, I. Sutskever, G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems* 25, pp. 1097–1105, 2012.

CNNs architectures. In search of depth

- AlexNet (8 layers).



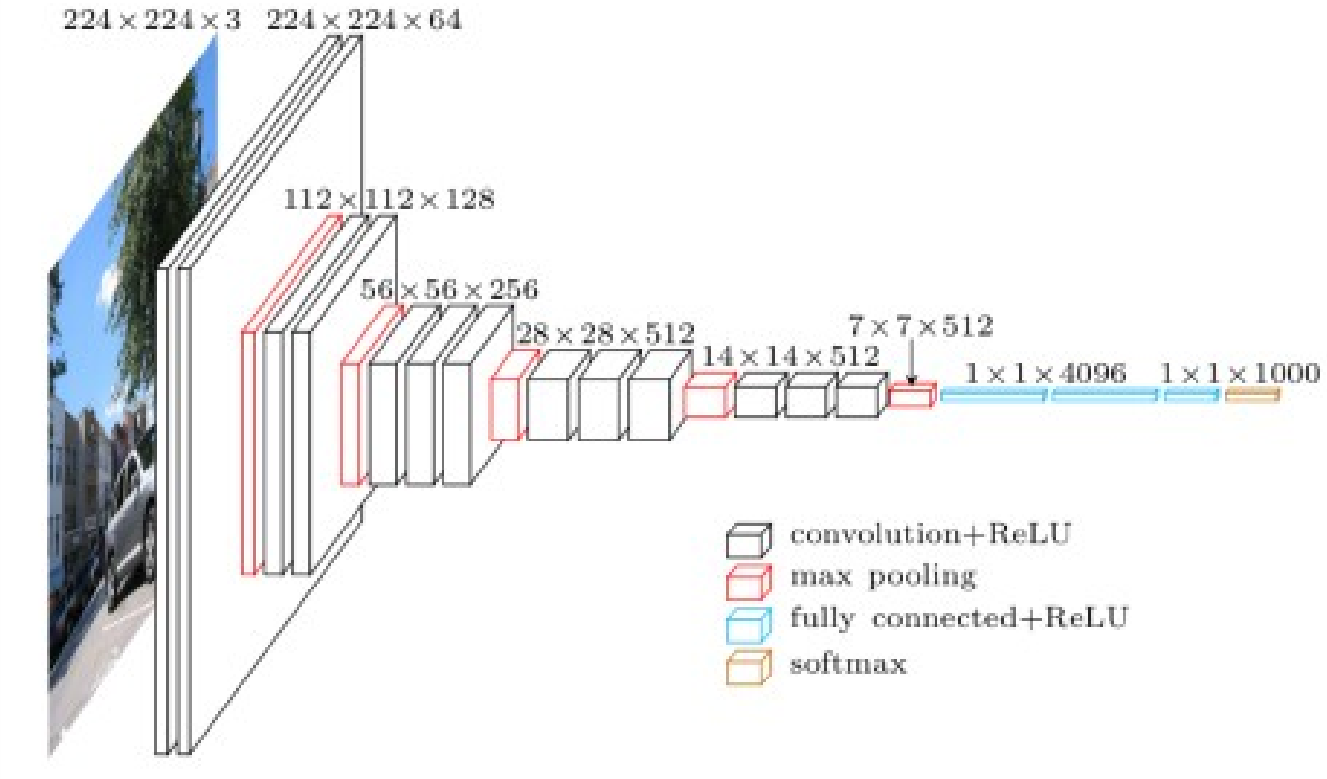
Key new ideas:

- Rectified Linear Unit non-linearity used for the first time
- GPU implementation
- Dropout regularization
- Data augmentation
- Grouped convolutions

A. Krizhevsky, I. Sutskever, G. Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems* 25, pp. 1097–1105, 2012.

CNNs architectures. In search of depth

- Visual Geometry Group (VGG) Net



Key ideas:

- Simple design: stacked conv+pool, **only** 3x3 filters stride 1, 2x2 maxpool stride 2

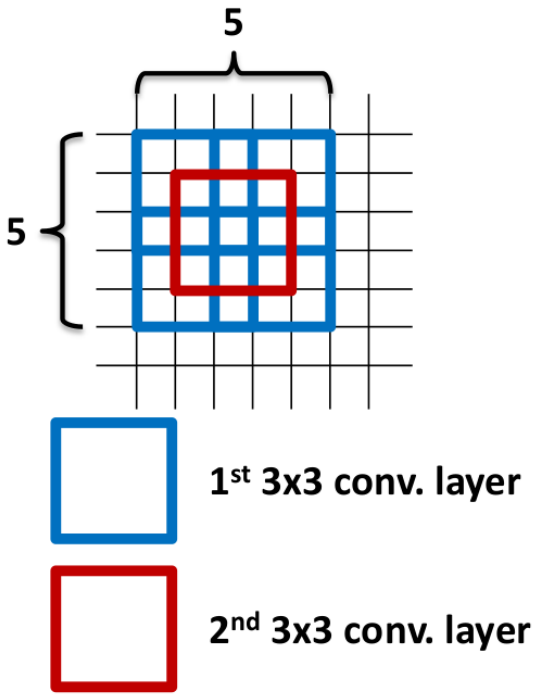
K. Simonyan, A. Zisserman. Very Deep ConvNets for Large-Scale Image Recognition. ICLR 2015

CNNs architectures. In search of depth

- Visual Geometry Group (VGG) Net

Why 3x3 layers?

- Stacked conv. layers have a large receptive field
  - two 3x3 layers – 5x5 receptive field
  - three 3x3 layers – 7x7 receptive field
- More non-linearity
- Less parameters to learn
  - ~140M per net



Key ideas:

- Simple design: stacked conv+pool, **only** 3x3 filters stride 1, 2x2 maxpool stride 2

K. Simonyan, A. Zisserman. Very Deep ConvNets for Large-Scale Image Recognition. ICLR 2015.

CNNs architectures. In search of depth

- Visual Geometry Group (VGG) Net

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					

Key ideas:

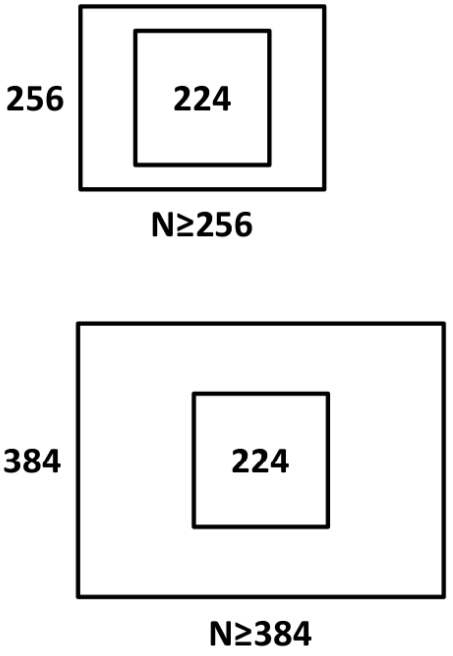
- Simple design: stacked conv+pool, **only** 3x3 filters stride 1, 2x2 maxpool stride 2
- Analyze the effect of depth from 11 to 19 layers
- As input space decreases, depth volume increases: double number of feature maps when spatial extent halves.
- Initialization with net A

K. Simonyan, A. Zisserman. Very Deep ConvNets for Large-Scale Image Recognition. ICLR 2015.

CNNs architectures. In search of depth

- Visual Geometry Group (VGG) Net

  - Multi-scale training
    - randomly-cropped ConvNet input
      - fixed-size 224x224
    - different training image size
      - 256xN
      - 384xN
      - [256;512]xN – random image size (scale jittering)
  - Standard jittering
    - random horizontal flips
    - random RGB shift



Key ideas:
- Simple design: stacked conv+pool, **only** 3x3 filters stride 1, 2x2 maxpool stride 2
- Analyze the effect of depth from 11 to 19 layers
- As input space decreases, depth volume increases.
- Data augmentation at train

K. Simonyan, A. Zisserman. Very Deep ConvNets for Large-Scale Image Recognition. ICLR 2015.

CNNs architectures. In search of depth

- Visual Geometry Group (VGG) Net

Method	top-1 val. error (%)	top-5 val. error (%)	top-5 test error (%)
VGG (2 nets, multi-crop & dense eval.)	**23.7**	**6.8**	**6.8**
VGG (1 net, multi-crop & dense eval.)	24.4	7.1	7.0
VGG (ILSVRC submission, 7 nets, dense eval.)	24.7	7.5	7.3
GoogLeNet (Szegedy et al., 2014) (1 net)	-	7.9	
GoogLeNet (Szegedy et al., 2014) (7 nets)	-	**6.7**	
MSRA (He et al., 2014) (11 nets)	-	-	8.1
MSRA (He et al., 2014) (1 net)	27.9	9.1	9.1
Clarifai (Russakovsky et al., 2014) (multiple nets)	-	-	11.7
Clarifai (Russakovsky et al., 2014) (1 net)	-	-	12.5
Zeiler & Fergus (Zeiler & Fergus, 2013) (6 nets)	36.0	14.7	14.8
Zeiler & Fergus (Zeiler & Fergus, 2013) (1 net)	37.5	16.0	16.1
OverFeat (Sermanet et al., 2014) (7 nets)	34.0	13.2	13.6
OverFeat (Sermanet et al., 2014) (1 net)	35.7	14.2	-
Krizhevsky et al. (Krizhevsky et al., 2012) (5 nets)	38.1	16.4	16.4
Krizhevsky et al. (Krizhevsky et al., 2012) (1 net)	40.7	18.2	-

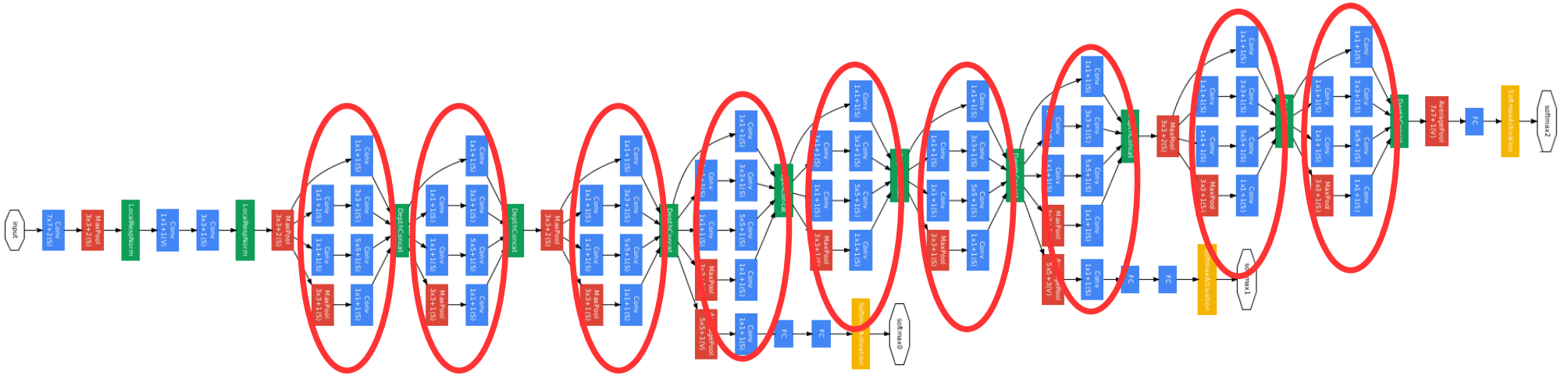
Key ideas:

- Simple design: stacked conv+pool, **only** 3x3 filters stride 1, 2x2 maxpool stride 2
- Analyze the effect of depth from 11 to 19 layers
- As input space decreases, depth volume increases.
- Data augmentation at train and test time.

K. Simonyan, A. Zisserman. Very Deep ConvNets for Large-Scale Image Recognition. ICLR 2015.

CNNs architectures. In search of depth

- GoogLeNet.



Key ideas:

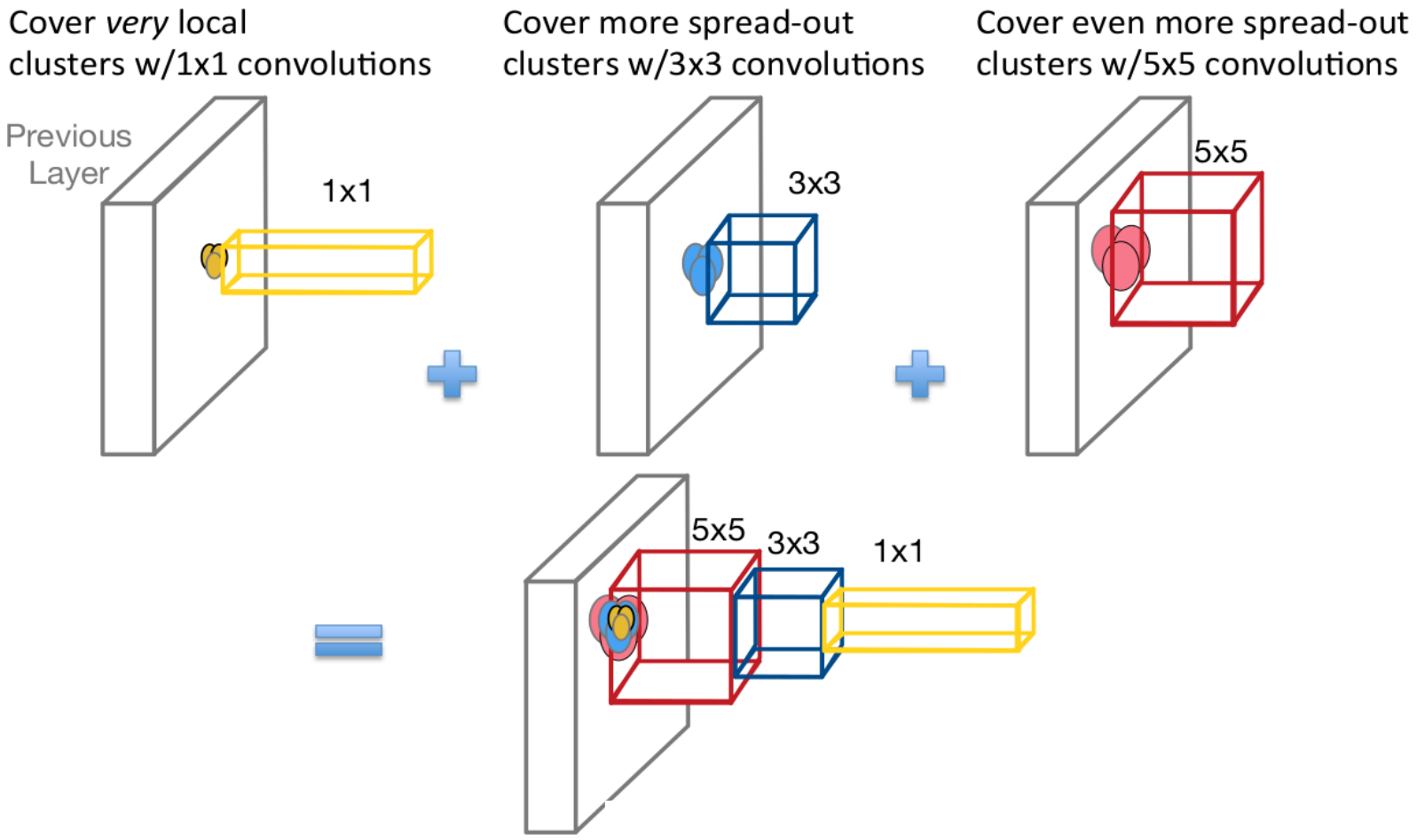
- Replicated modular structure (Inception, net in a net)
- Careful design to achieve computational savings:
  x12 less parameters than Alex Net: less memory, less overfitting → better accuracy.
  x2 less computations than Alex Net.
  More depth, less parameters.

