165B
Machine Learning
ResNet

Lei Li (leili@cs)
UCSB

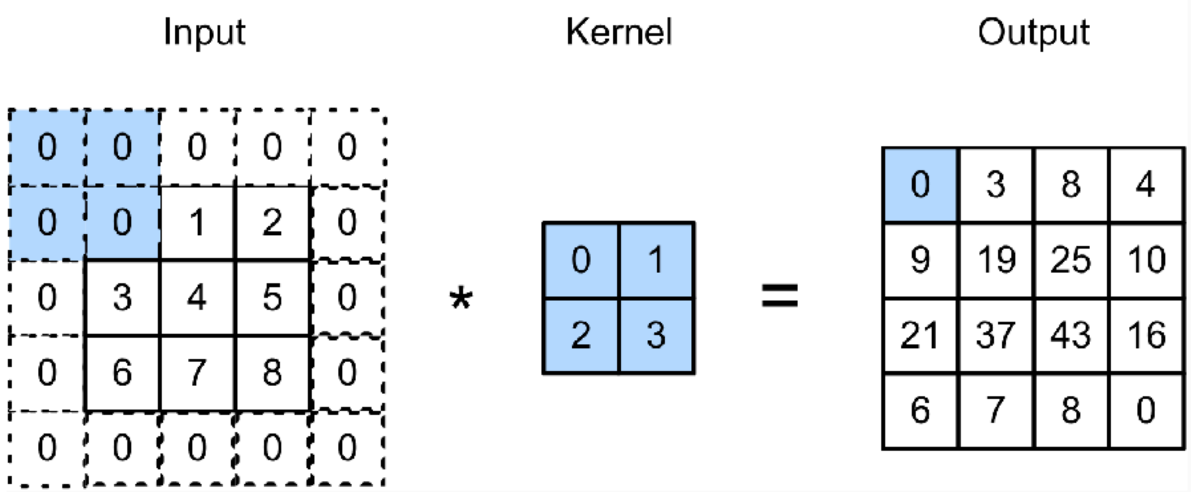
Acknowledgement: Slides borrowed from Bhiksha Raj's 11485 and Mu Li & Alex Smola's 157 courses on Deep Learning, with modification

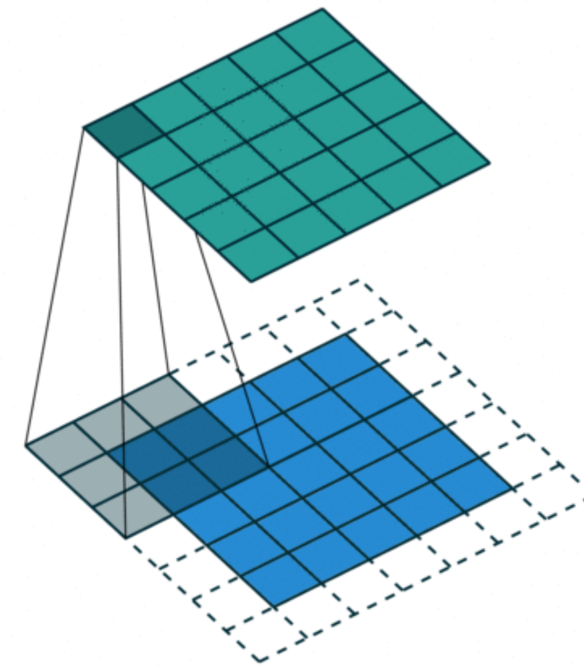
Recap

- Convolutional layer
  - Reduced model capacity compared to dense layer
  - Efficient at detecting spatial pattens
  - High computation complexity
  - Control output shape via padding, strides and channels
- Max/Average Pooling layer
  - Provides some degree of invariance to translation

2-D Convolution Layer

$$y_{i,j} = \sum_{a=1}^{h} \sum_{b=1}^{w} w_{a,b} x_{i+a,j+b}$$





$$0 \times 0 + 0 \times 1 + 0 \times 2 + 0 \times 3 = 0$$

2-D Convolution Layer Summary

- Input $\mathbf{X} : c_i \times n_h \times n_w$
- Kernel $\mathbf{W} : c_o \times c_i \times k_h \times k_w$
- Bias $\mathbf{B} : c_o$
- Output $\mathbf{Y} : c_o \times m_h \times m_w$
- Complexity (number of floating point operations FLOP)

$$\mathbf{Y} = \mathbf{X} \star \mathbf{W} + \mathbf{B}$$

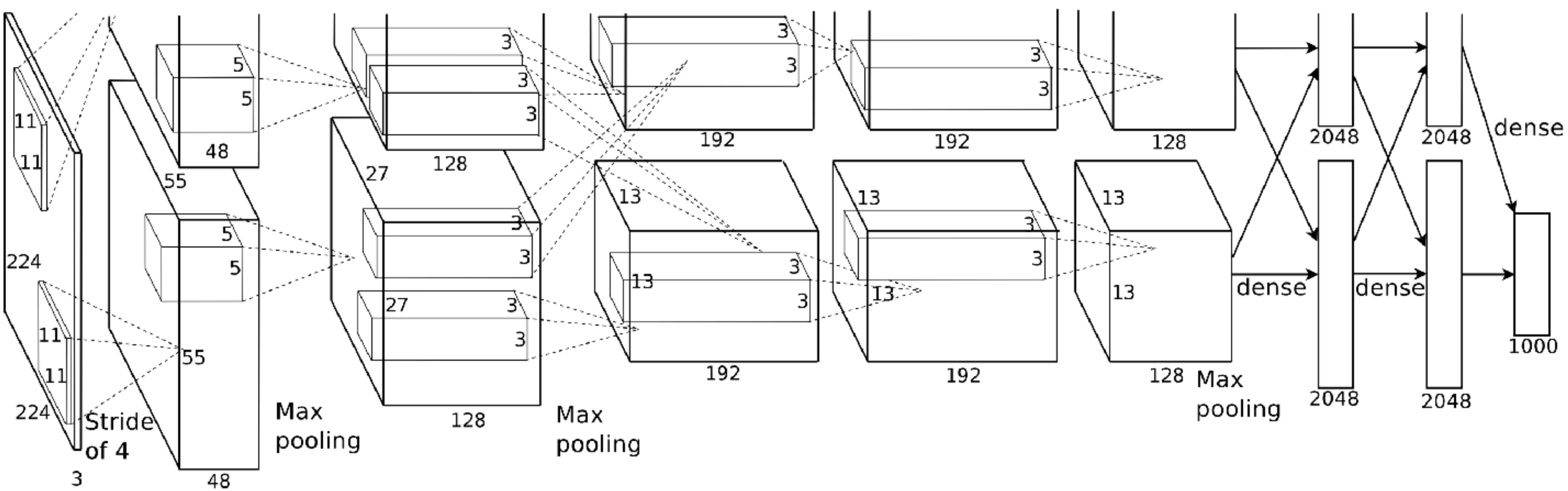
$$c_i = c_o = 100$$
$$k_h = h_w = 5$$
$$m_h = m_w = 64$$

$$O(c_i c_o k_h k_w m_h m_w)$$

1GFLOP

- 10 layers, 1M examples: 10PF
(CPU: 0.15 TF = 18h, GPU: 12 TF = 14min)

AlexNet



224
224
11
11
11
11
Stride of 4
3
55
55
5
5
5
5
48
48
Max pooling
27
27
3
3
3
3
3
3
3
3
128
128
Max pooling
13
13
3
3
3
3
192
192
13
13
3
3
3
3
192
192
13
13
128
128
Max pooling
dense
dense
dense
2048
2048
2048
2048
1000

SVM

- In the 1990s, algorithms based on support vector machines (SVM) are developed
- Kernel methods
- There are (shallow) models
- Linear classifier with margin loss (hinge loss)

Vladimir **V**apnik

Computer Vision Pre-2012

- Extract features
- Describe geometry (e.g. multiple cameras) analytically
- **(Non)Convex** optimization problems
- Many beautiful theorems …
- Works very well in theory when the assumptions are satisfied

Feature Engineering

- Feature engineering is crucial
- Feature descriptors, e.g. SIFT (Scale-invariant feature transform), SURF
- Bag of visual words (clustering)
- Then apply SVM …

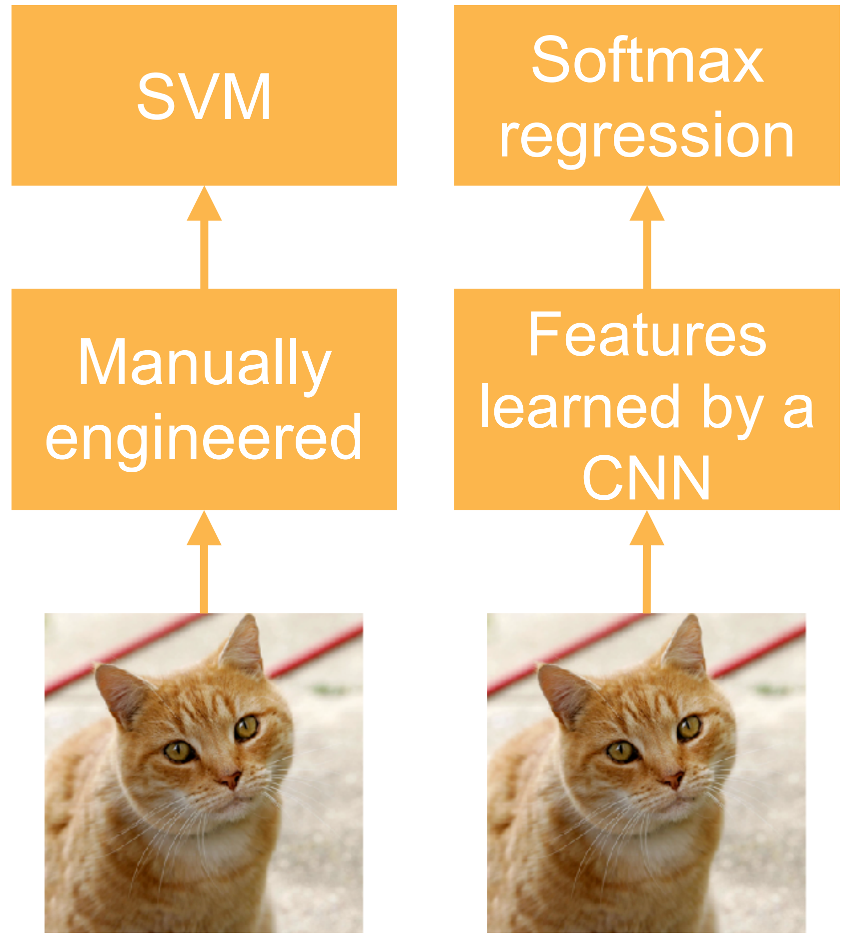
(opencv)

ImageNet (2010)

Images	Color images with nature objects	Gray image for hand-written digits
Size	469 x 387	28 x 28
# examples	1.2 M	60 K
# classes	1,000	10

AlexNet

- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet
- Key modifications
  - Dropout (regularization)
  - ReLu (training)
  - MaxPooling
- Paradigm shift for computer vision

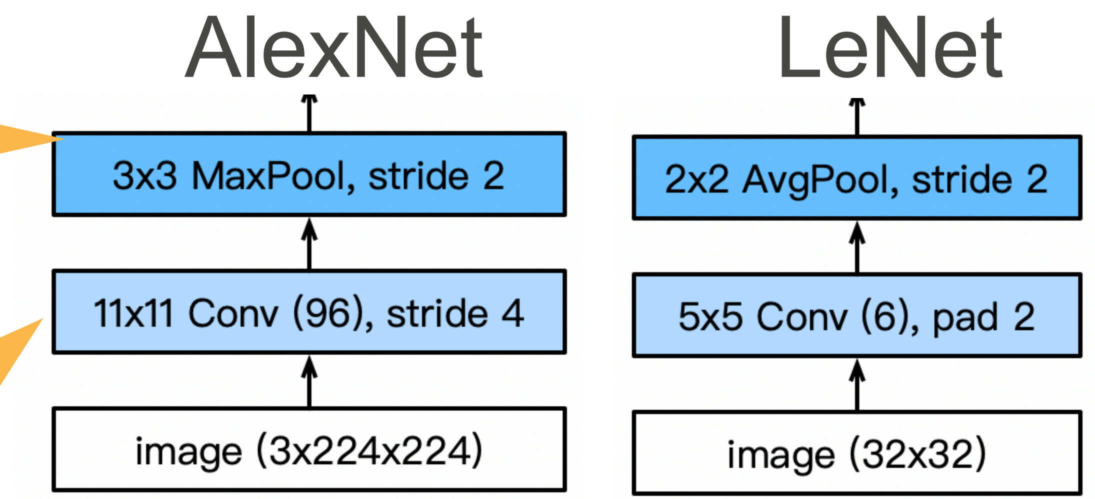


Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. 2012

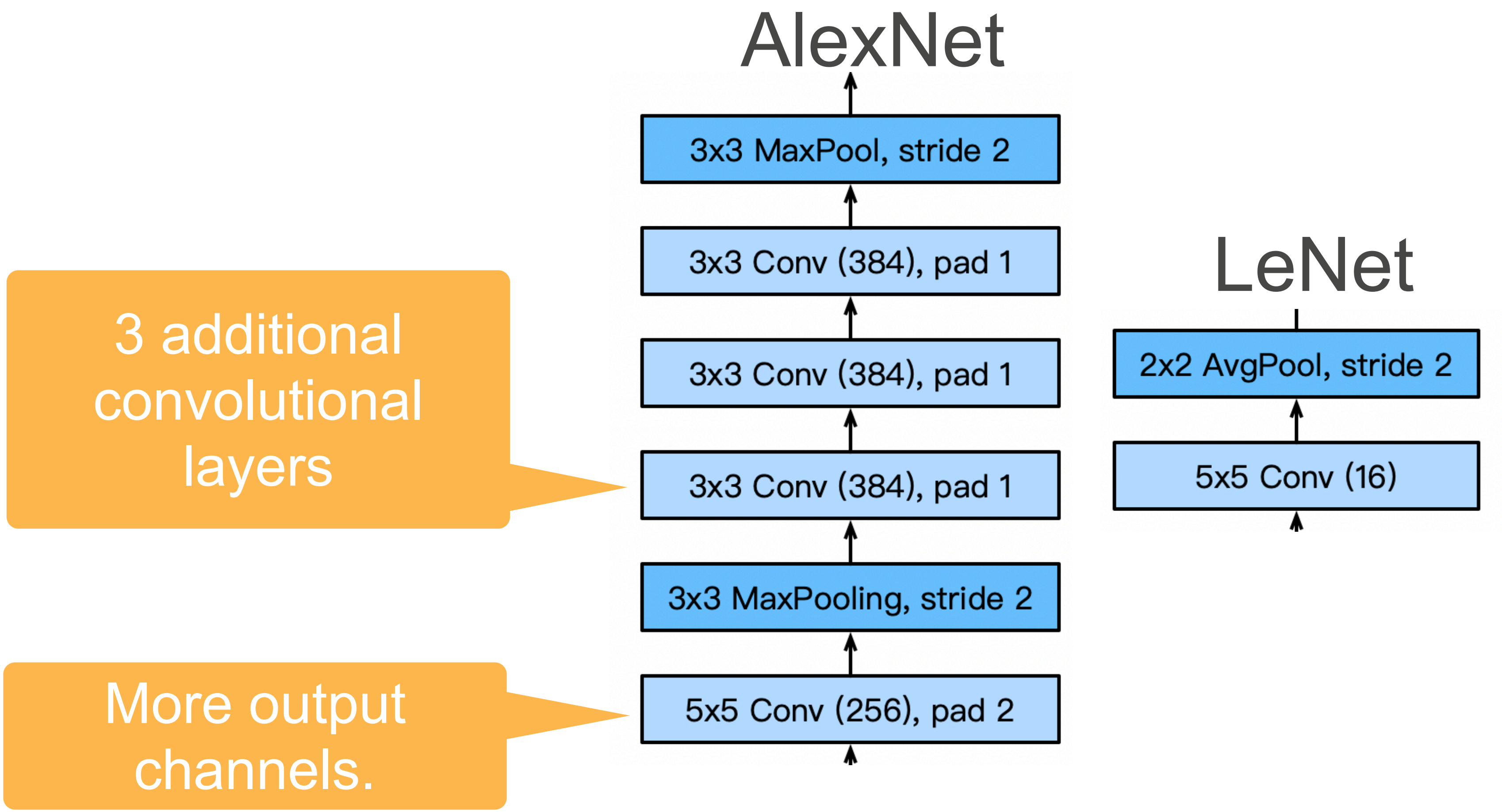
AlexNet Architecture

Larger pool size, change to max pooling

Larger kernel size, stride because of the increased image size, and more output channels.