C. Szegedy et al. Going Deeper with Convolutions. CVPR, 2015.

CNNs architectures. In search of depth

- GoogLeNet. **Inception module**

Intuition



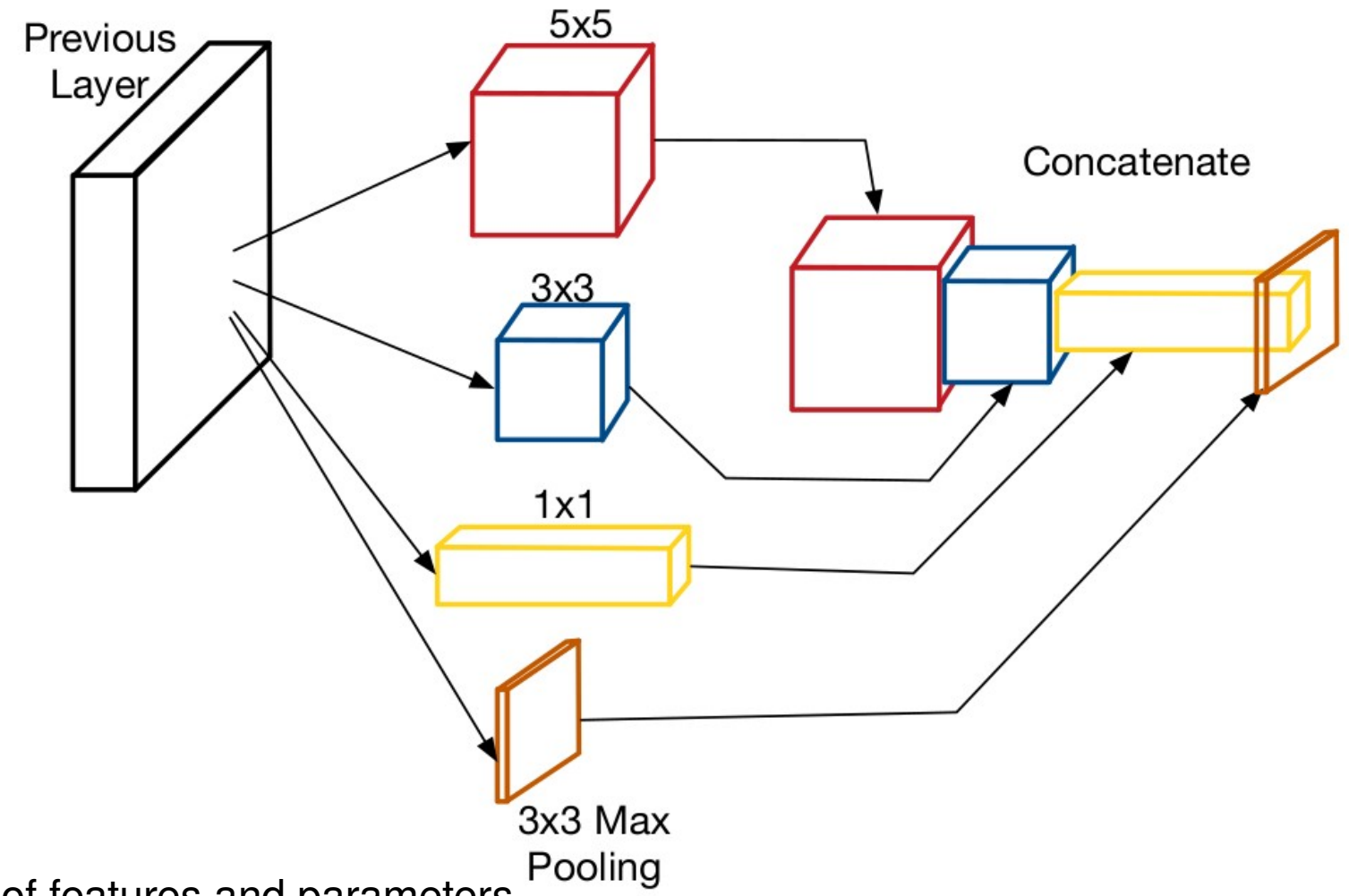
Key ideas:

- Exploit local correlations in parallel.

C. Szegedy et al. Going Deeper with Convolutions. CVPR, 2015.

CNNs architectures. In search of depth

- GoogLeNet. **Inception module**
  First conception



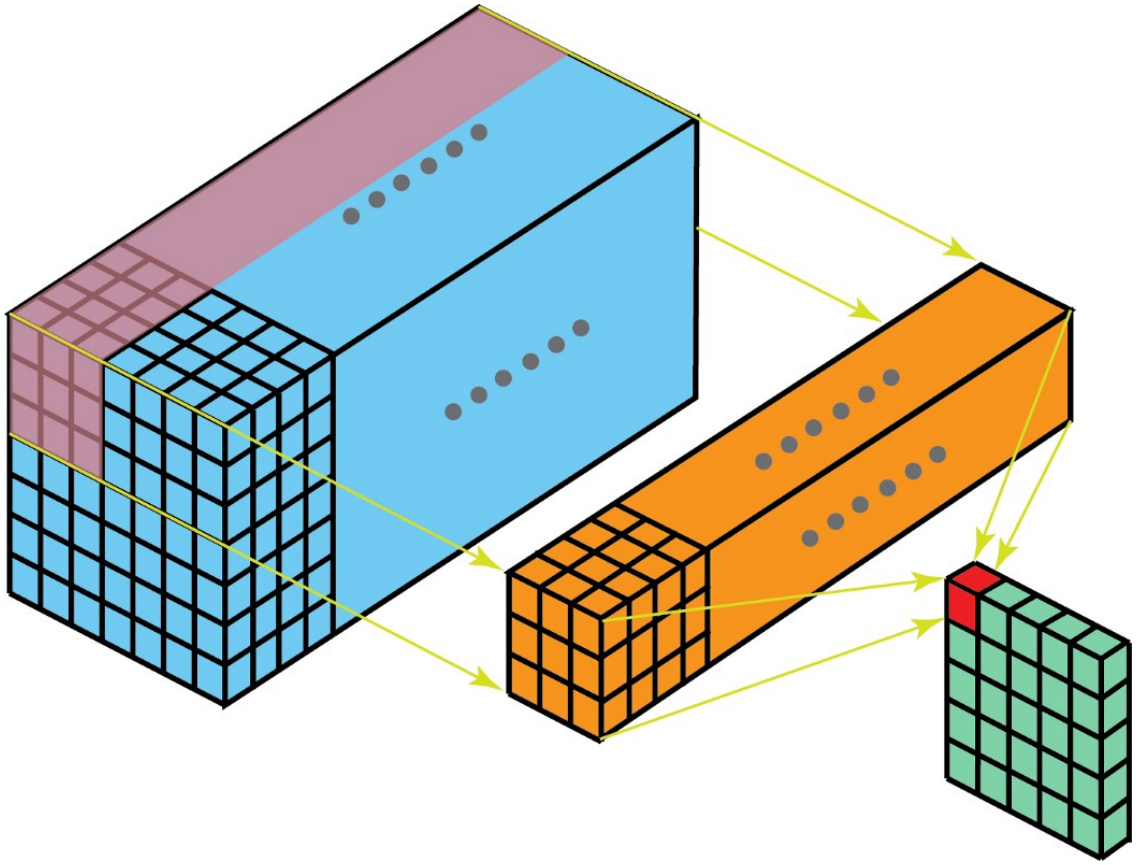
Problem:

- Explosion of features and parameters.

C. Szegedy et al. Going Deeper with Convolutions. CVPR, 2015.

CNNs architectures. In search of depth

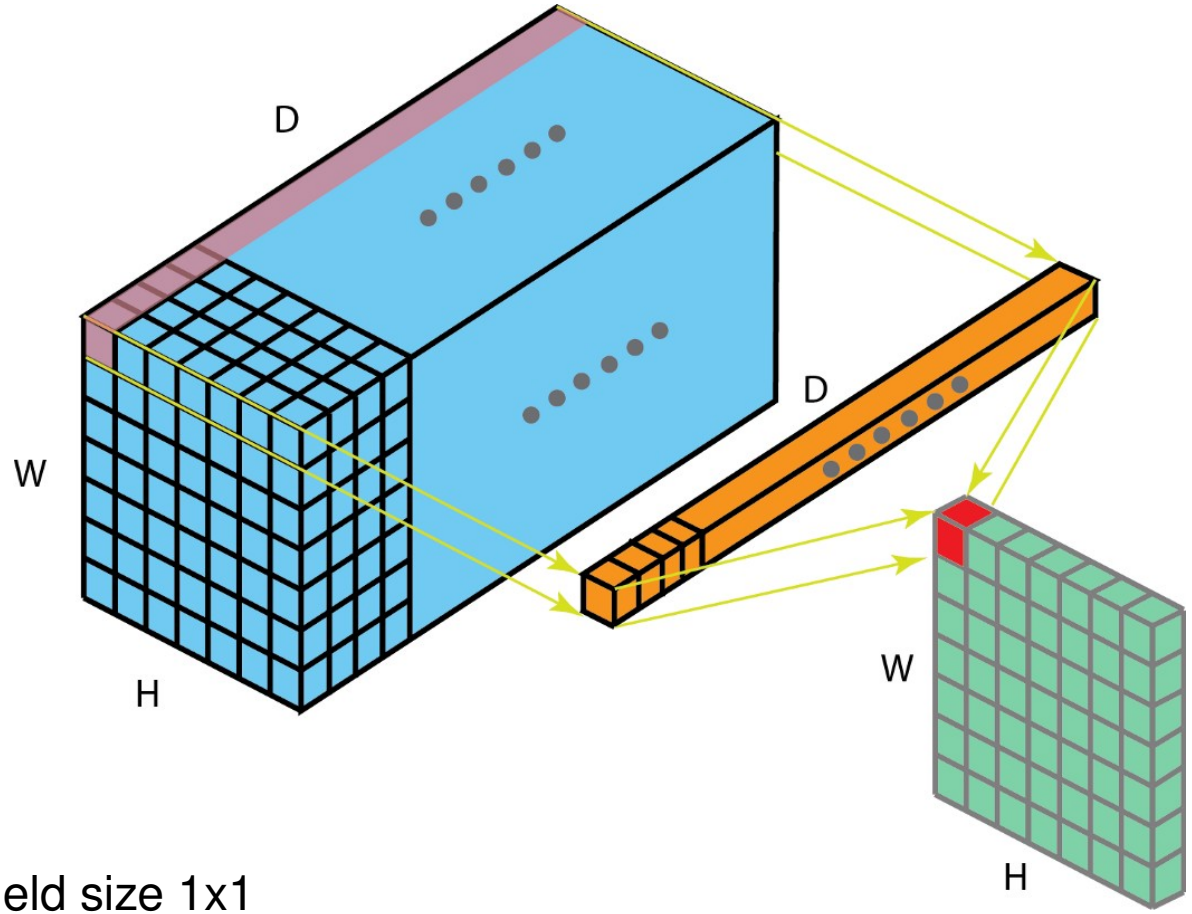
- GoogLeNet. **Inception module**
  Dimensionality reduction with 1x1 convolutions
  A spatial insight into a standard convolution



Kunlun Bai. A Comprehensive Introduction to Different Types of Convolutions in Deep Learning

CNNs architectures. In search of depth

- GoogLeNet. **Inception module**
- Dimensionality reduction with 1x1 convolutions



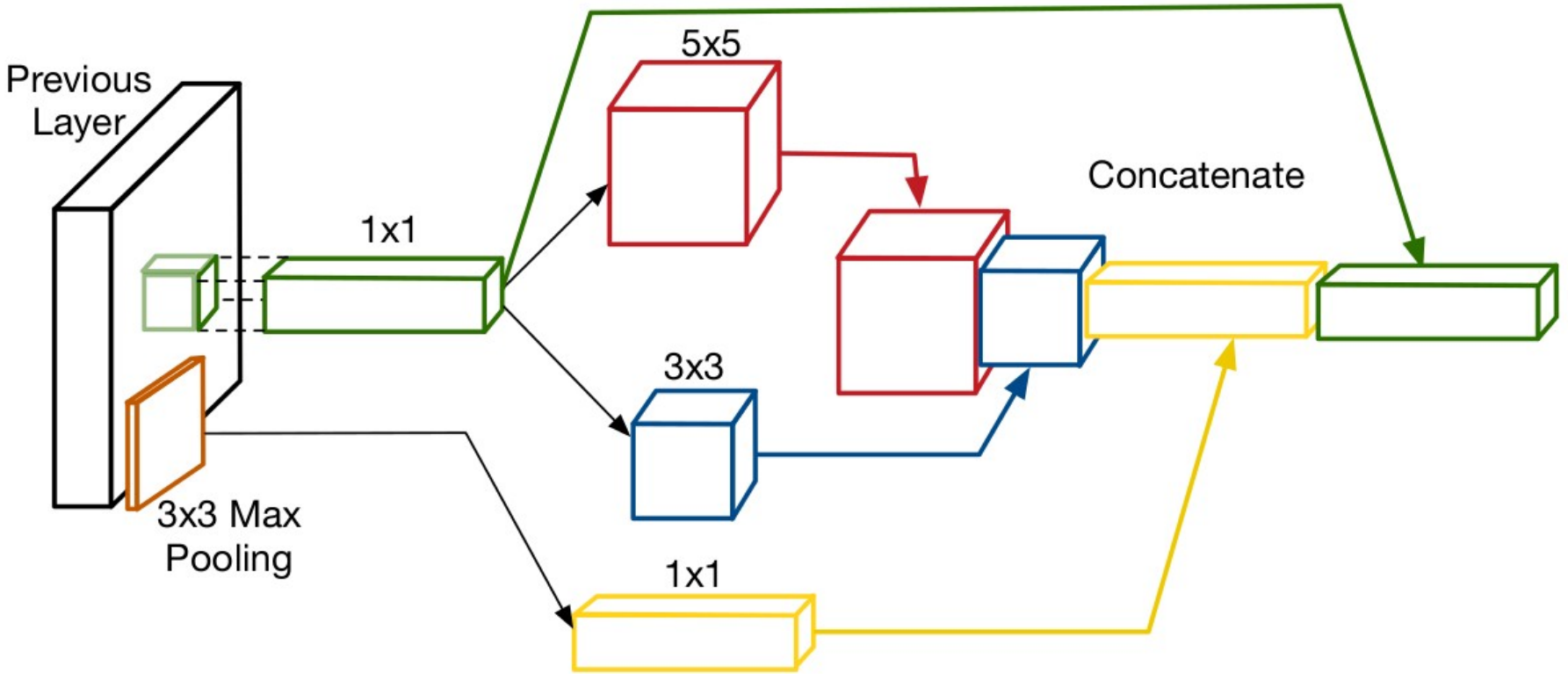
Key ideas:

- Receptive field size 1x1
- Input (w x h x d_1) → Output (w x h x d_2)
- Spatial dimension fixed, depth dimensio

Kunlun Bai. A Comprehensive Introduction to Different Types of Convolutions in Deep Learning

CNNs architectures. In search of depth

- GoogLeNet. **Inception module**
  Final design



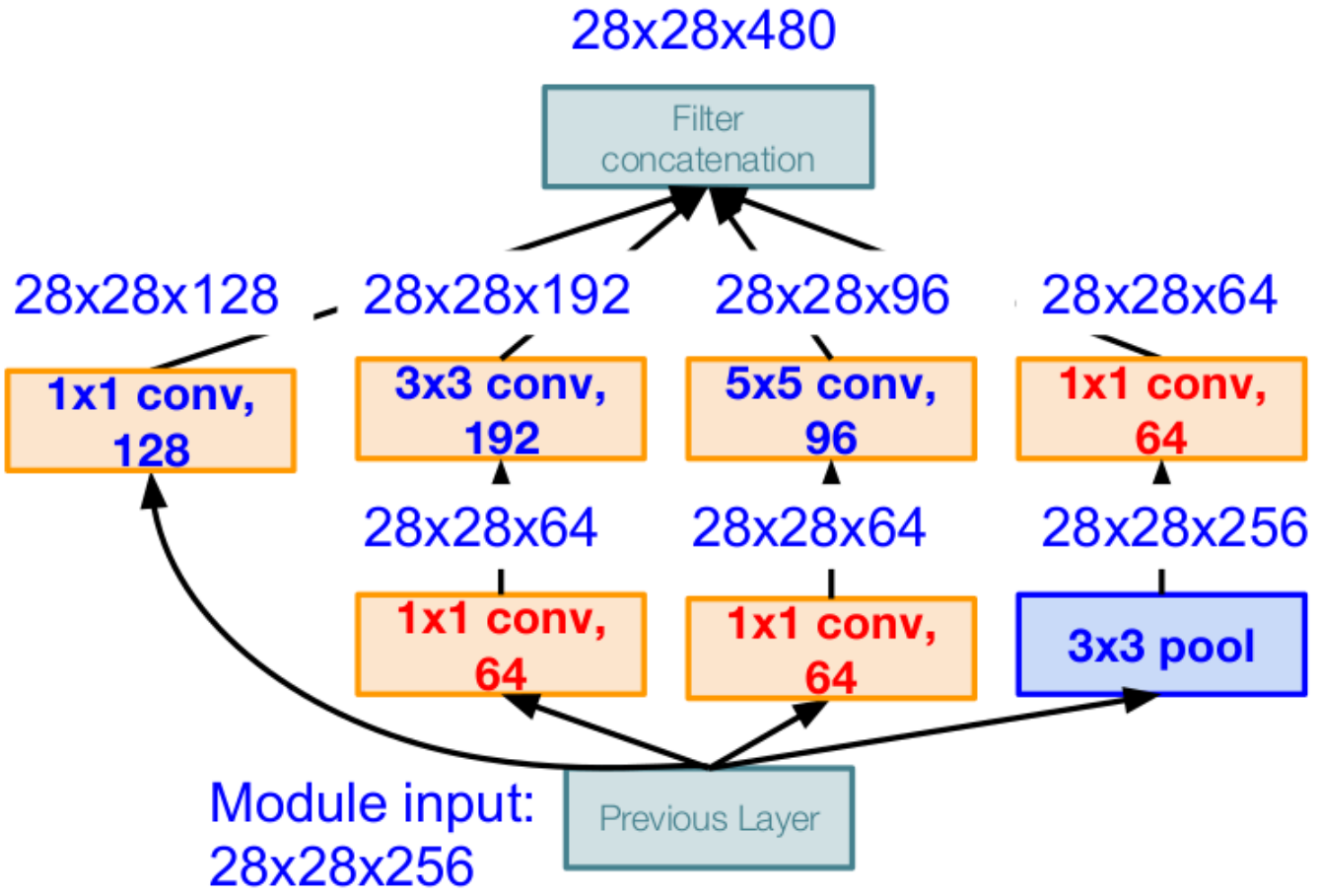
Solution:

- Use 1x1 convs to introduce “bottlenecks”: reduce the dimesionality of feature maps.