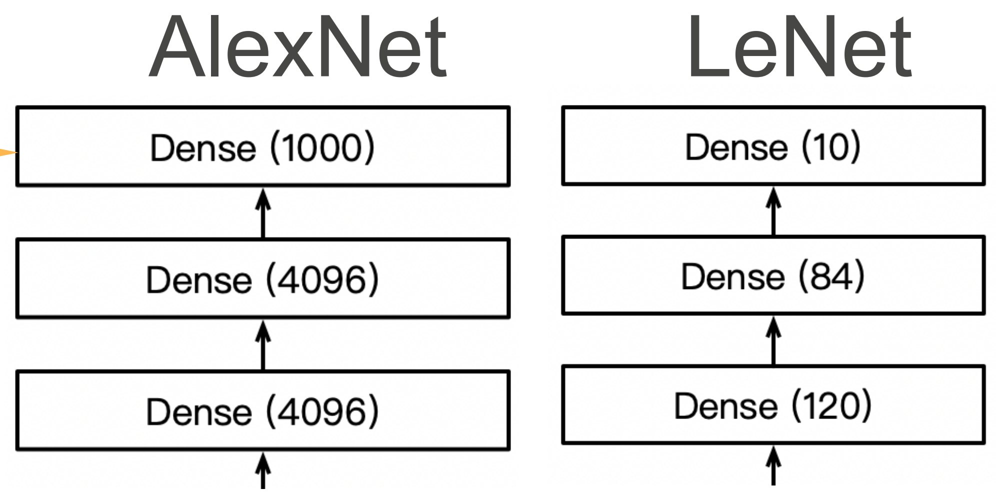
AlexNet Architecture



AlexNet Architecture

1000 classes output

Increase hidden size from 120 to 4096



More Tricks

- Change activation function from sigmoid to ReLu (no more vanishing gradient)
- Add a dropout layer after two hidden FFN layers (better robustness / regularization)
- Data augmentation

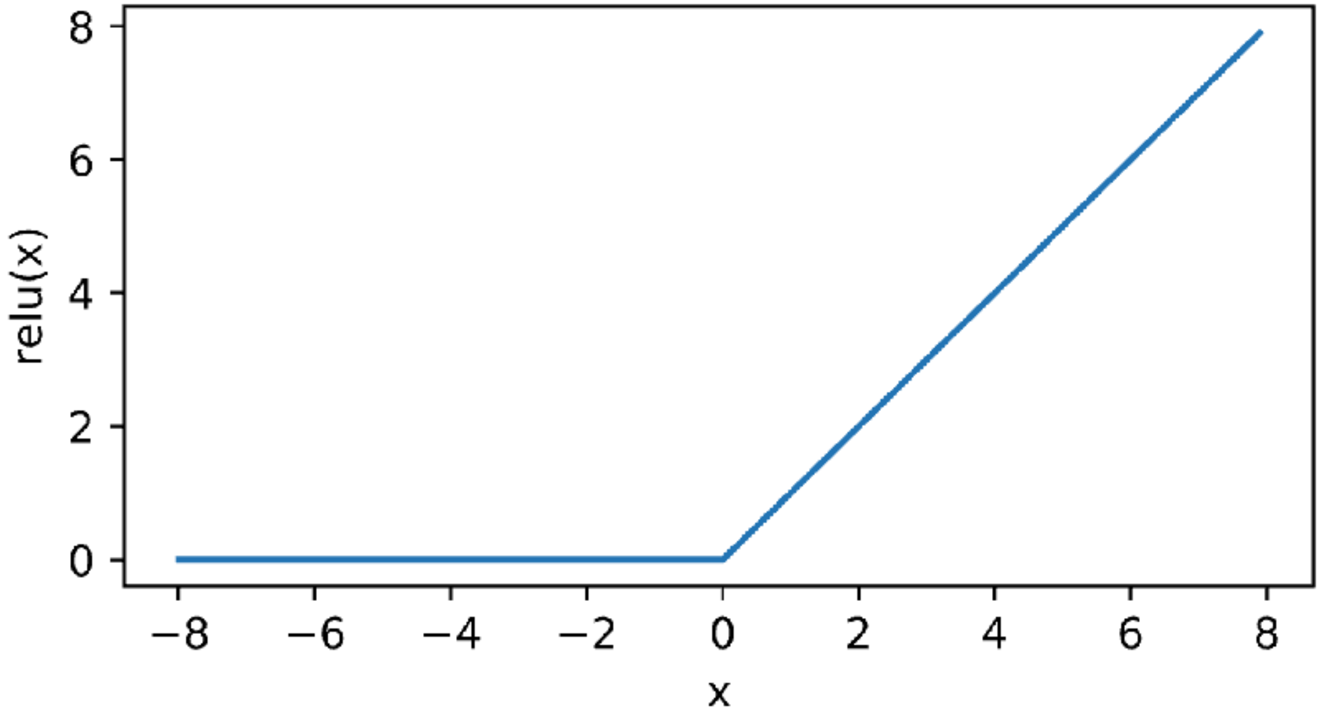
Data Augmentation

- Create additional training data with existing data

ReLU Activation

ReLU: rectified linear unit

$$\mathrm{ReLU}(x) = \max(x, 0)$$

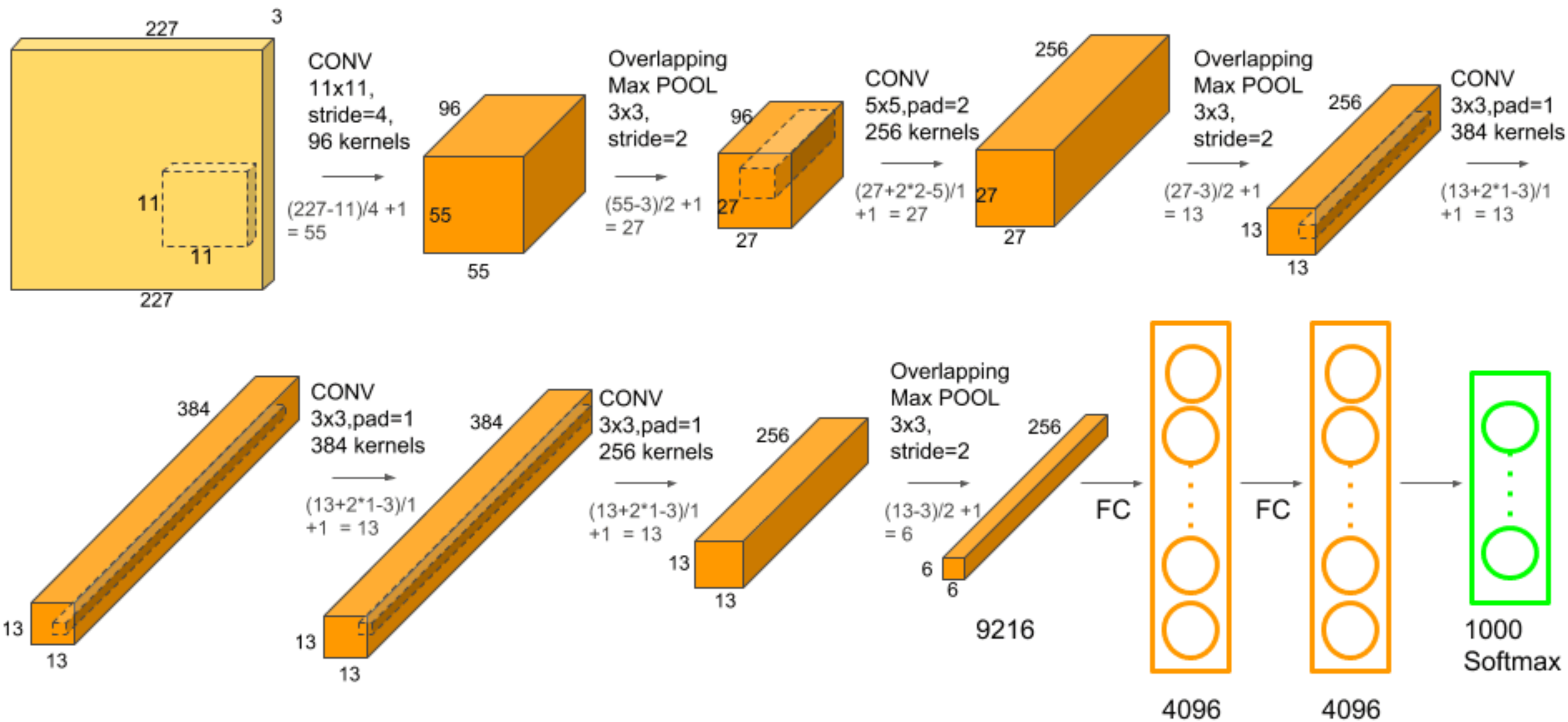


Dropout Layer

- For every input x_i, Dropout produces

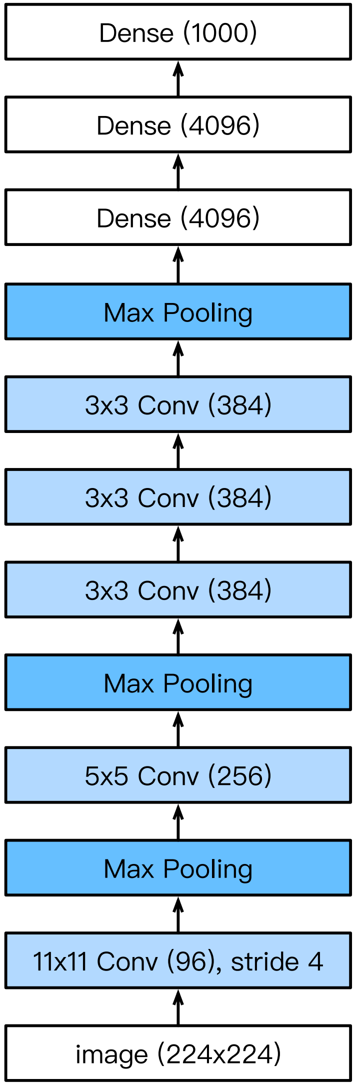
$$x_i' = \begin{cases} 0 & \text{with probablity } p \\ \dfrac{x_i}{1-p} & \text{otherise} \end{cases}$$

AlexNet

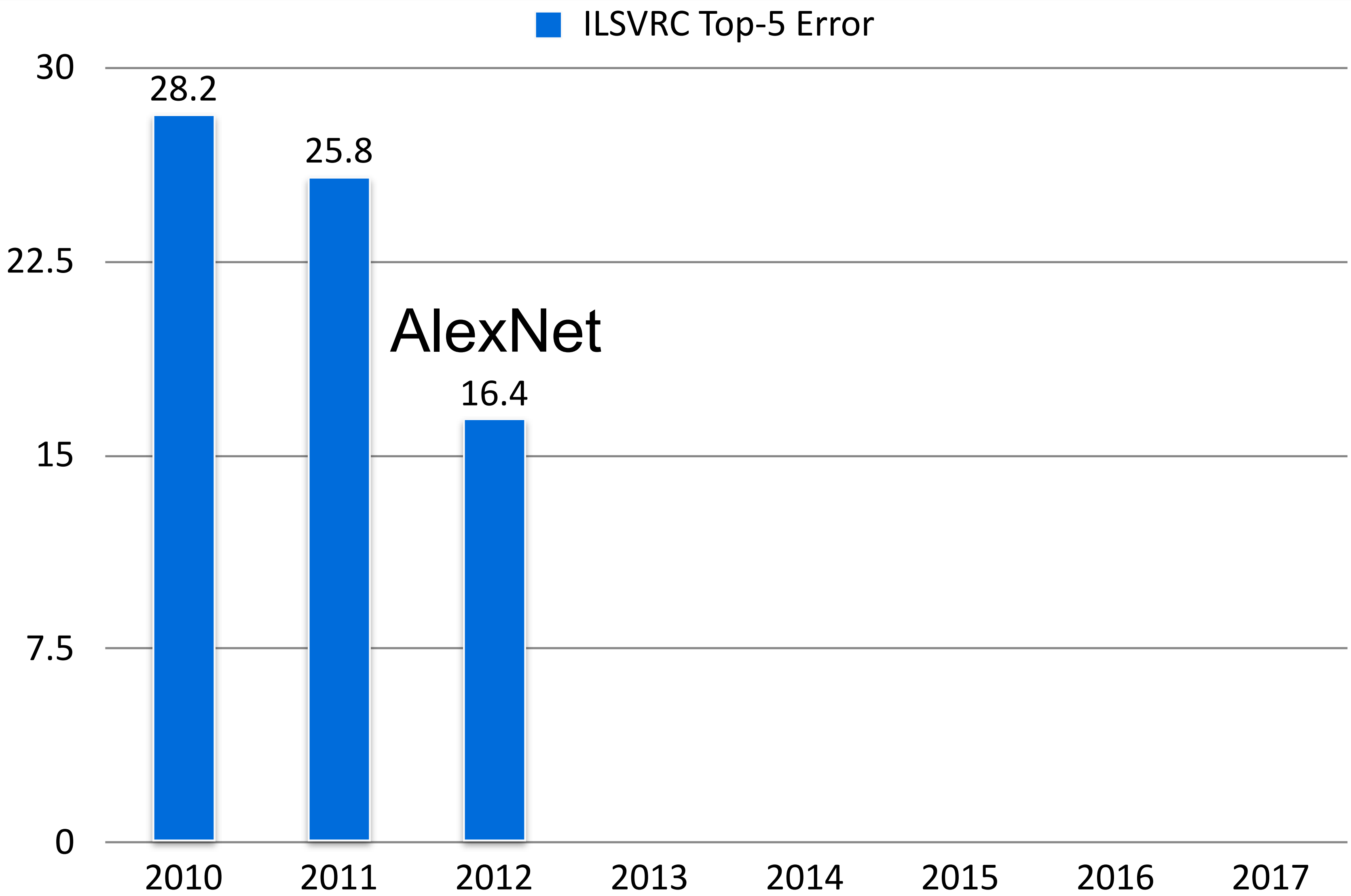

227
3
227
11
11
CONV
11x11,
stride=4,
96 kernels
(227-11)/4 +1
= 55
96
55
55
Overlapping
Max POOL
3x3,
stride=2
(55-3)/2 +1
= 27
96
27
27
CONV
5x5,pad=2
256 kernels
(27+2*2-5)/1
+1 = 27
256
27
27
Overlapping
Max POOL
3x3,
stride=2
(27-3)/2 +1
= 13
256
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
384 kernels
(13+2*1-3)/1
+1 = 13
384
13
13
CONV
3x3,pad=1
256 kernels
(13+2*1-3)/1
+1 = 13
256
13
13
Overlapping
Max POOL
3x3,
stride=2
(13-3)/2 +1
= 6
256
6
6
9216
FC
4096
FC
4096
1000
Softmax

Complexity

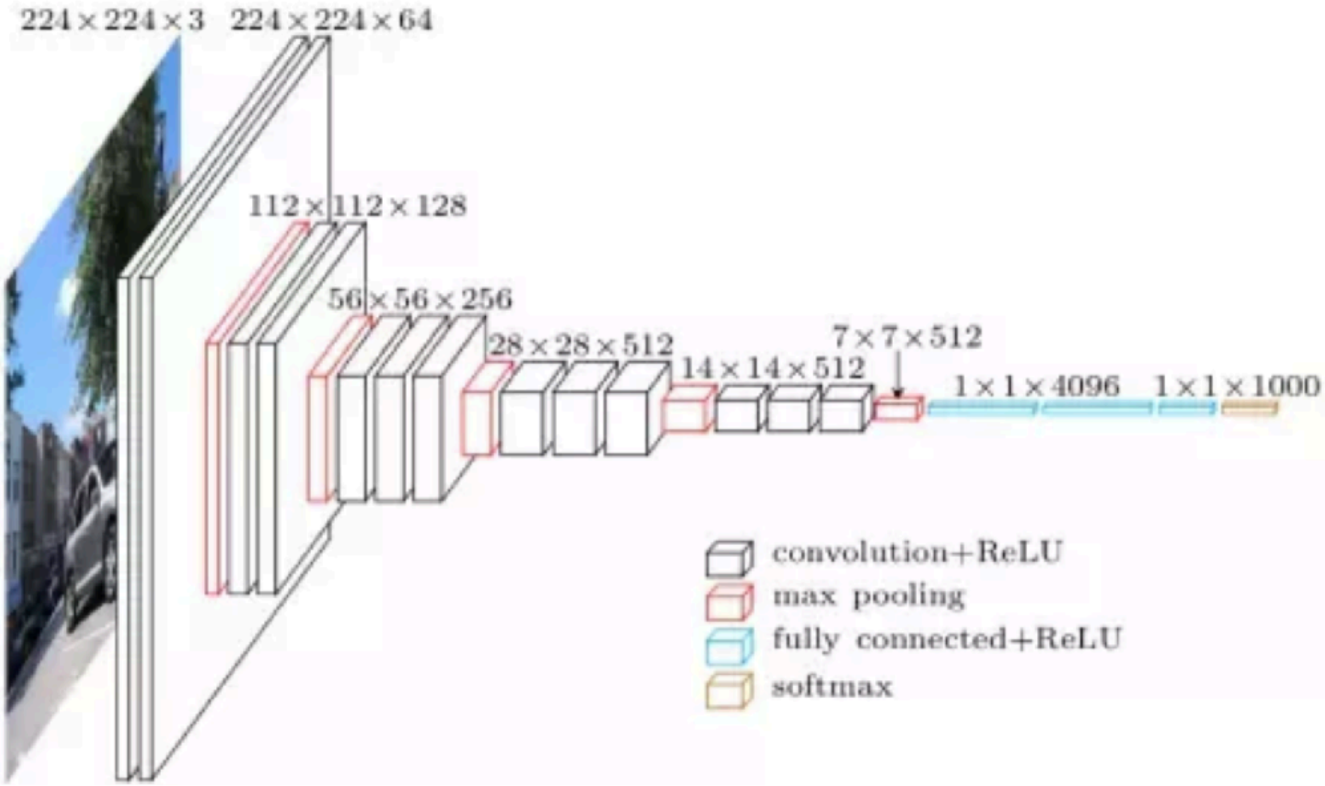
	#parameters		FLOP	
	AlexNet	LeNet	AlexNet	LeNet
Conv1	35K	150	101M	1.2M
Conv2	614K	2.4K	415M	2.4M
Conv3-5	3M		445M	
Dense1	26M	0.48M	26M	0.48M
Dense2	16M	0.1M	16M	0.1M
Total	46M	0.6M	1G	4M
Increase	11x	1x	250x	1x