C. Szegedy et al. Going Deeper with Convolutions. CVPR, 2015.

CNNs architectures. In search of depth

- GoogLeNet. **Inception module**
  sample dimensions



C. Szegedy et al. Going Deeper with Convolutions. CVPR, 2015.

CNNs architectures. In search of depth

- GoogLeNet. **Inception module**

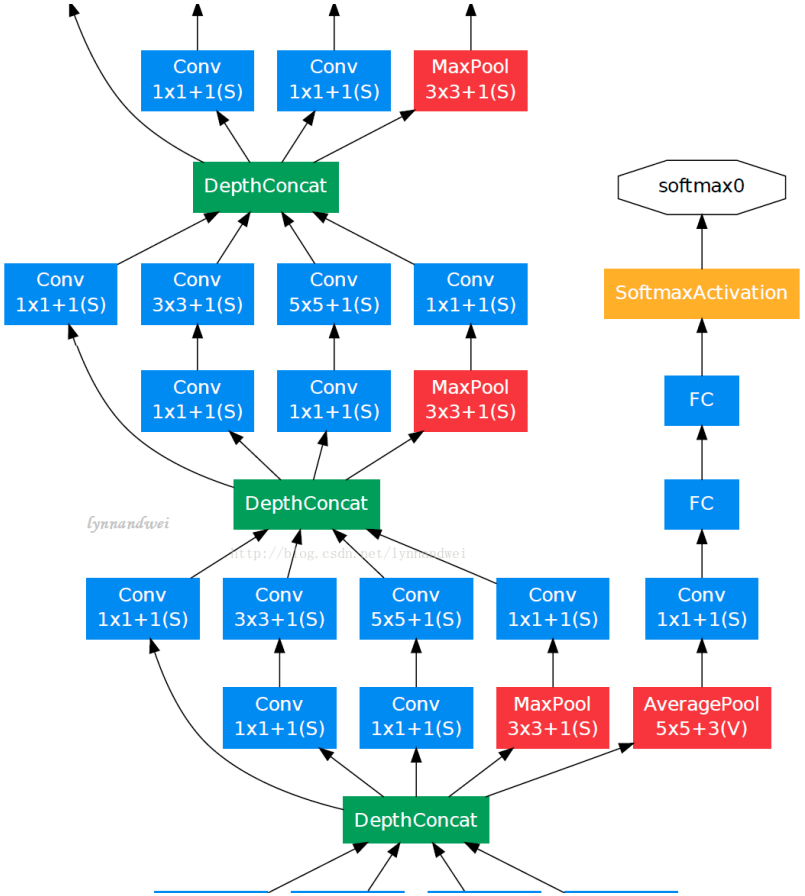
  Final design

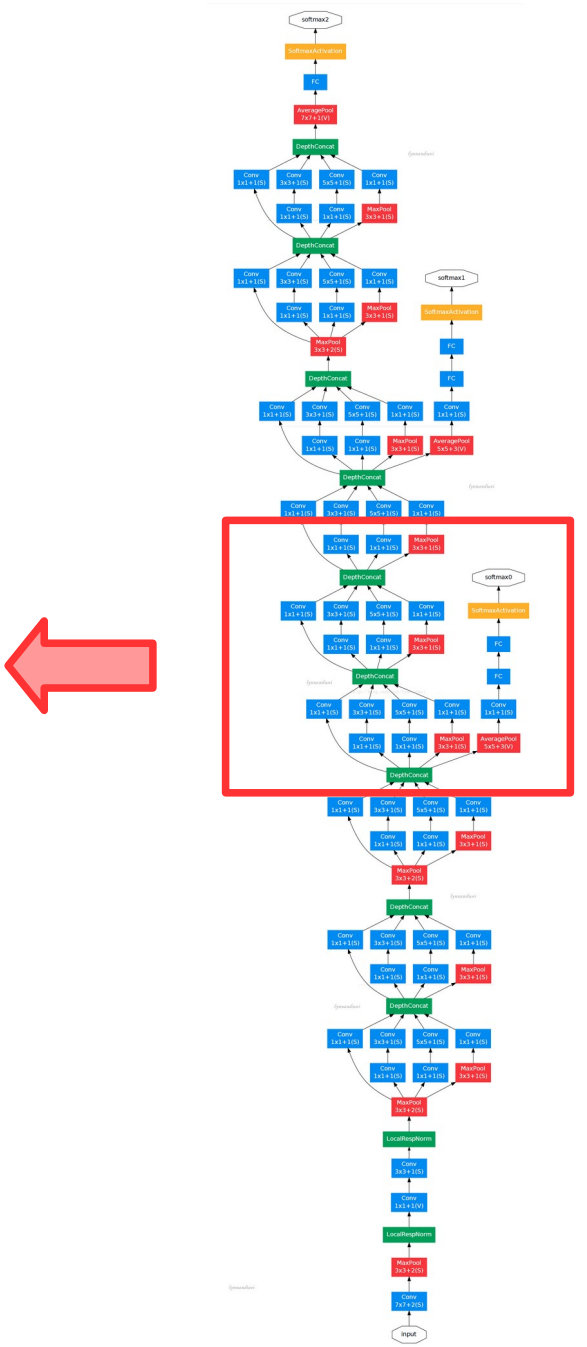
type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	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								

C. Szegedy et al. Going Deeper with Convolutions. CVPR, 2015.

CNNs architectures. In search of depth

- GoogLeNet. **Inception module**
  Final design





C. Szegedy et al. Going Deeper with Convolutions. CVPR, 2015.

CNNs architectures. In search of depth

- GoogLeNet. **Inception module**
  Key parameters and results

  - No FC layer (avg pool instead)
  - Only 5M params!
  - Computational efficiency (2x less AlexN)
  - At testing, multiple crops of image fed into the network, softmax probs averaged, resize the image to 4 scales and take 144 crops per image.
  - dropout layer with 70% ratio of dropped outputs.
  - SGD 0.9 momentum. fixed learning rate schedule (decreasing the learning rate by 4% every 8 epochs).

Team	Year	Place	Error (top-5)	Uses external data
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

Table 2: Classification performance.

Number of models	Number of Crops	Cost	Top-5 error	compared to base
1	1	1	10.07%	base
1	10	10	9.15%	-0.92%
1	144	144	7.89%	-2.18%
7	1	7	8.09%	-1.98%
7	10	70	7.62%	-2.45%
7	144	1008	6.67%	-3.45%

Table 3: GoogLeNet classification performance break down.

C. Szegedy et al. Going Deeper with Convolutions. CVPR, 2015.

CNNs architectures. In search of depth

- GoogLeNet. **Inception V2, V3 modules**

  Key new ideas

  - Introduce **Batch Normalization**
  - Improve convolution performance: (1xn)-factorization, conv stacking
  - Efficient grid size reduction

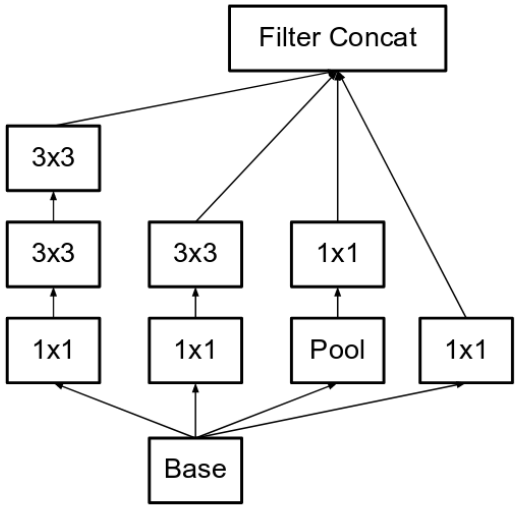


Figure 5. Inception modules where each 5×5 convolution is replaced by two 3×3 convolution, as suggested by principle 3 of Section 2.

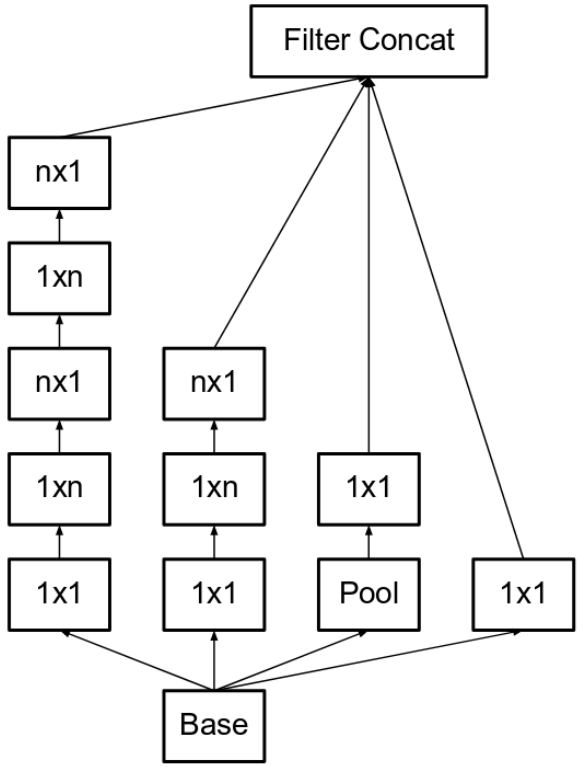


Figure 6. Inception modules after the factorization of the $n \times n$ convolutions. In our proposed architecture, we chose $n = 7$ for the 17×17 grid. (The filter sizes are picked using principle 3)

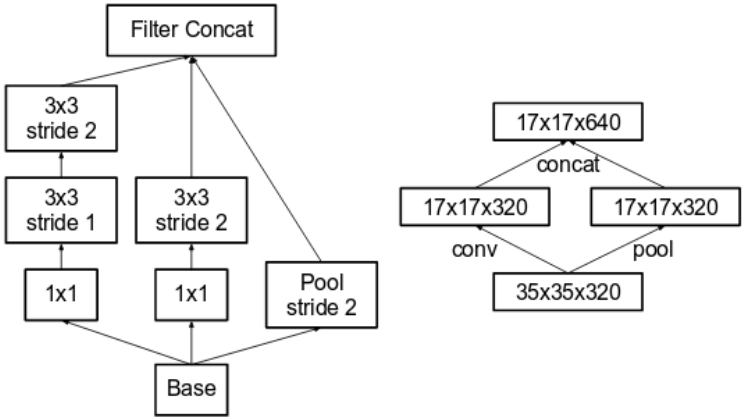


Figure 10. Inception module that reduces the grid-size while expands the filter banks. It is both cheap and avoids the representational bottleneck as is suggested by principle 1. The diagram on the right represents the same solution but from the perspective of grid sizes rather than the operations.

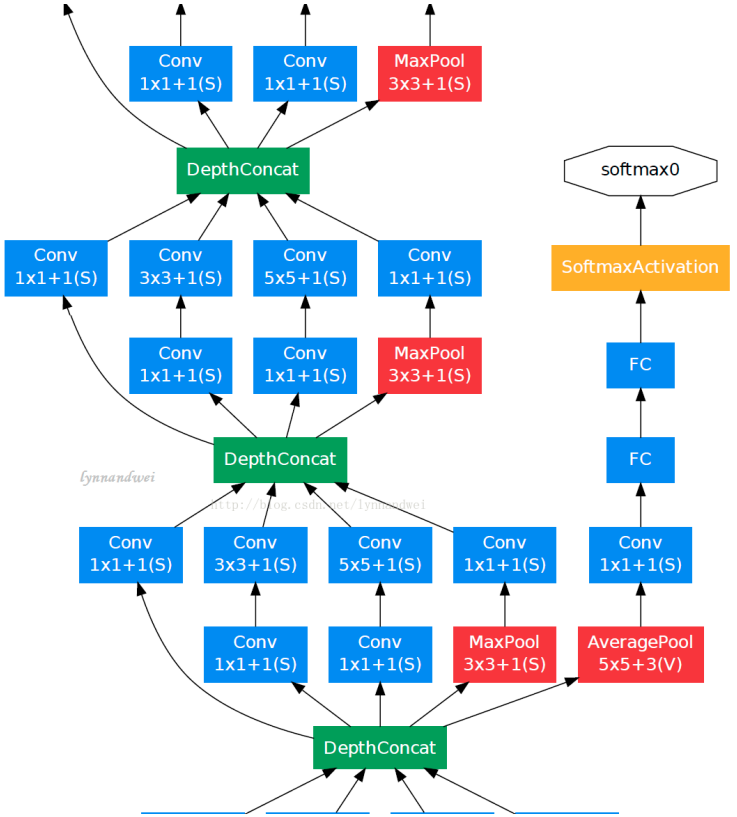
Iofe et al. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, 2015

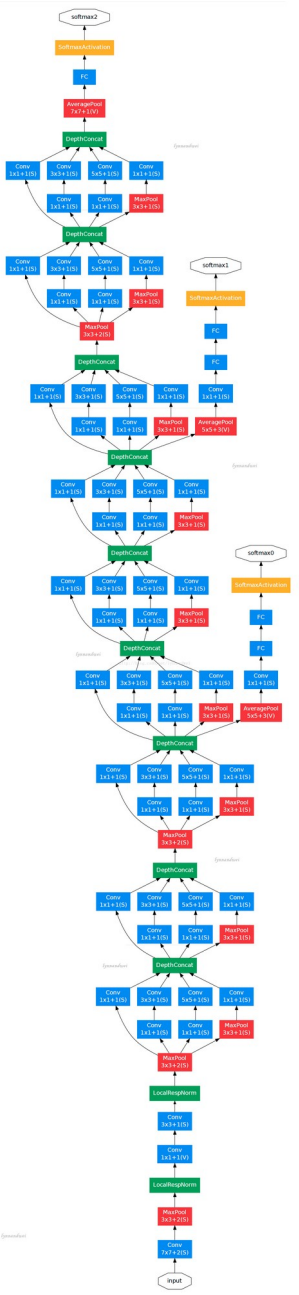
Szegedy et al. Rethinking the Inception Architecture for Computer Vision, 2015

CNNs architectures. In search of depth

- GoogLeNet. **Inception V2, V3 modules**
  Fundamental ideas common to inception nets

1. Net composed of replicated instances of a single module
2. Module is composed of multiple branches
3. Module has “bottlenecks”

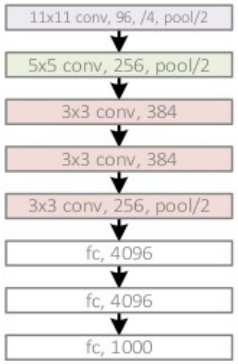




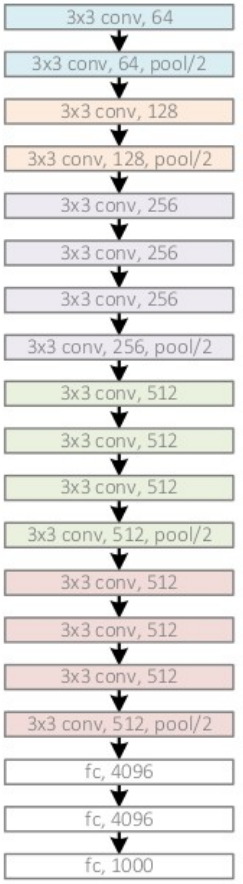
CNNs architectures. In search of depth

- Residual Networks (ResNet).