ImageNet Results: ILSVRC Winners

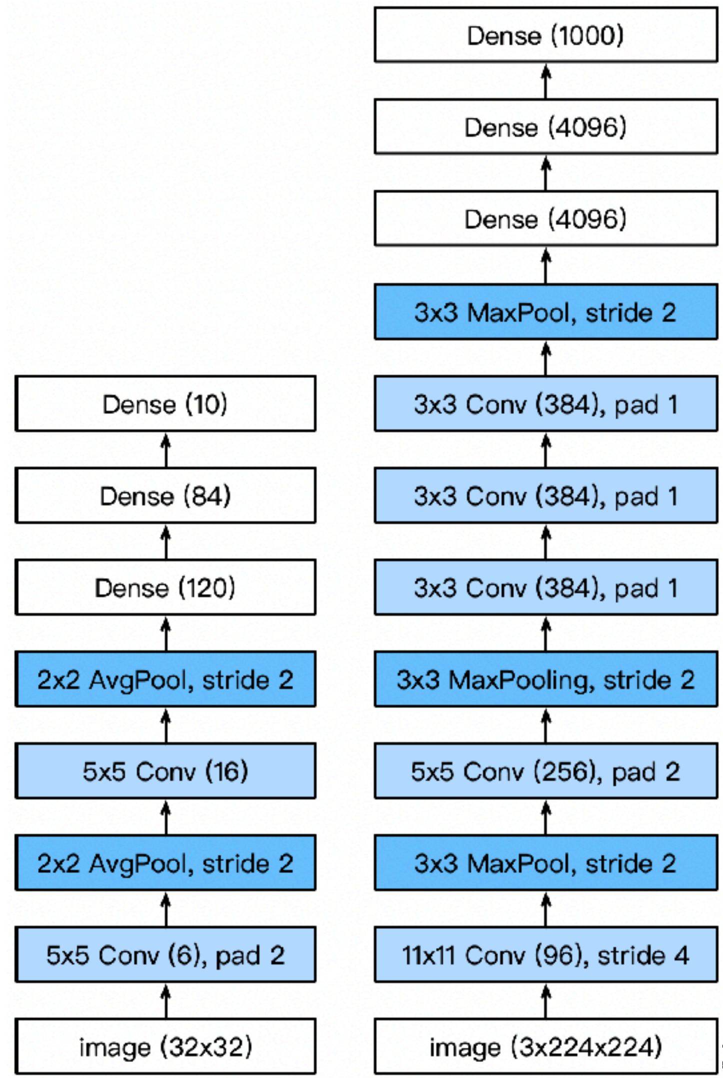


VGG


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

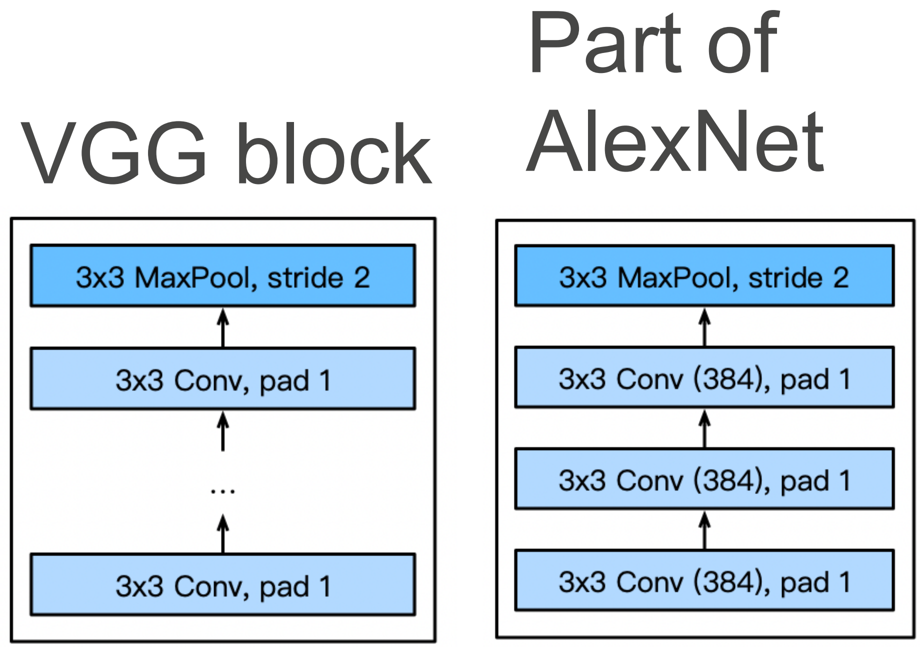
VGG

- AlexNet is deeper and bigger than LeNet to get performance
- Go even bigger & deeper?
- Options
  - More dense layers (too expensive)
  - **More** convolutions
  - Group into **blocks**



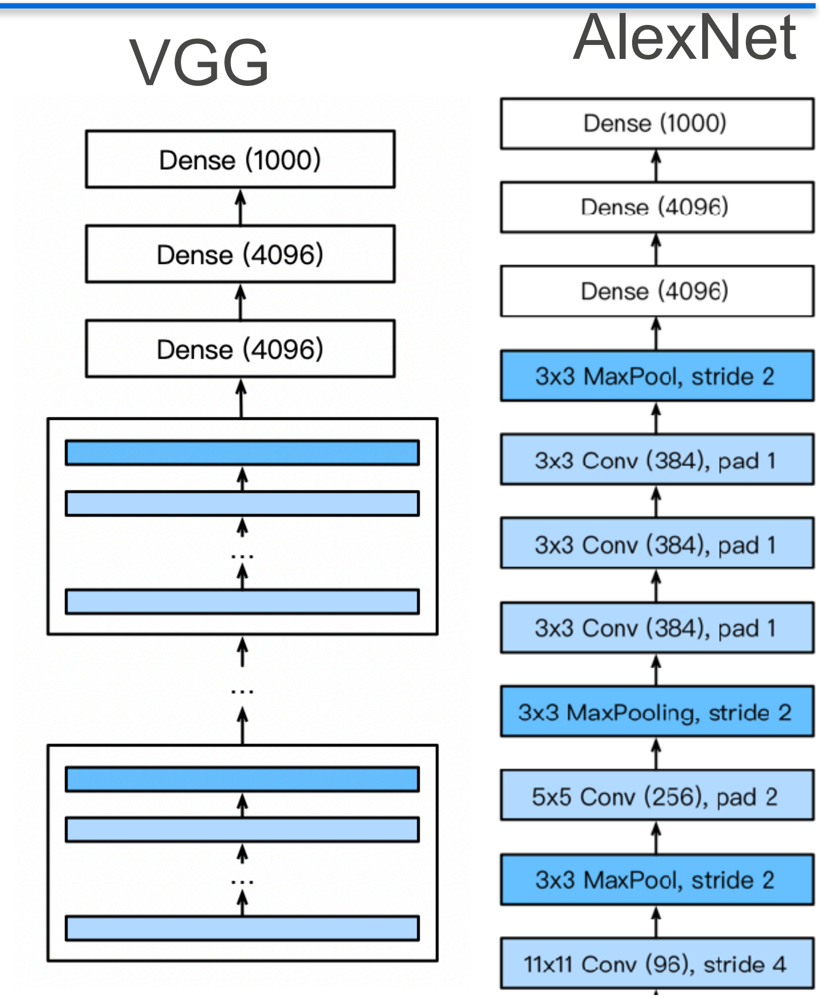
VGG Blocks

- Deeper vs. wider?
  - 5x5 convolutions
  - 3x3 convolutions (more)
  - **Deep & narrow better**
- VGG block
  - *3x3* convolutions (pad 1) **(n layers, m channels)**
  - 2x2 max-pooling (stride 2)



VGG Architecture

- Multiple VGG blocks followed by dense layers
- Vary the repeating number to get different architectures, such as VGG-16, VGG-19, …



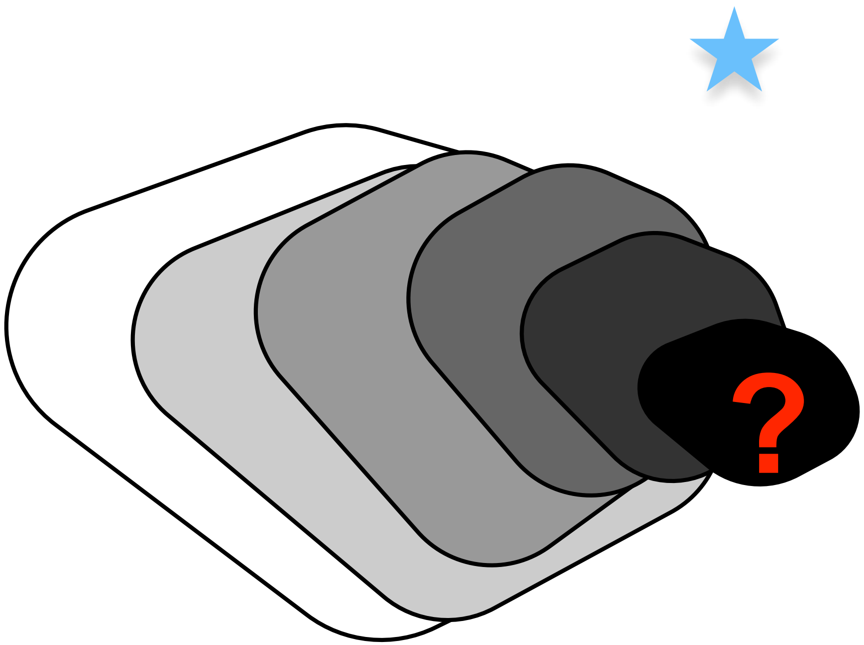
Going Deeper

- LeNet (1995)
  - 2 convolution + pooling layers
  - 2 hidden dense layers
- AlexNet
  - Bigger and deeper LeNet
  - ReLu, Dropout, preprocessing
- VGG
  - Bigger and deeper AlexNet (repeated VGG blocks)

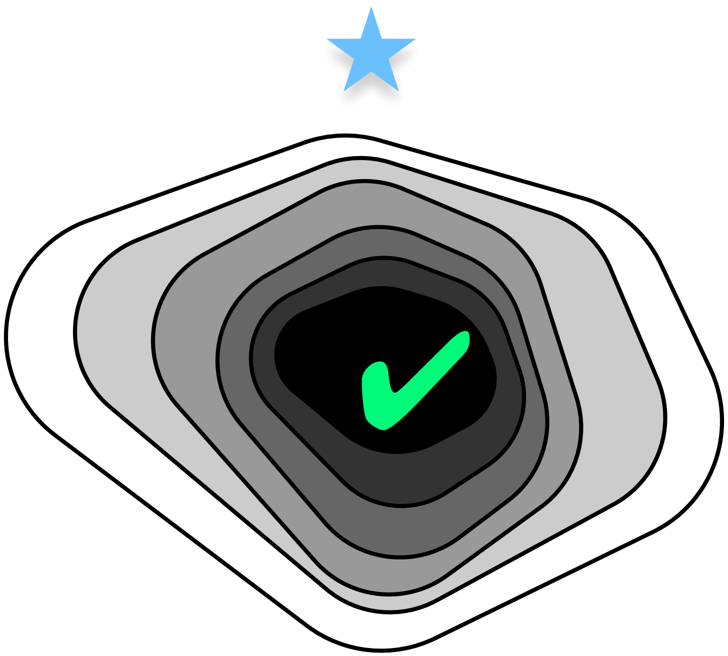
Residual Networks

Best paper CVPR 2016

Does adding layers improve accuracy?