AlexNet, 8 layers
(ILSVRC 2012)



VGG, 19 layers
(ILSVRC 2014)

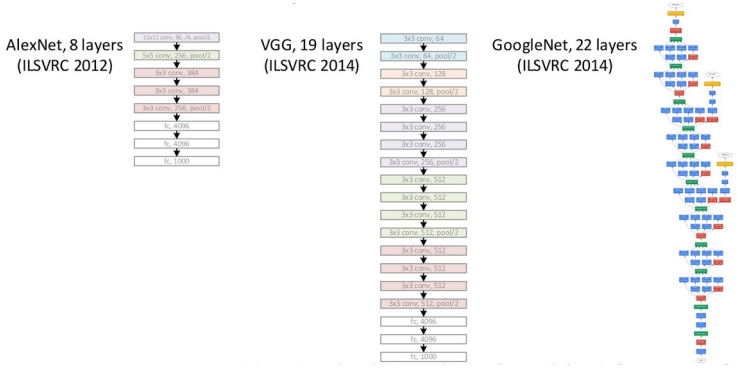


GoogleNet, 22 layers
(ILSVRC 2014)

K. He, X. Zhang, S. Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

CNNs architectures. In search of depth

- Residual Networks (ResNet).



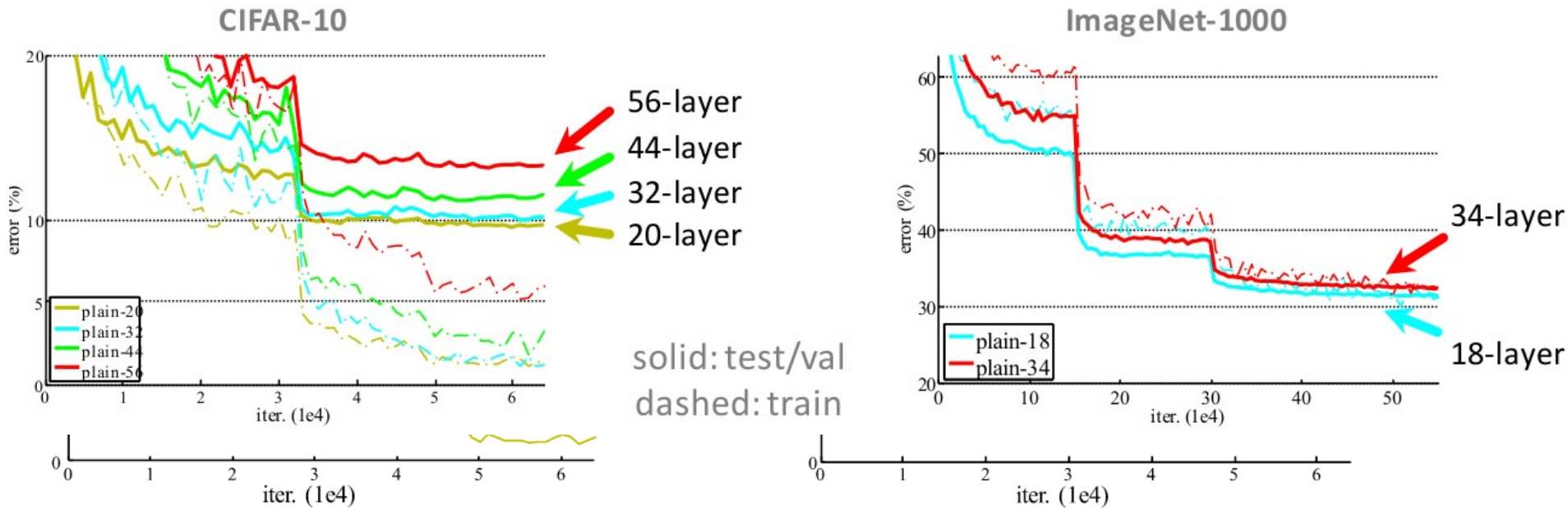
ResNet, 152 layers
(ILSVRC 2015)

K. He, X. Zhang, S. Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

CNNs architectures. In search of depth

- Residual Networks (ResNet).
  Introduction

  Just stack more and more layers to improve accuracy?

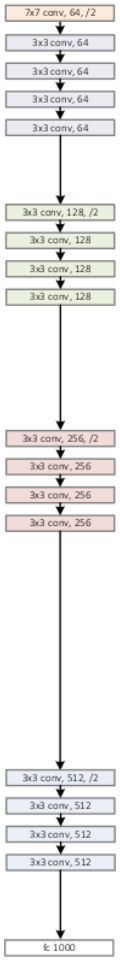


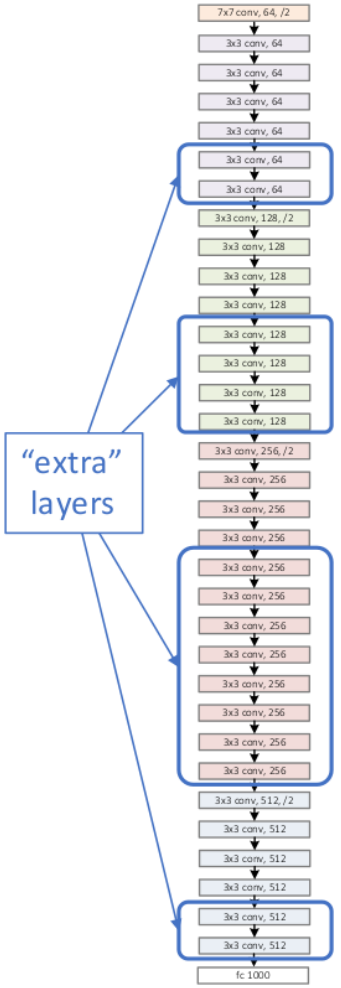
Optimization problems, the solver cannot cope with so many layers

CNNs architectures. In search of depth

- Residual Networks (ResNet).
  Introduction

18 layers model



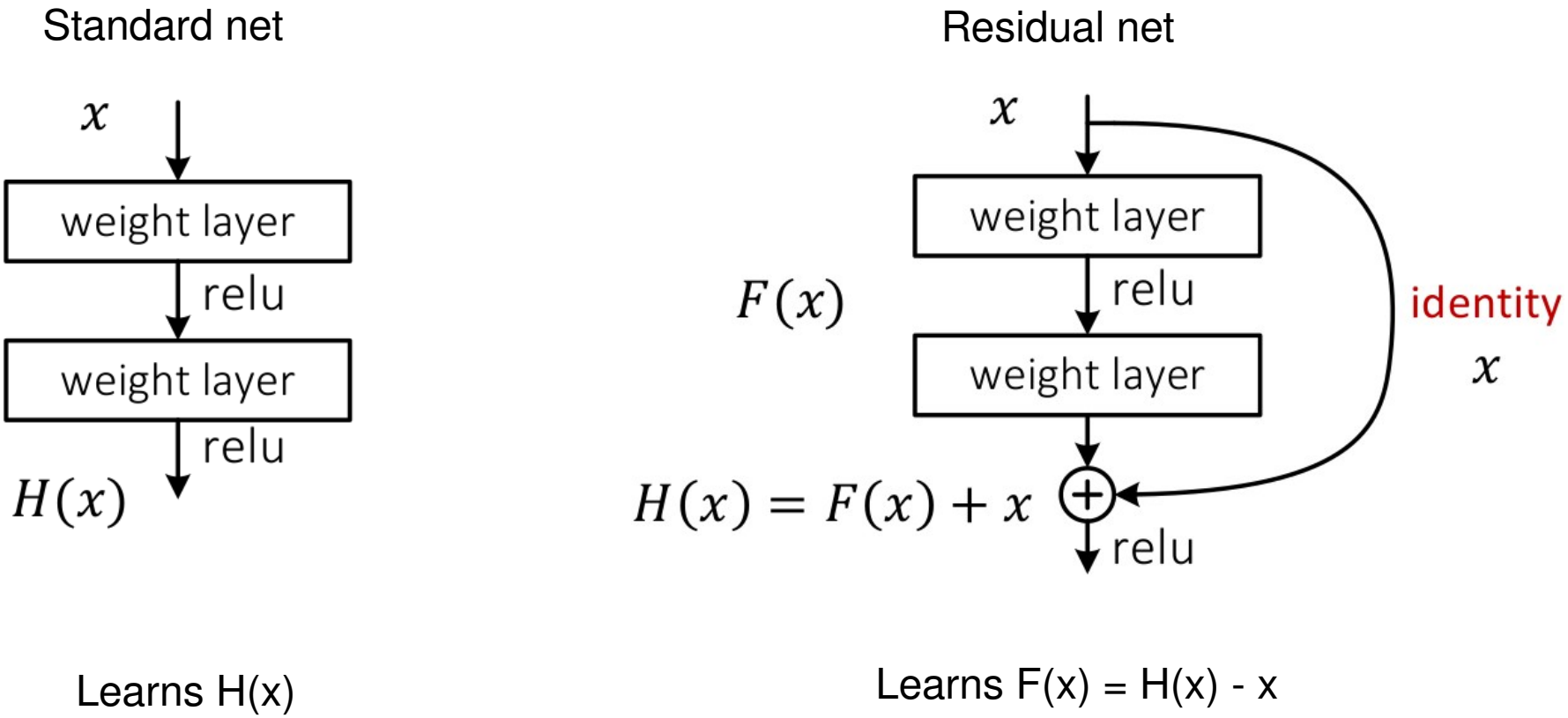


34 layers model

Same performance if extra layers set as identity.

CNNs architectures. In search of depth

- Residual Networks (ResNet).
  Fundaments



Learns H(x)

Learns F(x) = H(x) - x

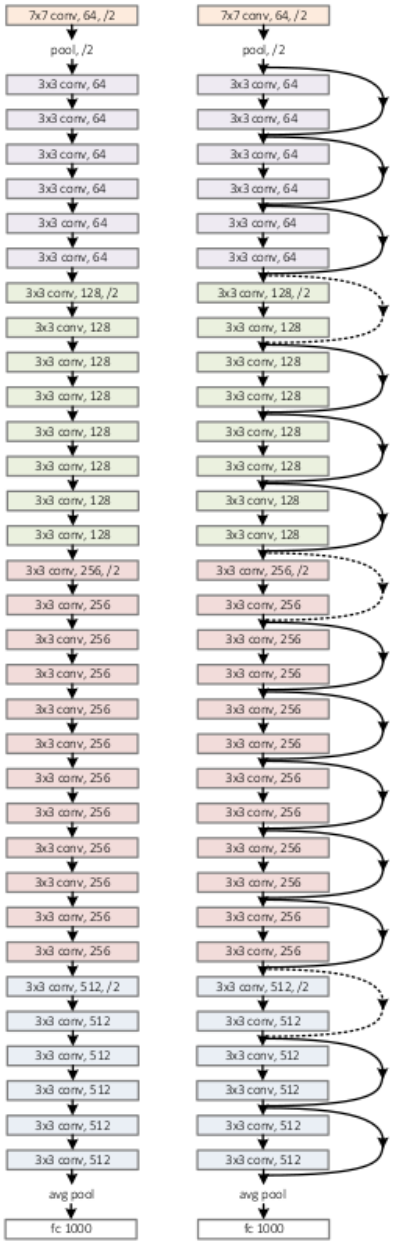
Residual Net: Easier to learn identity (or close to) mappings.

K. He, X. Zhang, S. Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

CNNs architectures. In search of depth

- Residual Networks (ResNet).
  Network design

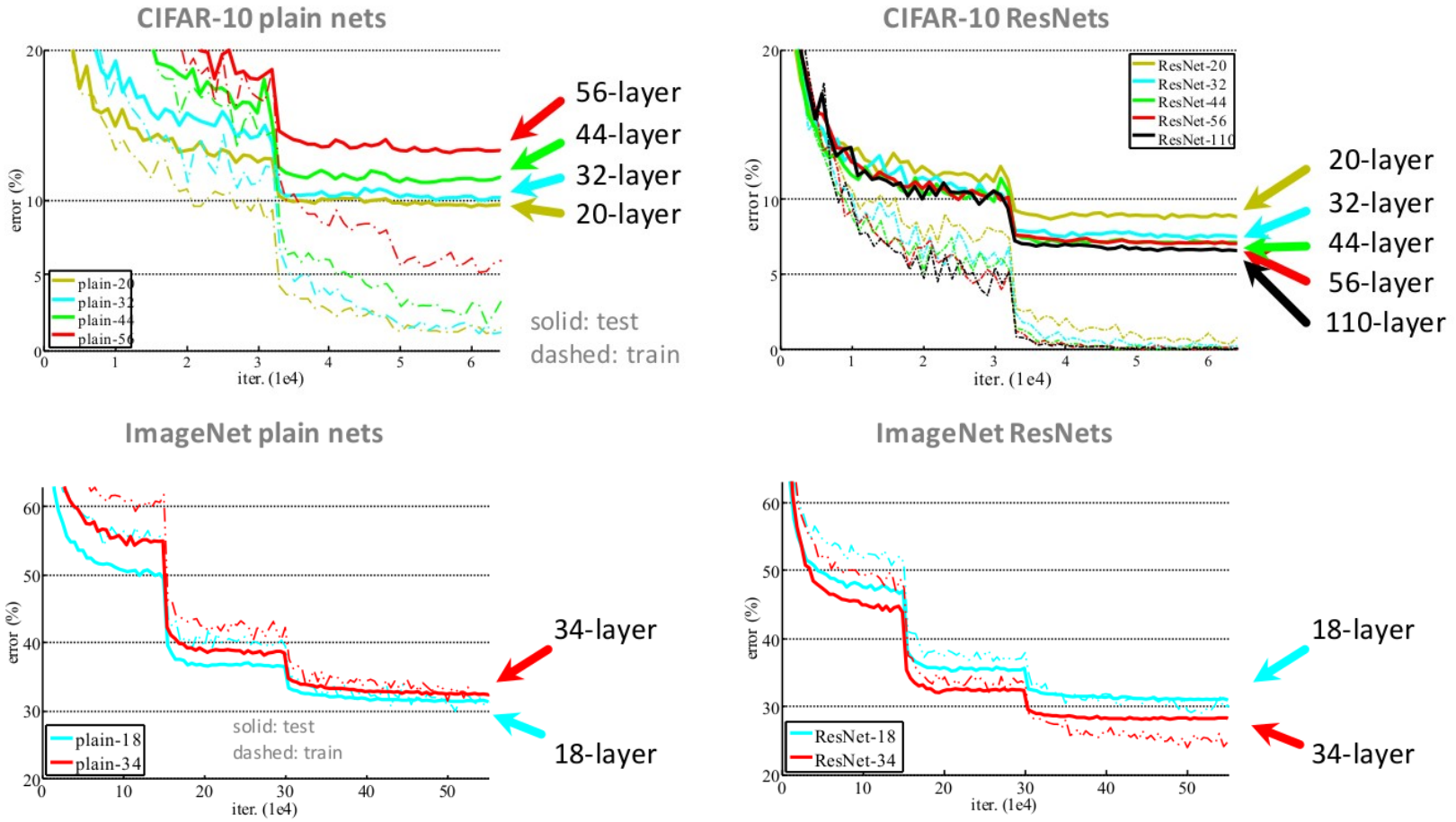
  - VGG style
    - ➔ only 3x3 convs, stride 1
    - ➔ 2x2 max pooling, stride 2
    - ➔ double number of filters (depth) after each pooling layer.
    - ➔ only one fc layer
  - Batch normalization
  - No dropout
  - Data augmentation



K. He, X. Zhang, S. Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

CNNs architectures. In search of depth

- Residual Networks (ResNet).
  Initial results

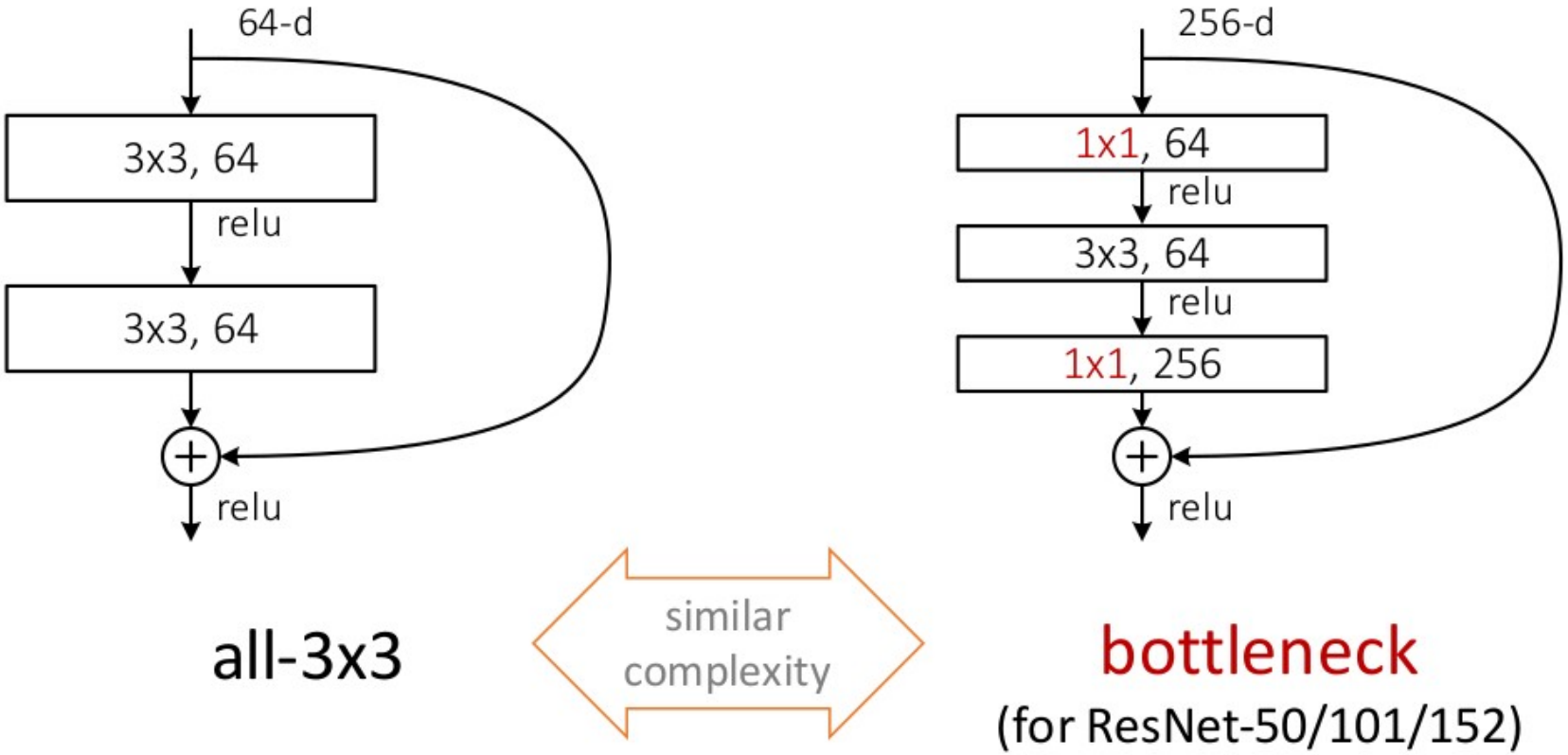


Residual Nets behave as expected from machine learning intuition.

K. He, X. Zhang, S. Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

CNNs architectures. In search of depth

- Residual Networks (ResNet).
  Building block refinement



CNNs architectures. In search of depth

- Residual Networks (ResNet).
  Network design

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

K. He, X. Zhang, S. Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

CNNs architectures. In search of depth

- Residual Networks (ResNet).
  Results

method	top-1 err.	top-5 err.
VGG [40] (ILSVRC'14)	-	8.43†
GoogLeNet [43] (ILSVRC'14)	-	7.89
VGG [40] (v5)	24.4	7.1
PReLU-net [12]	21.59	5.71
BN-inception [16]	21.99	5.81
ResNet-34 B	21.84	5.71
ResNet-34 C	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

single-model results on ImageNet

method	top-5 err. (**test**)
VGG [40] (ILSVRC'14)	7.32
GoogLeNet [43] (ILSVRC'14)	6.66
VGG [40] (v5)	6.8
PReLU-net [12]	4.94
BN-inception [16]	4.82
ResNet (ILSVRC'15)	**3.57**

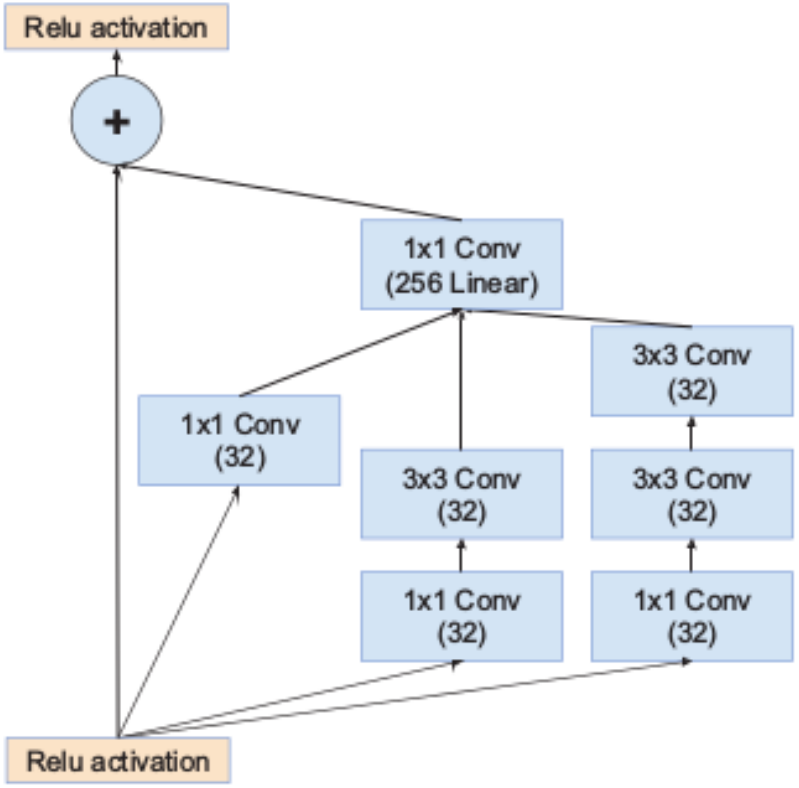
ensamble results on ImageNet

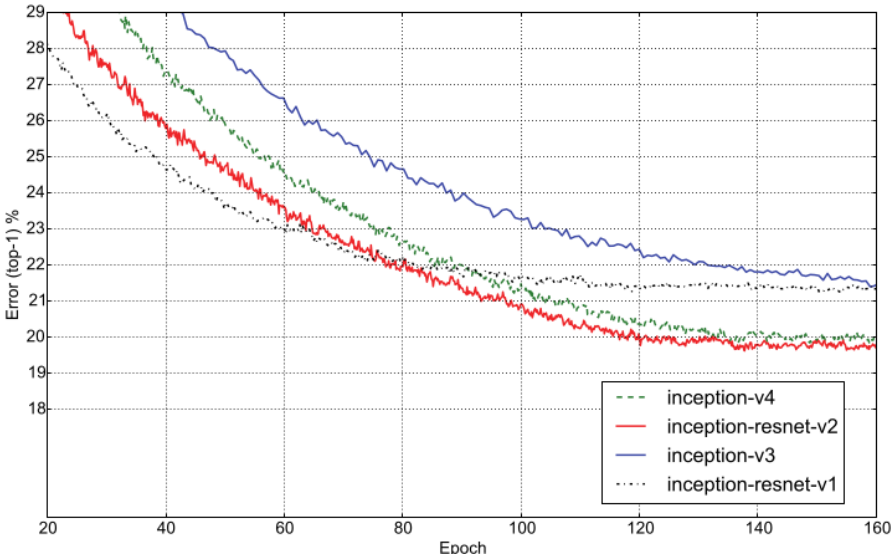
K. He, X. Zhang, S. Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

CNNs architectures. In search of depth

- Inception-ResNet architecture.

  Same initial philosophy of ResNet or GoogleNet, but now both combine ResNet shortcuts with Inception multiple branches

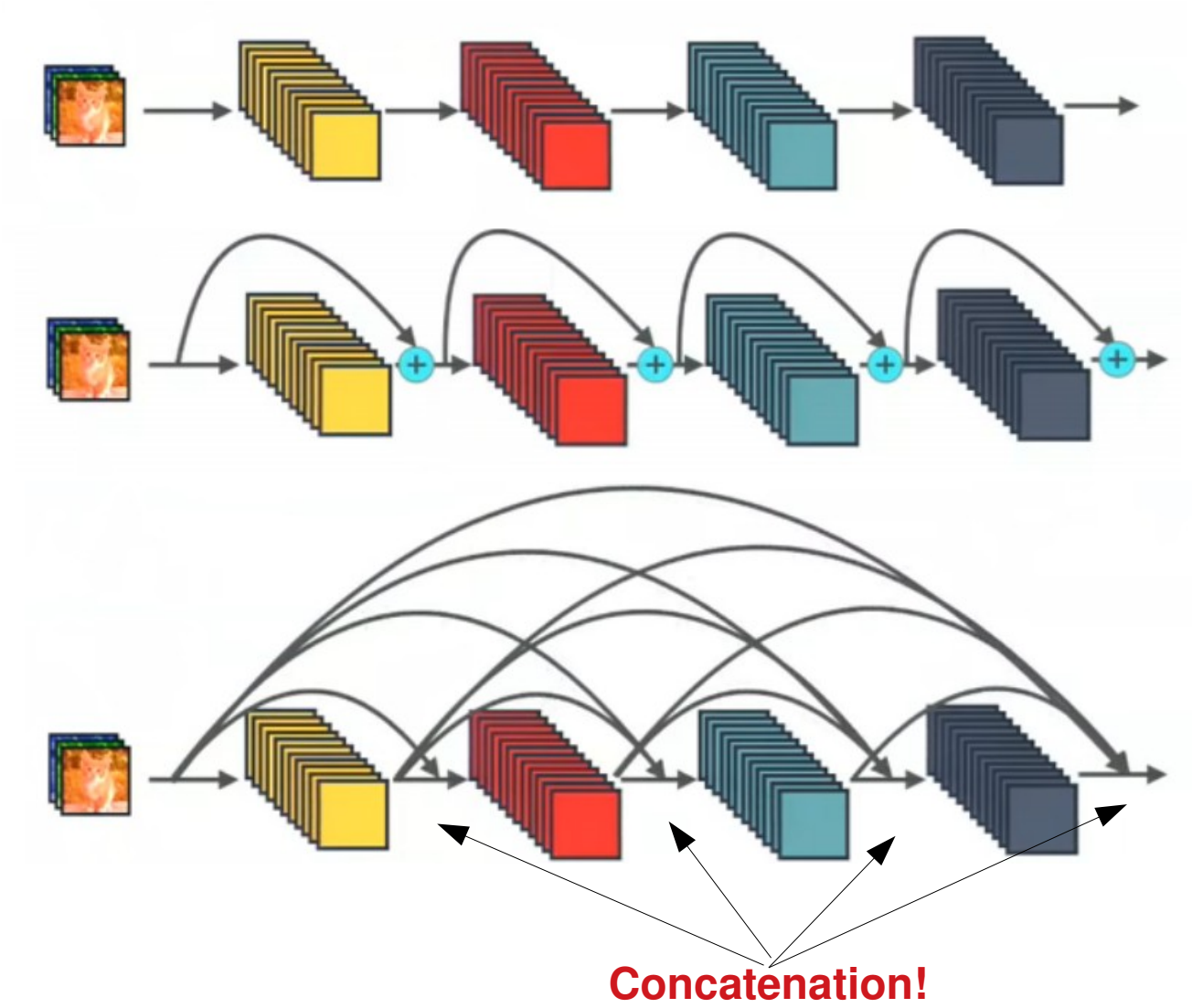




Szegedy et al. Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. AAAI 2017

CNNs architectures. In search of depth

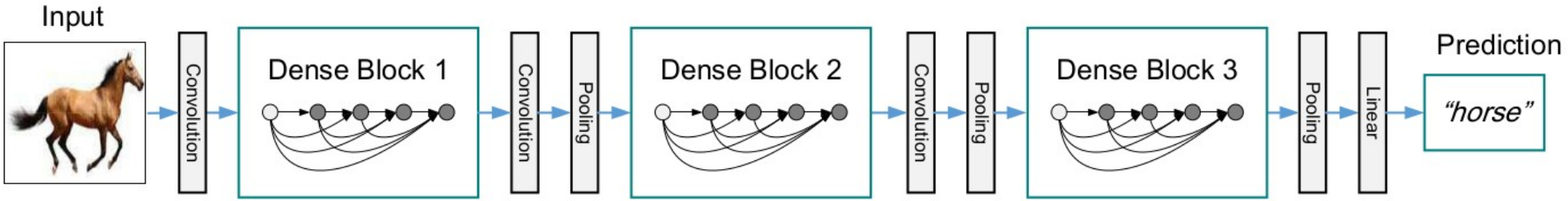
- DenseNet. Dense block



Huang et al. Densely Connected Convolutional Networks. CVPR 2017

CNNs architectures. In search of depth

- DenseNet. Dense Net architecture.
  Sample architecture:



- Properties
  - Feature reuse. Compact parameter efficient models.
  - Improved gradient flow. Direct access to gradients from output and input singnals.
  - Implicit regularization. Dense connections reduce overfitting.

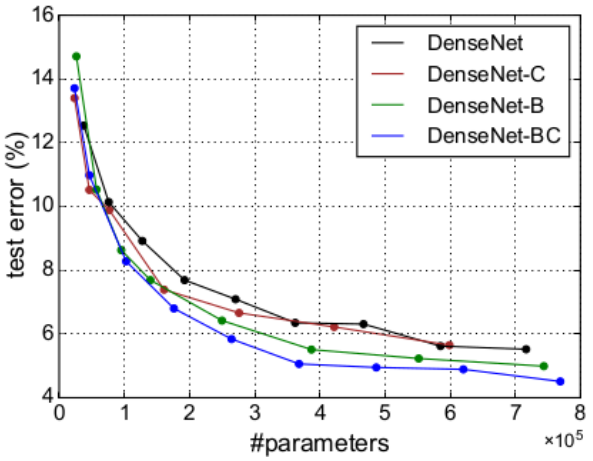
Huang et al. Densely Connected Convolutional Networks. CVPR 2017

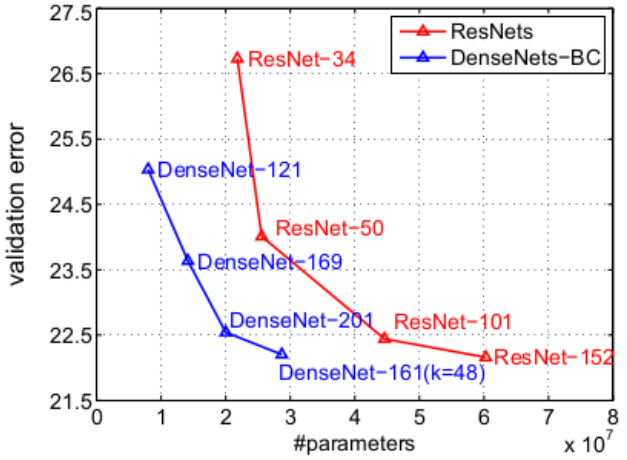
CNNs architectures. In search of depth

- DenseNet. Experimental results.