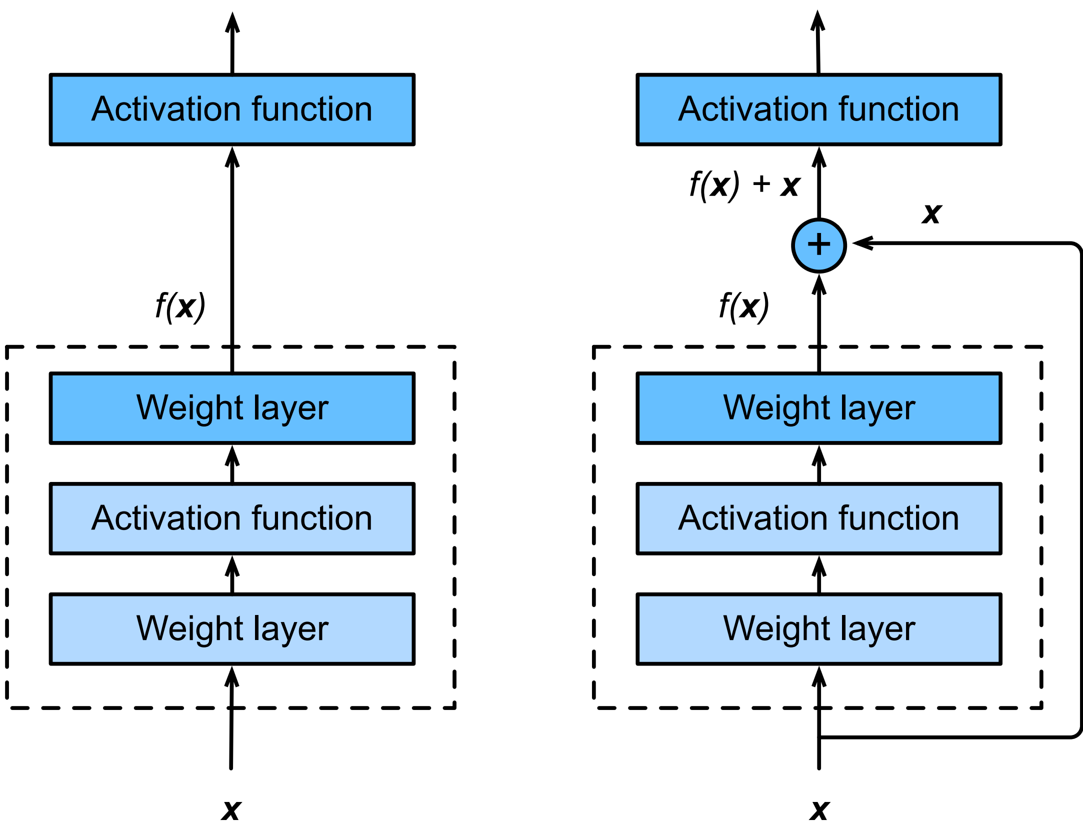
generic function classes



nested function classes

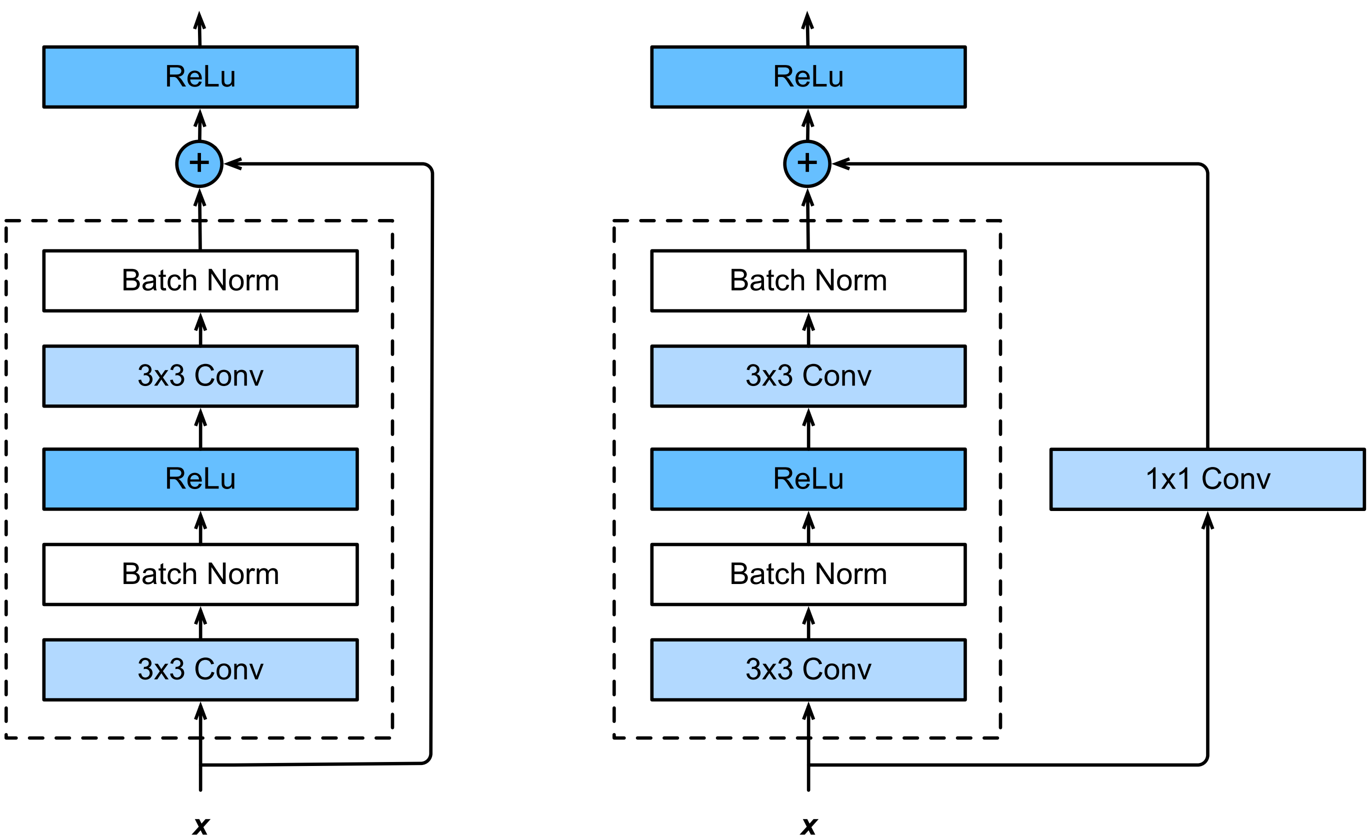
Residual Networks

- Adding a layer **changes** function class
- We want to **add to** the function class
- 'Taylor expansion' style $f(x) = x + g(x)$ parametrization



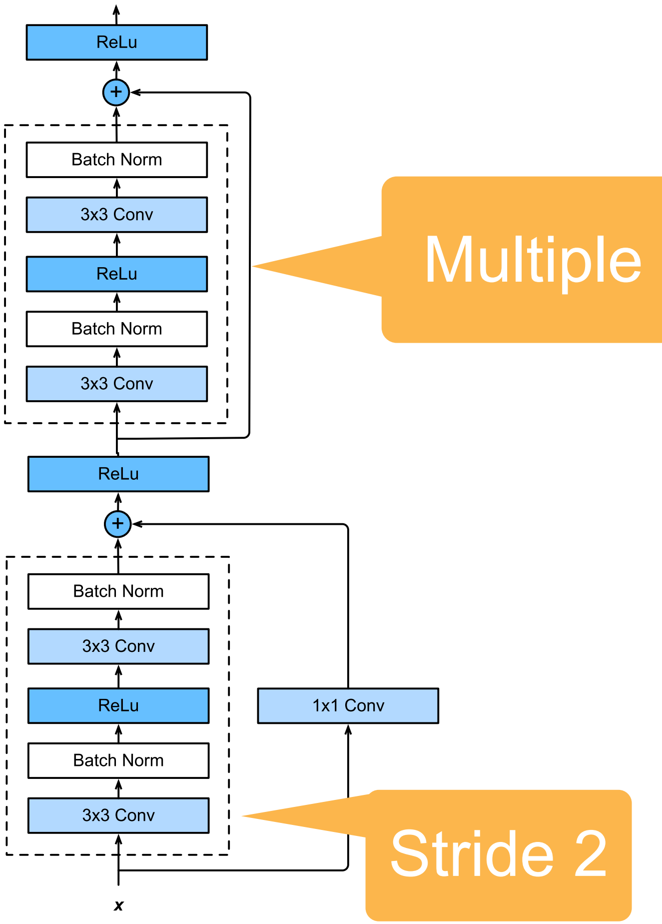
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep Residual Learning for Image Recognition. 2016

ResNet Block in detail



ResNet Module

- Downsample per module (stride=2)
- Enforce some nontrivial nonlinearity per module (via 1x1 convolution)
- Stack up in blocks

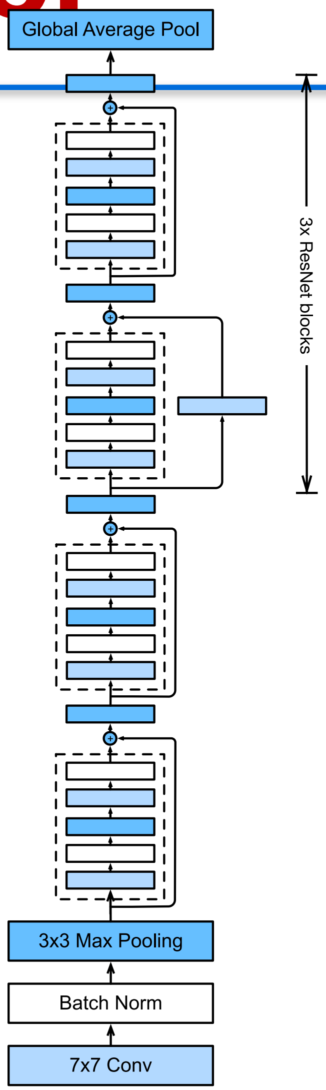


```python
blk = nn.Sequential()
for i in range(num_residuals):
   if i == 0 and not first_block:
      blk.add(Residual(num_channels,
         use_1x1conv=True, strides=2))
   else:
      blk.add(Residual(num_channels))
```