ImageNet results

Model	top-1	top-5
DenseNet-121 ($k=32$)	25.02 (23.61)	7.71 (6.66)
DenseNet-169 ($k=32$)	23.80 (22.08)	6.85 (5.92)
DenseNet-201 ($k=32$)	22.58 (21.46)	6.34 (5.54)
DenseNet-161 ($k=48$)	22.33 (20.85)	6.15 (5.30)



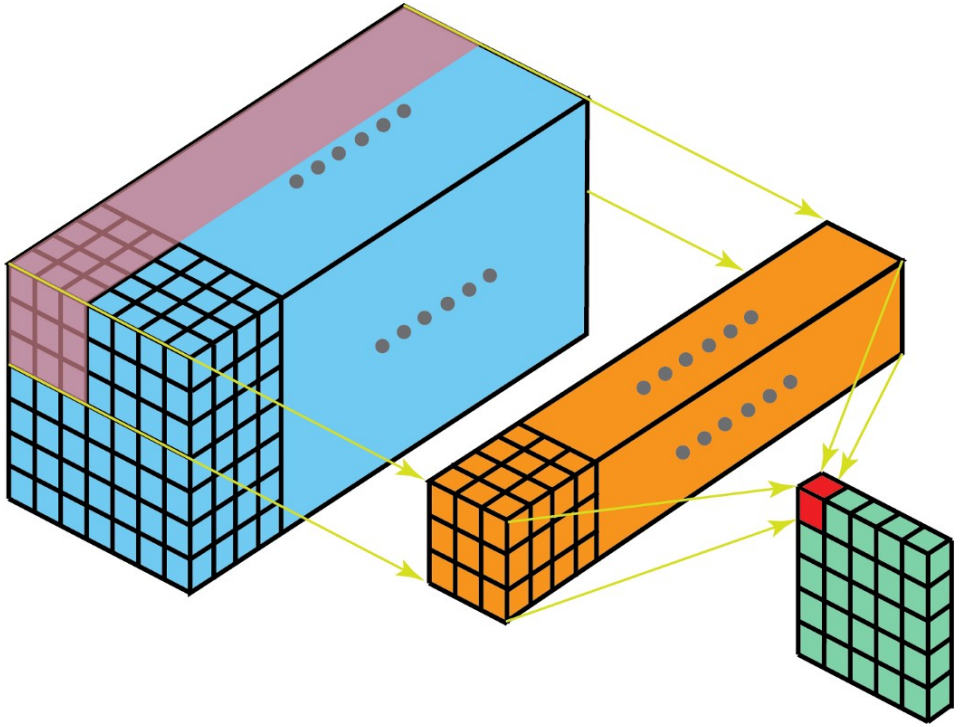


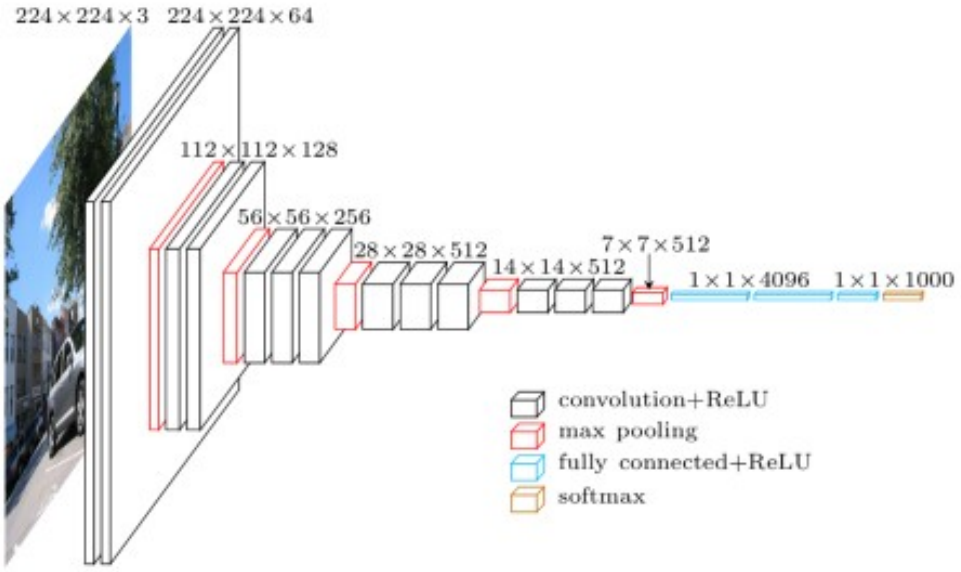
- Performance increases with depth (*L*) and growth rate (*k*).
- Dense Nets are more parameter efficient than previous models.
- Performance improves with bottleneck (B) and (C=0.5) feature map compression factor.

Huang et al. Densely Connected Convolutional Networks. CVPR 2017

Efficient architectures

- Depthwise Separable Convolution (DwSC). Introduction
  - The use of a small receptive field significantly reduces the number of parameters of the first convolutional layers.
  - Layers very deep in the model have small spatial and large depth dimensions.



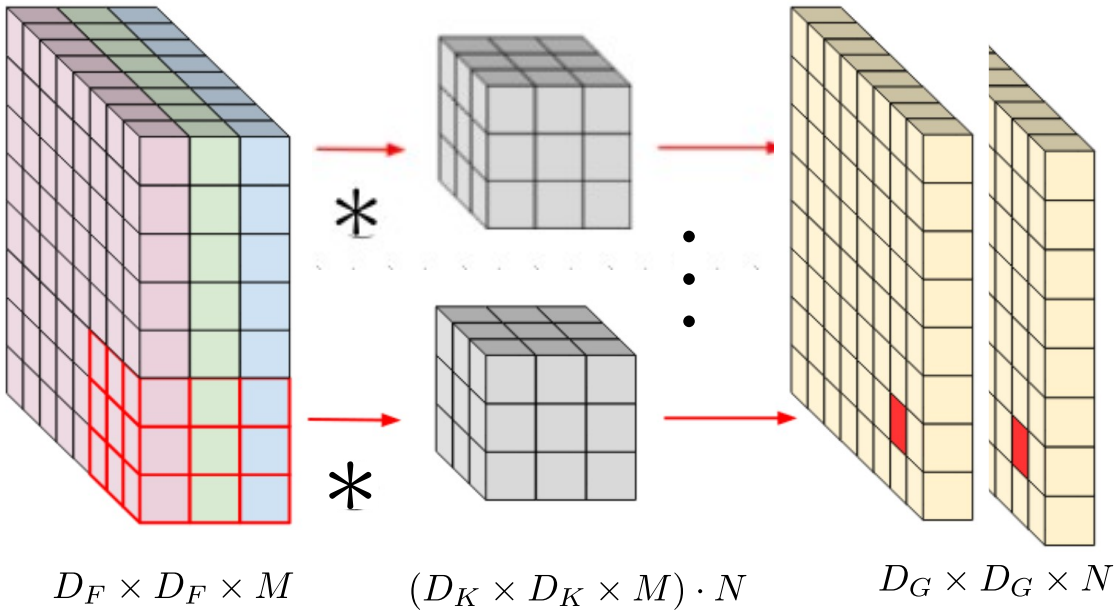


Convolutional layers are less efficient in very deep architectures.

Efficient architectures

- Depthwise Separable Convolution (DwSC)

  In a **standard convolutional** layer



Number of multiplications: $D_K^2 \times M \times D_G^2 \times N$
Number of parameters: $D_K^2 \times M \times N$

We filter the input layer N times, each filter involves $D_K^2 \times M \times D_G^2$ multiplications.

Efficient architectures

- Depthwise Separable Convolution (DwSC)

  Idea. Decouple the number and depth of kernels respectively from the number of output and input feature maps.

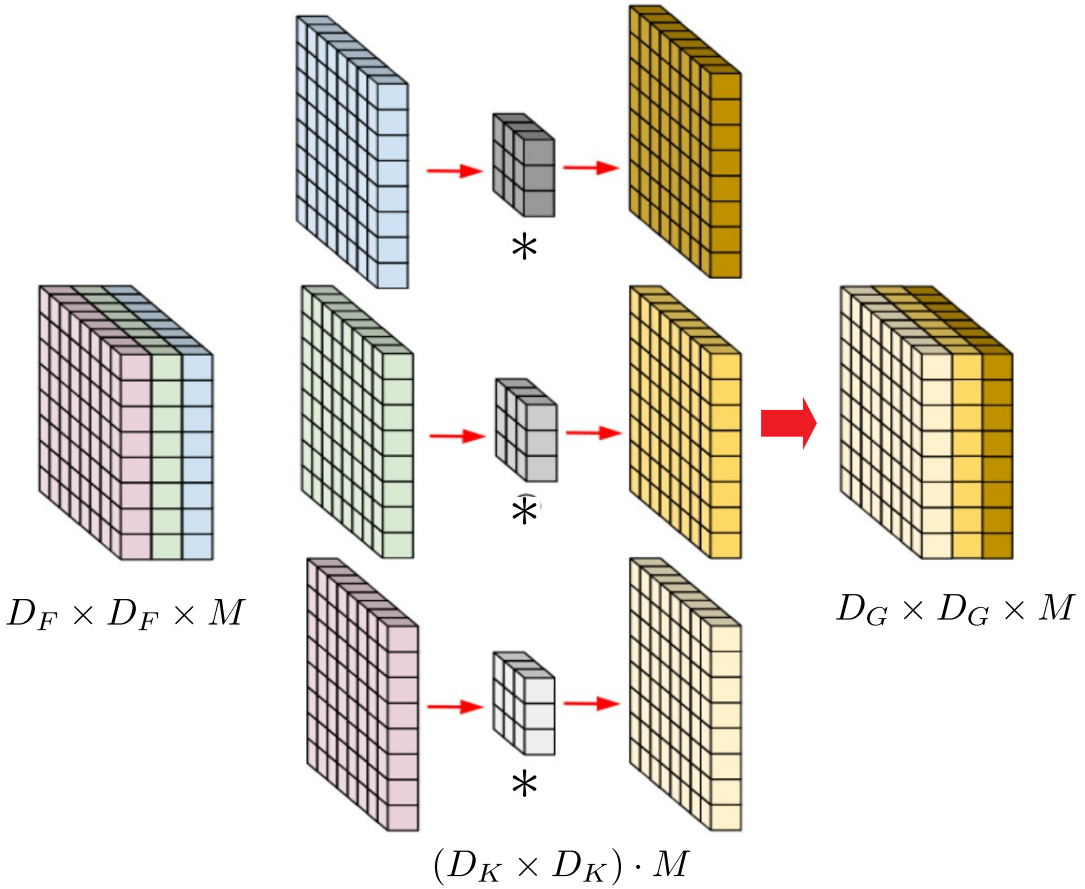
  In this way we will reduce the number of parameters and operations.

  Compute convolution in two steps:
  1. Depthwise convolution
     Introduce one $D_K \times D_K \times 1$ filter per input feature map, M filters.
  2. Pointwise convolution
     Introduce one $1 \times 1 \times M$ filter per output feature map, N filters.

Efficient architectures

- Depthwise Separable Convolution (DwSC)

  1. Depthwise convolution (DwC)

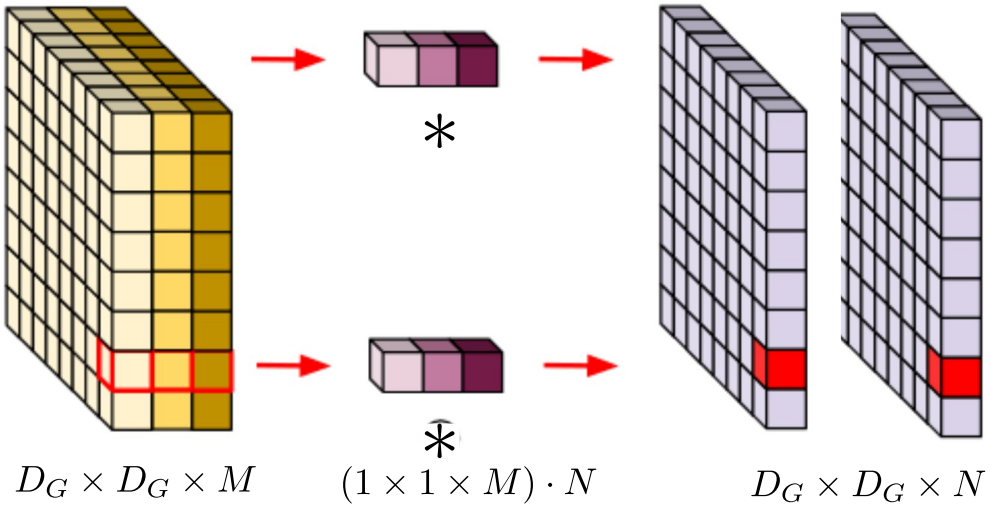


Number of multiplications: $D_K^2 \times M \times D_G^2$
Number of parameters: $D_K^2 \times M$

Efficient architectures

- Depthwise Separable Convolution (DwSC)

  2. Pointwise convolution (PwC)



Number of multiplications: $M \times D_G^2 \times N$
Number of parameters: $M \times N$

Efficient architectures

- Depthwise Separable Convolution (DwSC)

  Total number of multiplications: DwC + PwC

  $$D_K^2 \times M \times D_G^2 + M \times D_G^2 \times N = D_G^2 \times M \times (D_K^2 + N)$$

  Total number of parameters: DwC + PwC

  $$D_K^2 \times M + M \times N = M \times (D_K^2 + N)$$

- Depthwise Separable vs. Standard Convolution

  Number of multiplications:

  $$\frac{\text{Dw Sep Conv}}{\text{Stand Conv}} = \frac{D_G^2 \times M \times (D_K^2 + N)}{D_K^2 \times M \times D_G^2 \times N} = \frac{1}{N} + \frac{1}{D_K^2} \approx 0.1$$

  $$N = 1024, \quad D_K = 3$$

  $$\frac{\text{Dw Sep Conv}}{StandConv} = \frac{M \times (D_K^2 + N)}{D_K^2 \times M \times N} = \frac{1}{N} + \frac{1}{D_K^2} \approx 0.1$$

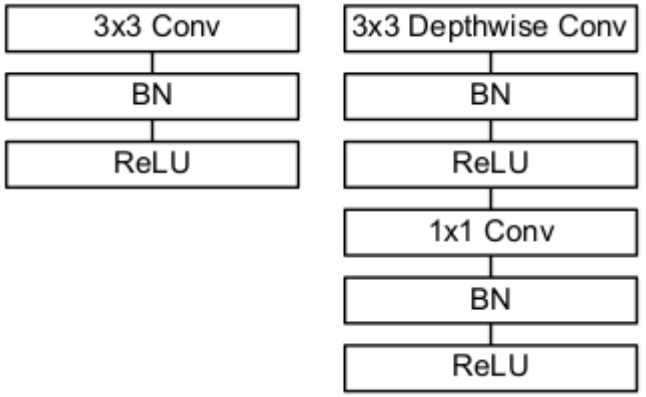
Efficient architectures

- Mobile Net architecture

  28 layers, VGG-like, extensive use of DwSC

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$



Stand conv DwS Conv

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

DwS vs. Stand Conv net Image Net results

Howard et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. ArXiv. 2017

Efficient architectures

- Mobile Net architecture

Simple computational complexity control with parameters:
- α base network width (# of channels)
- ρ base network resolution (input *w* x *h*)

the computational cost is

$$D_K^2 \times \alpha M \times \rho^2 D_G^2 + \rho^2 D_G^2 \times \alpha M \times \alpha N \sim o(\alpha^2 \times \rho^2)$$

MobileNet width mutiplier

Width Multiplier	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
0.75 MobileNet-224	68.4%	325	2.6
0.5 MobileNet-224	63.7%	149	1.3
0.25 MobileNet-224	50.6%	41	0.5

MobileNet resolution mutiplier

Resolution	ImageNet Accuracy	Million Mult-Adds	Million Parameters
1.0 MobileNet-224	70.6%	569	4.2
1.0 MobileNet-192	69.1%	418	4.2
1.0 MobileNet-160	67.2%	290	4.2
1.0 MobileNet-128	64.4%	186	4.2

Howard et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. ArXiv. 2017

Efficient architectures

- Mobile Net V.2 architecture (ResNet-like)

  Introduces **inverted residual** with **linear bottleneck** layer

  - **Linear bottleneck**

    The information in each feature map lies in a low-dimensional manifold, so it may be compressed.

    We can reduce network width (# feature maps) introducing linear bottleneck layers in the convolutional blocks.

Sandler et al. MobileNet V2: Inverted Residuals and Linear Bottlenecks. CVPR 2018

Efficient architectures

- Mobile Net V.2 architecture (ResNet-like)

  Introduces **inverted residual** with linear **bottleneck** layer
  - **Linear bottleneck**
  - **Inverted residual**

  The memory in a net with residual layers is the maximum total size of combined inputs/outputs across all operations.

$$M(G) = \max_{op \in G} \left[\sum_{A \in \text{op}_{inp}} |A| + \sum_{B \in \text{op}_{out}} |B| + |op| \right]$$

  If a residual block is a single operation (and the inner convolution a disposable tensor), then the amount of memory is dominated by the size of input/output bottleneck tensors**.**

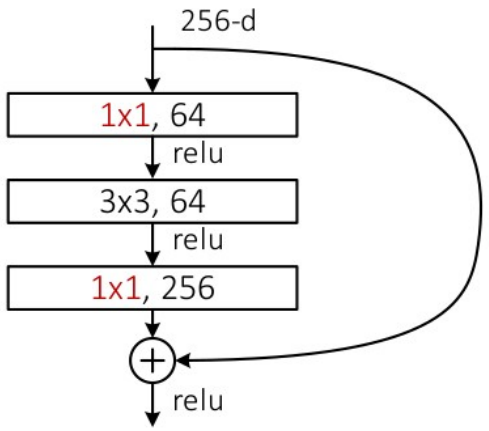
  A small i/o tensor has a memory efficient implementation.