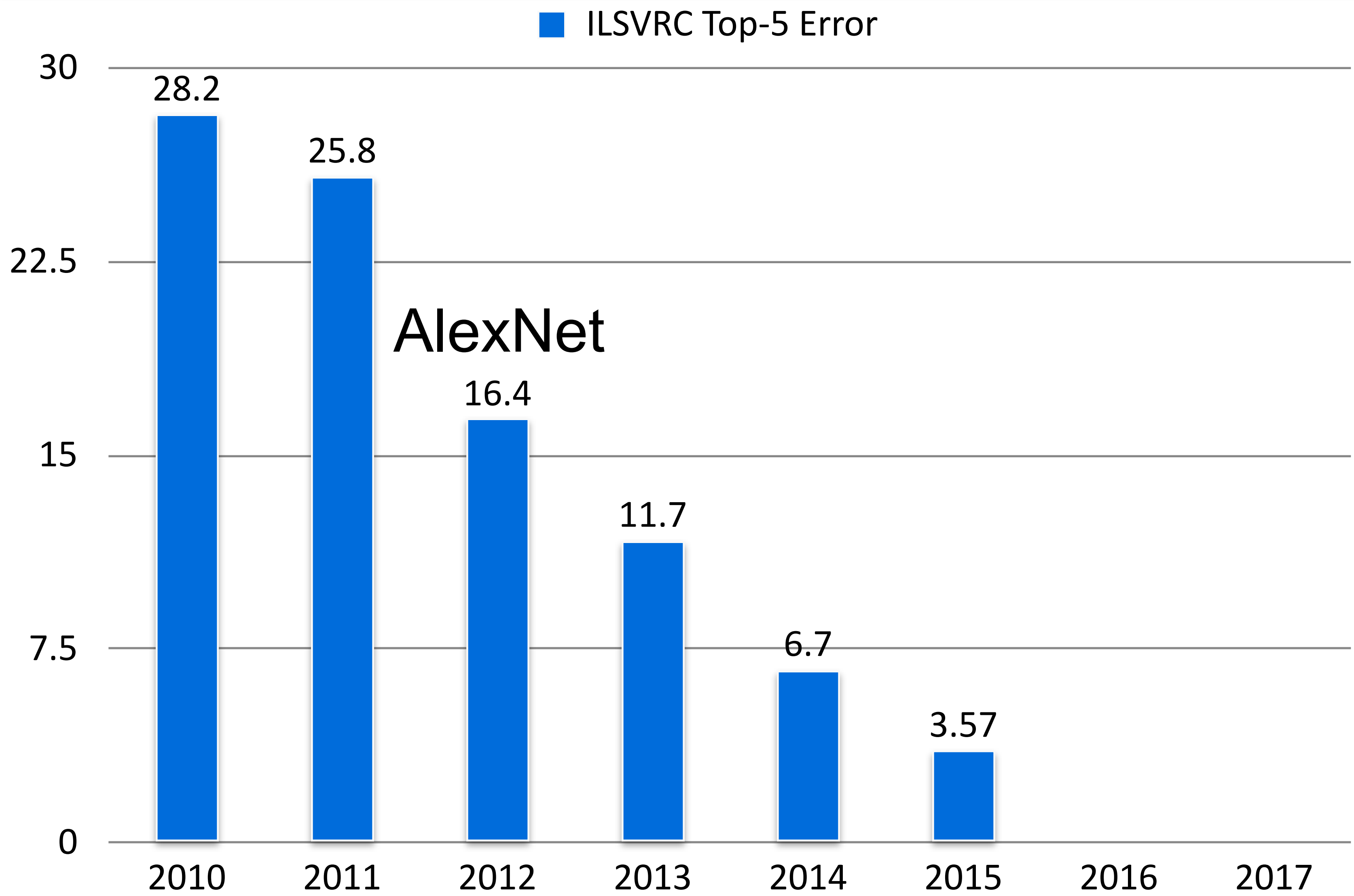
Putting it all together

- Same block structure as e.g. VGG or GoogleNet
- Residual connection to add to expressiveness
- Pooling/stride for dimensionality reduction
- Batch Normalization for capacity control

… train it at scale …



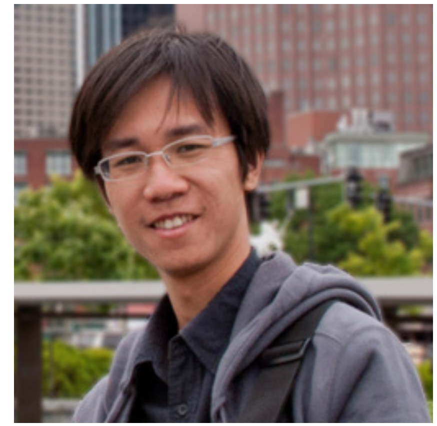
ImageNet Results: ILSVRC Winners



Notes

- ResNet won the champion for ILSVRC 2015
- The ResNet paper won the best paper award from CVPR 2016 (one of the leading CV conferences)
- Kaimin He won multiple best papers.

Papers of Kaimin He



- Exploring Simple Siamese Representation Learning. CVPR Best Paper Honorable Mention, 2021
- Group Normalization. ECCV Best Paper Honorable Mention, 2018
- Mask R-CNN. ICCV Best Paper Award (Marr Prize), 2017
- Focal Loss for Dense Object Detection. ICCV Best Student Paper Award, 2017
- Deep Residual Learning for Image Recognition. CVPR Best Paper Award, 2016
- Single Image Haze Removal using Dark Channel Prior. CVPR Best Paper Award, 2009

ResNext

Reducing the cost of Convolutions

- **Parameters**

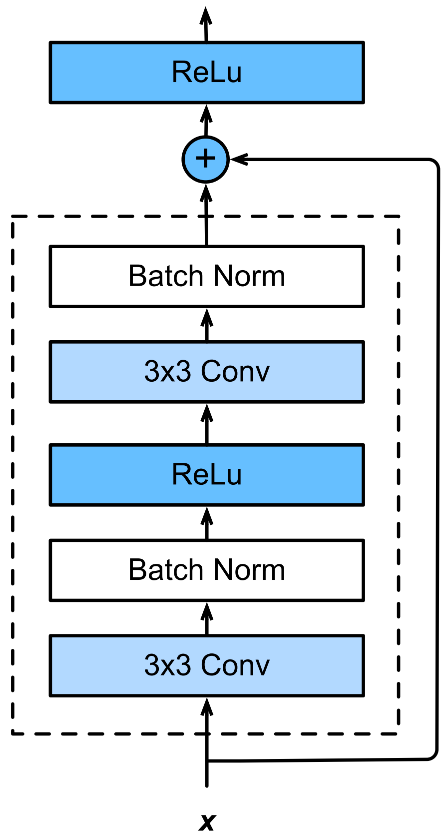
$$k_h \cdot k_w \cdot c_i \cdot c_o$$

- **Computation**

$$m_h \cdot m_w \cdot k_h \cdot k_w \cdot c_i \cdot c_o$$

- **Slicing convolutions** (Inception v4)
e.g. 3x3 vs. **1x5** and **5x1**
- **Break up channels** (mix only within)

$$m_h \cdot m_w \cdot k_h \cdot k_w \cdot \frac{c_i}{b} \cdot \frac{c_o}{b} \cdot b$$



Reducing the cost of Convolutions

- **Parameters**

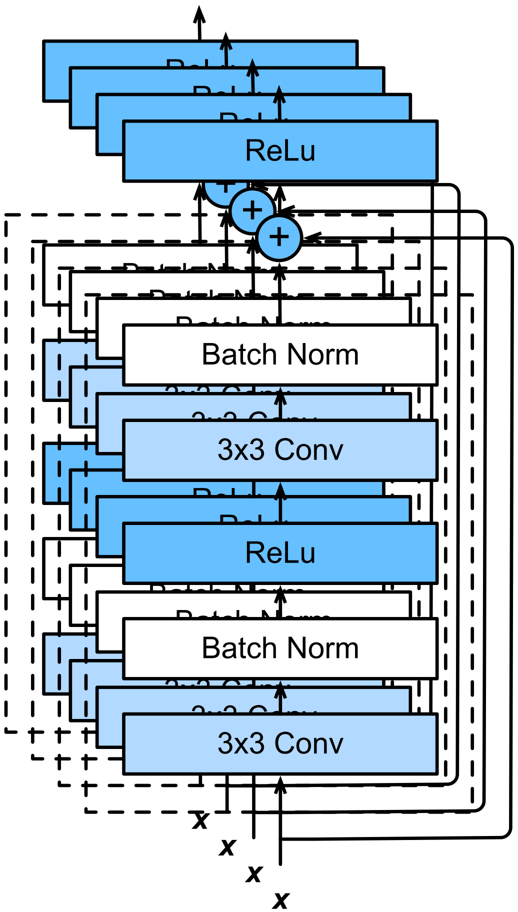
$$k_h \cdot k_w \cdot c_i \cdot c_o$$

- **Computation**

$$m_h \cdot m_w \cdot k_h \cdot k_w \cdot c_i \cdot c_o$$

- **Slicing convolutions** (Inception v4)
  e.g. 3x3 vs. **1x5** and **5x1**
- **Break up channels** (mix only within)

$$m_h \cdot m_w \cdot k_h \cdot k_w \cdot \frac{c_i}{b} \cdot \frac{c_o}{b} \cdot b$$



RexNext budget

- Slice blocks into 32 sub-blocks
- Can use more dimensions
- Higher accuracy

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix} \times 6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**$\times 10^6$	**25.0**$\times 10^6$
FLOPs		**4.1**$\times 10^9$	**4.2**$\times 10^9$

Recap

- AlexNet
  - 11 layers, bigger convolusion
  - ReLu, Dropout, preprocessing
- VGG
  - Bigger and deeper AlexNet (repeated VGG blocks)
  - VGG-16 and VGG-19
- ResNet
  - 50 or 153 layers
  - Residual connection

Next Up

- Advanced optimization methods