Sandler et al. MobileNet V2: Inverted Residuals and Linear Bottlenecks. CVPR 2018

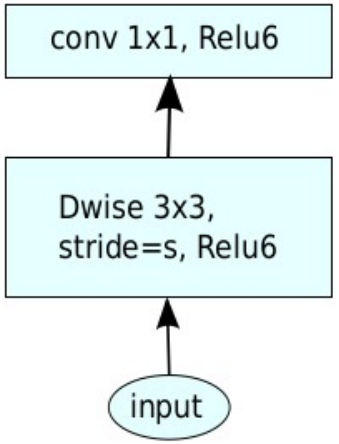
Efficient architectures

- Mobile Net V.2 architecture (ResNet-like)

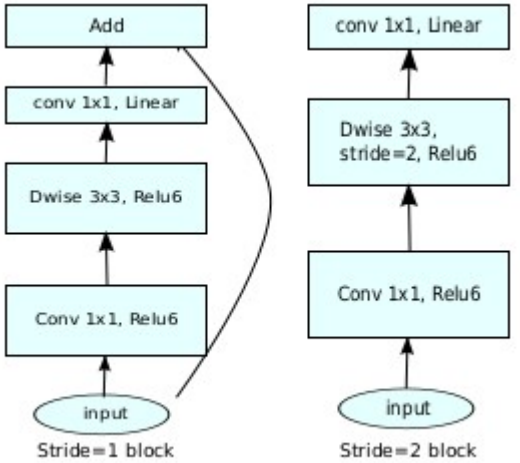
  Inverted residuals and **linear bottlenecks**



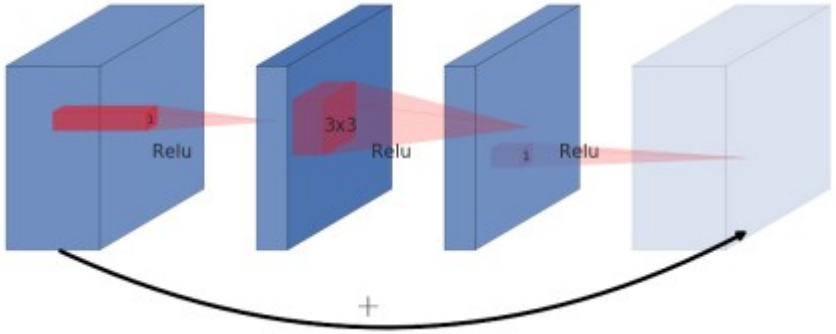
ResNet bottleneck



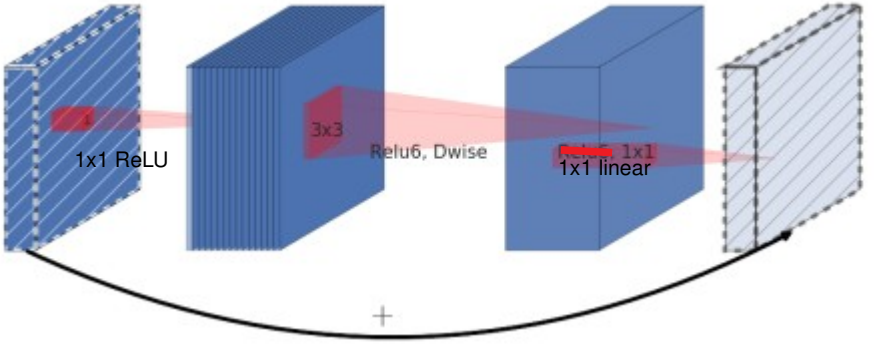
Mobile Net



Mobile Net V.2



Residual block

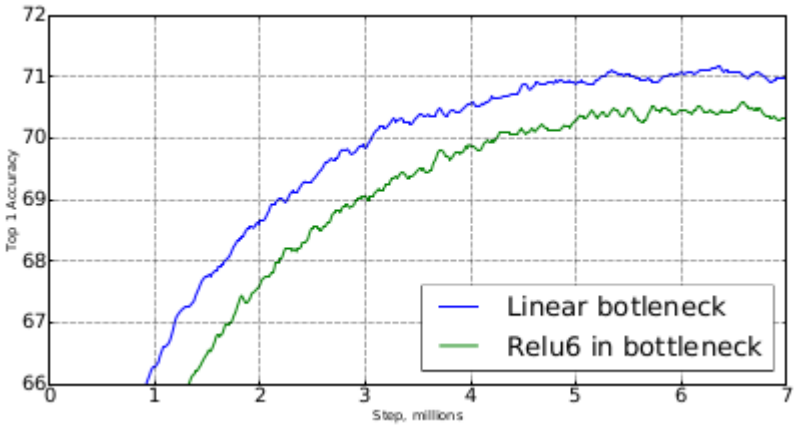


Inverted residual block

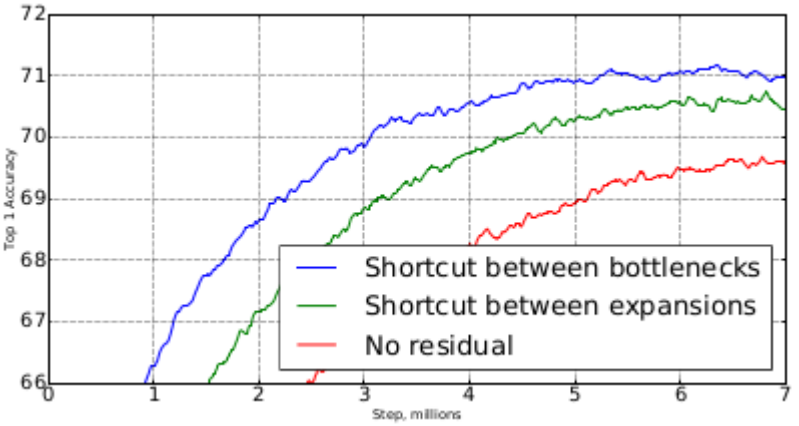
Sandler et al. MobileNet V2: Inverted Residuals and Linear Bottlenecks. CVPR 2018

Efficient architectures

- Mobile Net V.2 architecture. Results



Impact of non-linearity in blottleneck



Impact of various residual blocks

Network	Top 1	Params	MAdds	CPU
MobileNetV1	70.6	4.2M	575M	113ms
MobileNetV2	**72.0**	**3.4M**	**300M**	**75ms**
MobileNetV2 (1.4)	**74.7**	6.9M	585M	**143ms**

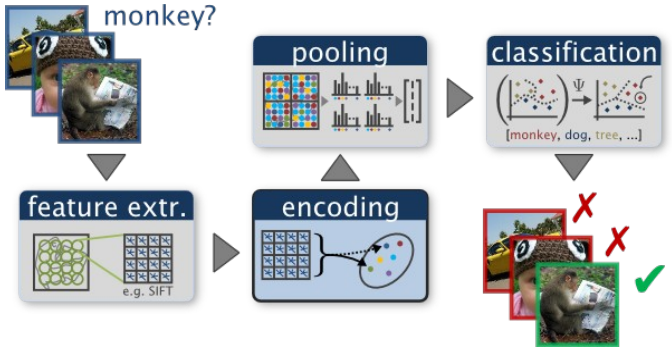
Performance on ImageNet

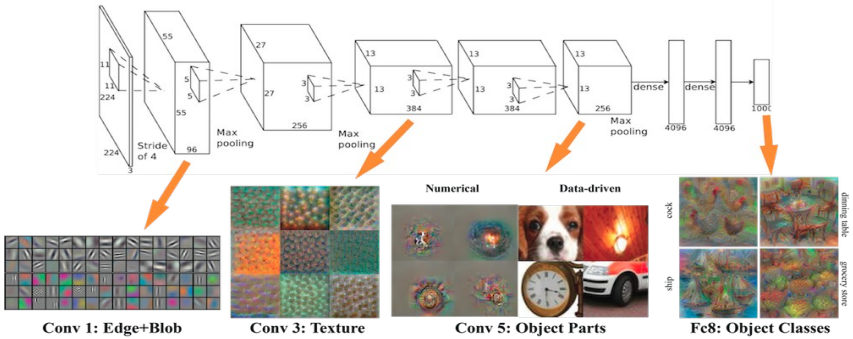
Sandler et al. MobileNet V2: Inverted Residuals and Linear Bottlenecks. CVPR 2018

Efficient architectures

- Network architecture search (NAS). Introduction

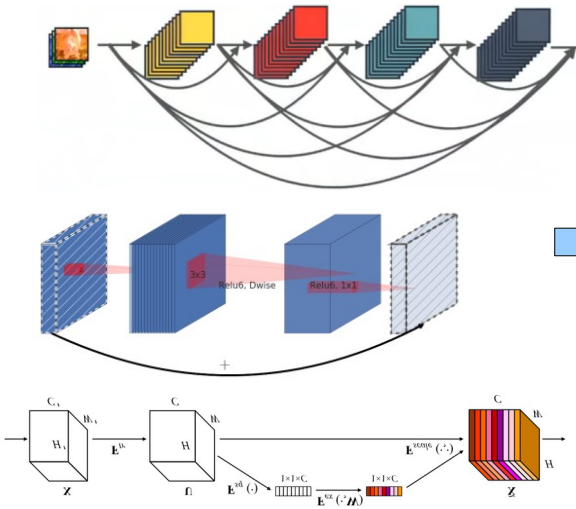
  From **feature engineering** to **architecture engineering**.
  - Deep models automate feature search, …

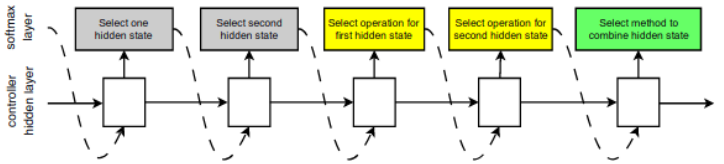


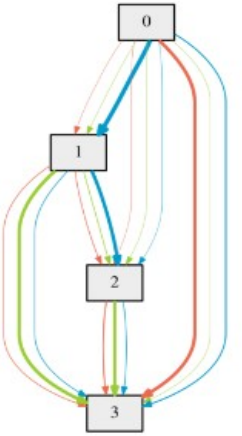


but require careful architecture optimization for top performance.

- **Network architecture search** automates architecture design

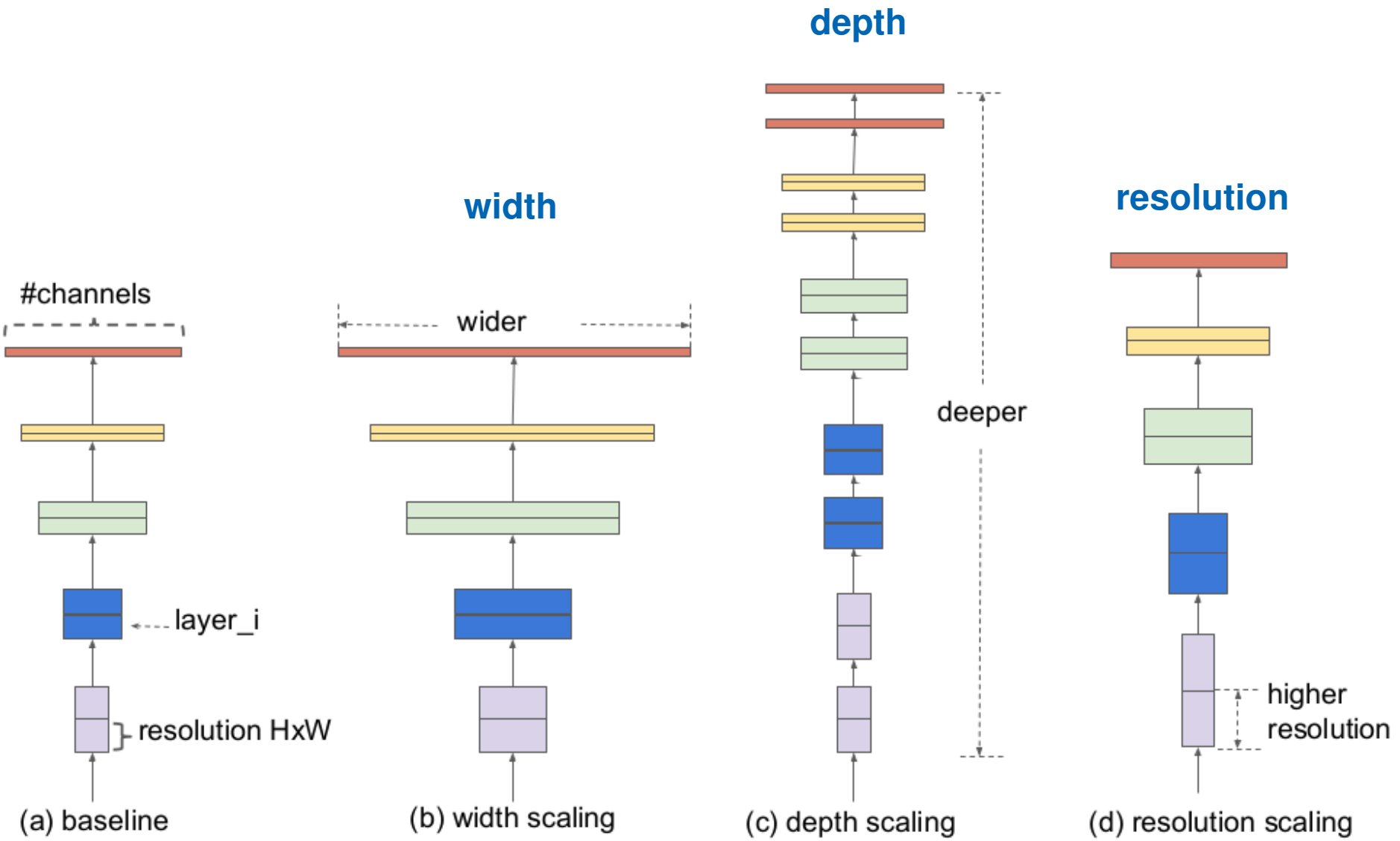






Efficient architectures

- **Efficient Net** architecture. Introduction.
  Network accuracy/efficiency usually considered one-dimensionaly

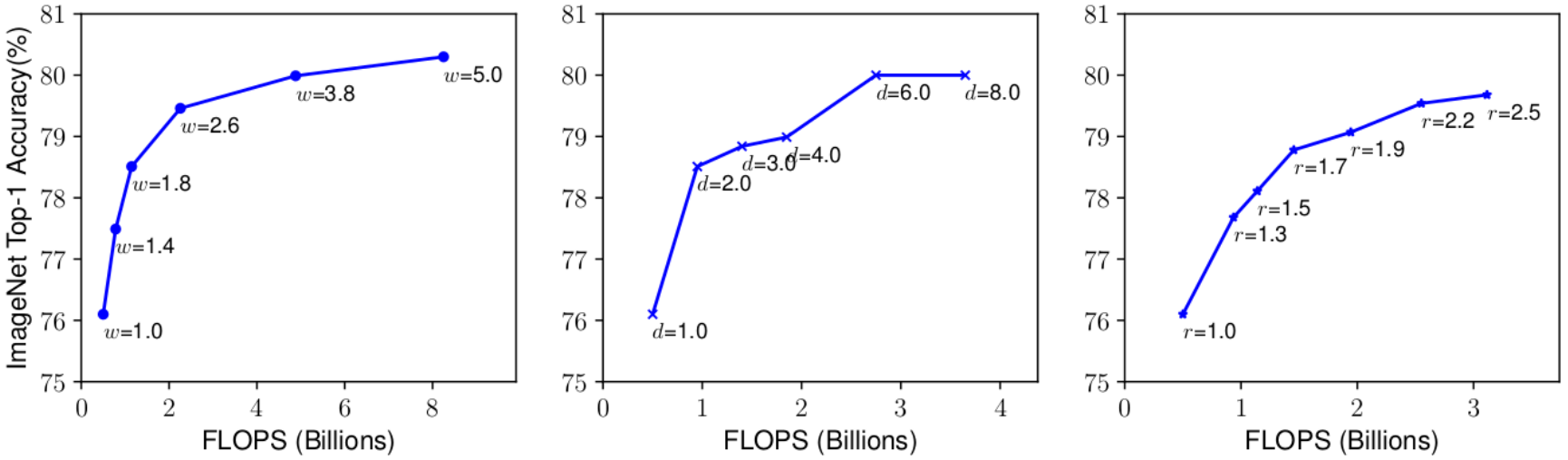


Tan. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML 2019

Efficient architectures

- **Efficient Net** architecture. Introduction.
  Network accuracy/efficiency usually considered one-dimensionaly:
  - Depth. VGG, GoogleLeNet (inception), ResNet, ...
  - Width. Mobile Net, ...
  - Resolution. Mobile Net, ...

  However all have dimishing returns.

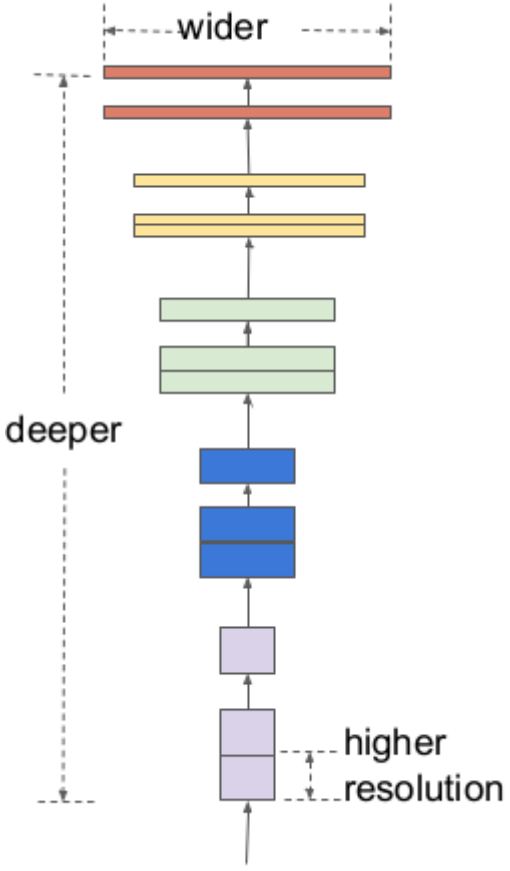


To achieve top accuracy and efficiency all dimensions must be optimized simultaneously.

Tan. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML 2019

Efficient architectures

- **Efficient Net** architecture. Contribution
  Scaling method that resizes a baseline convnet to any target resource constraints.



A net unit *i* may be defined as

$$Y_i = \mathcal{F}_i(X_i)$$

A conv net $\mathcal{N}$ can be represented by a list of composed units

$$\mathcal{N} = \mathcal{F}_k \odot ... \odot \mathcal{F}_1 \odot \mathcal{F}_1(X_1) = \bigodot_{j=1...k} \mathcal{F}_j(X_1)$$

The goal is to maximize the model accuracy

$$\max_{d,w,r} \quad Accuracy\big(\mathcal{N}(d,w,r)\big)$$

$$s.t. \quad \mathcal{N}(d,w,r) = \bigodot_{i=1...s} \hat{\mathcal{F}}_i^{d\cdot\hat{L}_i}\left(X_{\langle r\cdot\hat{H}_i, r\cdot\hat{W}_i, w\cdot\hat{C}_i\rangle}\right)$$

$$\text{Memory}(\mathcal{N}) \leq \text{target\_memory}$$

$$\text{FLOPS}(\mathcal{N}) \leq \text{target\_flops}$$

Tan. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML 2019

Efficient architectures

- **Efficient Net** architecture. Contribution
  Scaling method that resizes a baseline convnet to any target resource constraints.

  The compound coefficient ϕ uniformly scales network width, depth, and resolution in a principled way:

$$\begin{aligned} \text{depth: } d &= \alpha^\phi \\ \text{width: } w &= \beta^\phi \\ \text{resolution: } r &= \gamma^\phi \end{aligned} \qquad \begin{aligned} \text{s.t. } \alpha \cdot \beta^2 \cdot \gamma^2 &\approx 2 \\ \alpha \geq 1, \beta &\geq 1, \gamma \geq 1 \end{aligned}$$

  ϕ: coefficient controling available resources
  α, β, γ: empirical constants to distribute resources

  So, scaling a ConvNet increases the computational cost by

$$(\alpha \cdot \beta^2 \cdot \gamma^2)^\phi$$

  They are constrained such that for any ϕ, the cost will increase by 2^ϕ.

Tan. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML 2019

Efficient architectures

- **Efficient Net** architecture. Procedure.
- STEP 0.
  Establish baseline architecture **EfficientNet-B0** using multi-objective NAS to optimize accuracy and FLOPS .

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	28×28	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

MBConv. Mobile Inverted Bottleneck with SE.

Tan. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML 2019

Efficient architectures

- **Efficient Net** architecture. Procedure.

- STEP 0.
  Establish baseline architecture EfficientNet-B0.

- STEP 1.
  Fix $\phi = 1$ perform grid search to optimize the acuracy of EfficientNet-B0, s.t. $\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2$. Resulting:

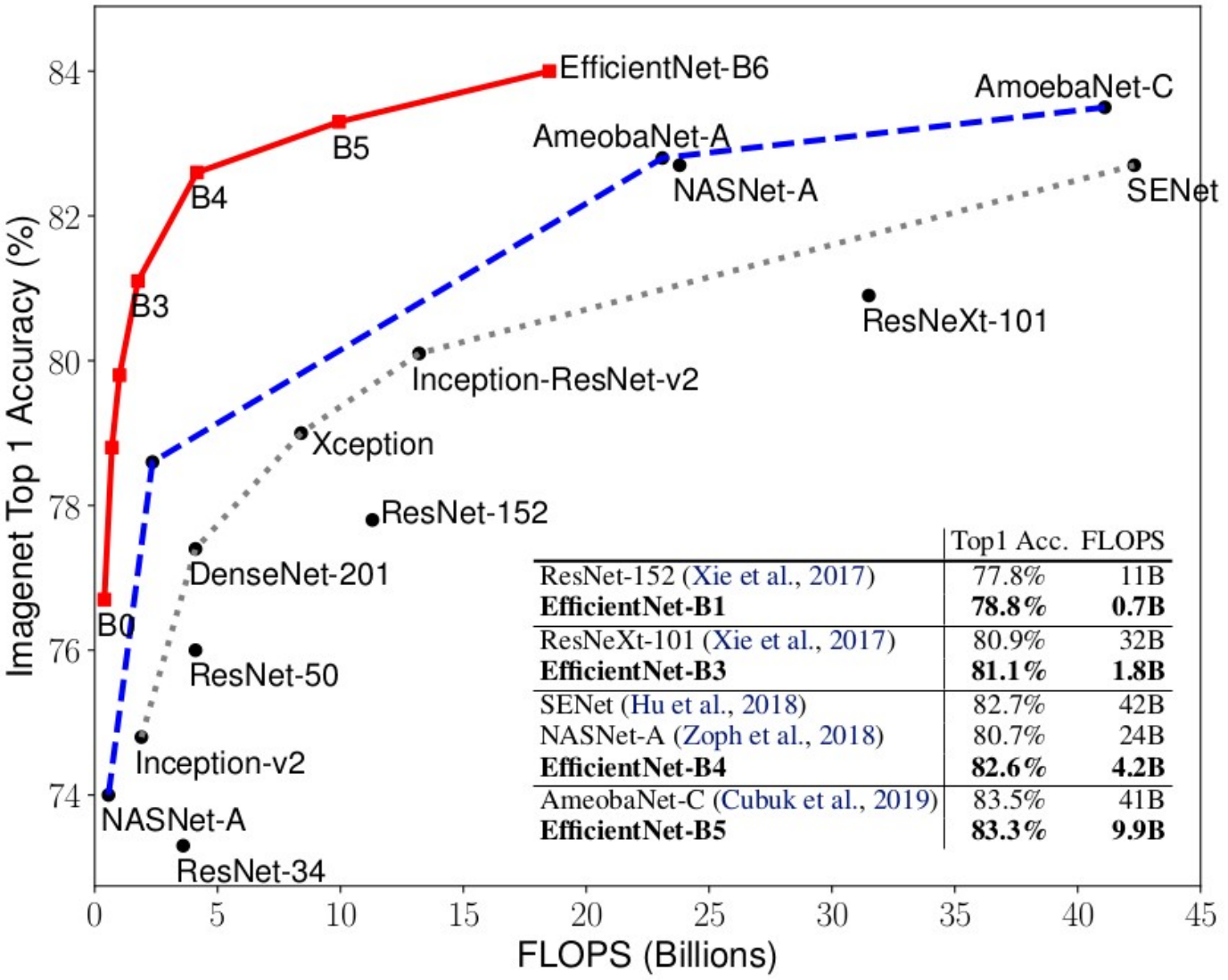
$$\alpha = 1.2, \beta = 1.1, \gamma = 1.15$$

- STEP 2.
  Fix α, β, γ and scale-up the baseline net with different values of ϕ to obtain EfficientNet-B1 to B7.

Tan. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML 2019

Efficient architectures

- **Efficient Net** architecture. Results.



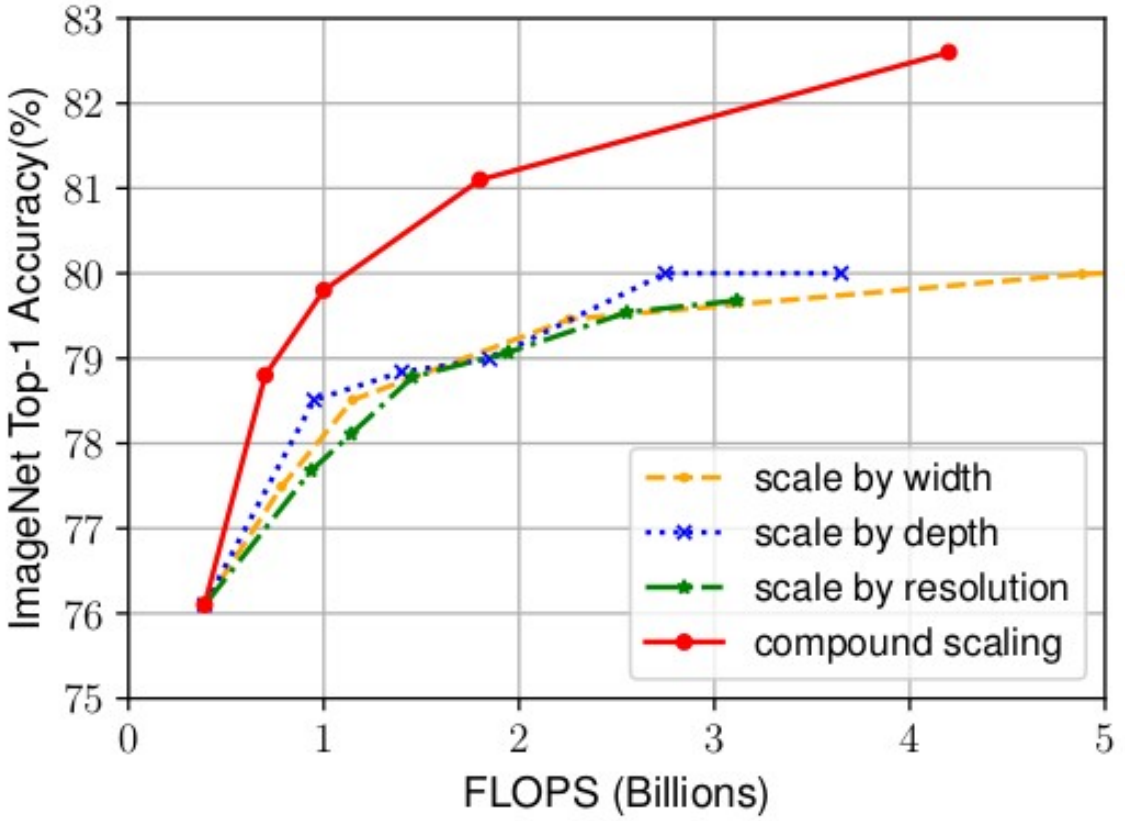
	Top1 Acc.	FLOPS
ResNet-152 (Xie et al., 2017)	77.8%	11B
EfficientNet-B1	**78.8%**	**0.7B**
ResNeXt-101 (Xie et al., 2017)	80.9%	32B
EfficientNet-B3	**81.1%**	**1.8B**
SENet (Hu et al., 2018)	82.7%	42B
NASNet-A (Zoph et al., 2018)	80.7%	24B
EfficientNet-B4	**82.6%**	**4.2B**
AmeobaNet-C (Cubuk et al., 2019)	83.5%	41B
EfficientNet-B5	**83.3%**	**9.9B**

Tan. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML 2019

Efficient architectures

- **Efficient Net** architecture. Results.

One vs multi-dimensional network scaling

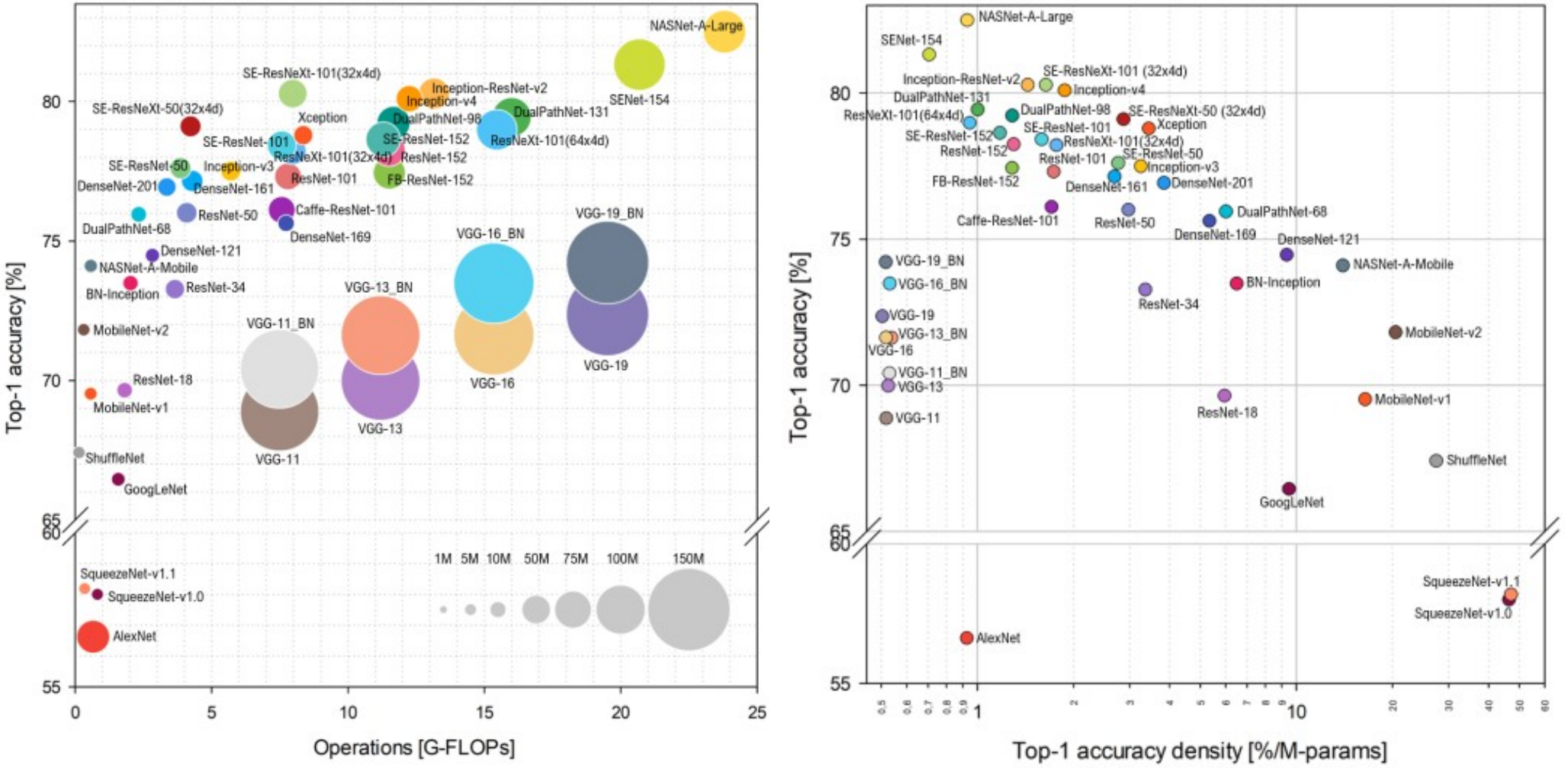


compound scaling improves accuracy by up to 2.5%

Tan. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. ICML 2019

Efficient architectures

- Discussion



Top-1 accuracy vs. Operations and accuracy density.
The accuracy density measures how efficiently each model uses its parameters.

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

CNNs architectures

- Conclusion.
  - CNN architectures are the backbone of most deep learning computer vision applications.
  - They provide the invariant features required for succesfully solving many higher-level computer vision problems:
    - Object detection
    - Image segmentation
    - Tracking
    - …..
  - The key to success is selecting the best CNN architecture