Advanced Deep Learning Architectures
COMP 5214 & ELEC 5680

- Instructor: Dr. Qifeng Chen
- https://cqf.io

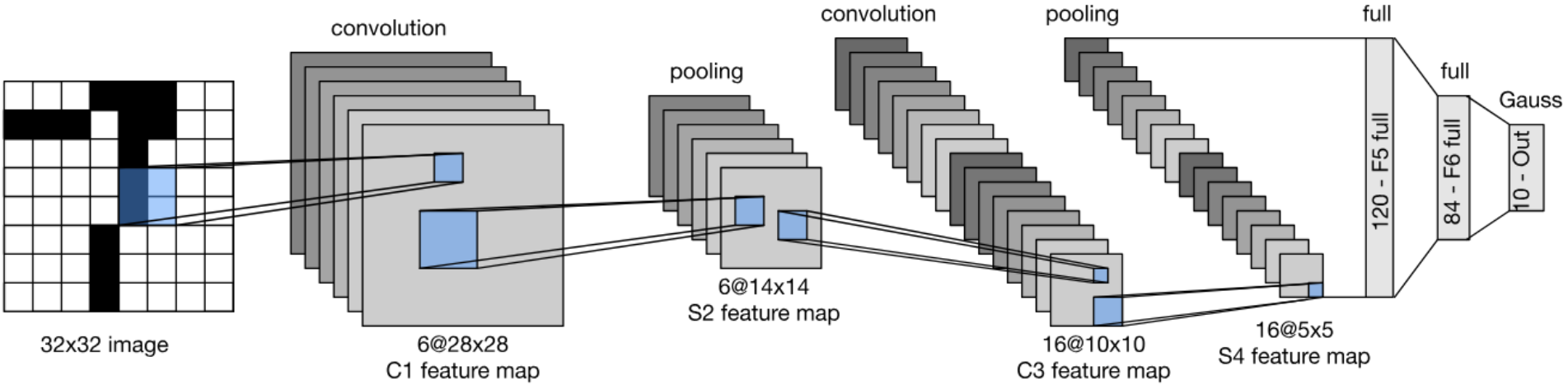
Credit: https://d2l.ai/

Logistics

- The course project details are out
  - https://course.cse.ust.hk/comp5214/project.html
  - 1-3 students in each group
- Find groupmates (if needed)
  - https://docs.google.com/spreadsheets/d/1en0_Dz2qjKcntZZFt33CMQzjzagf5dV2ROzkecsPXgc/edit?usp=sharing
- Project proposal: due Friday, 11 March 2022.
- Project milestone: due Friday, 15 April 2022.
- Final report: due Friday, 13 May 2022.

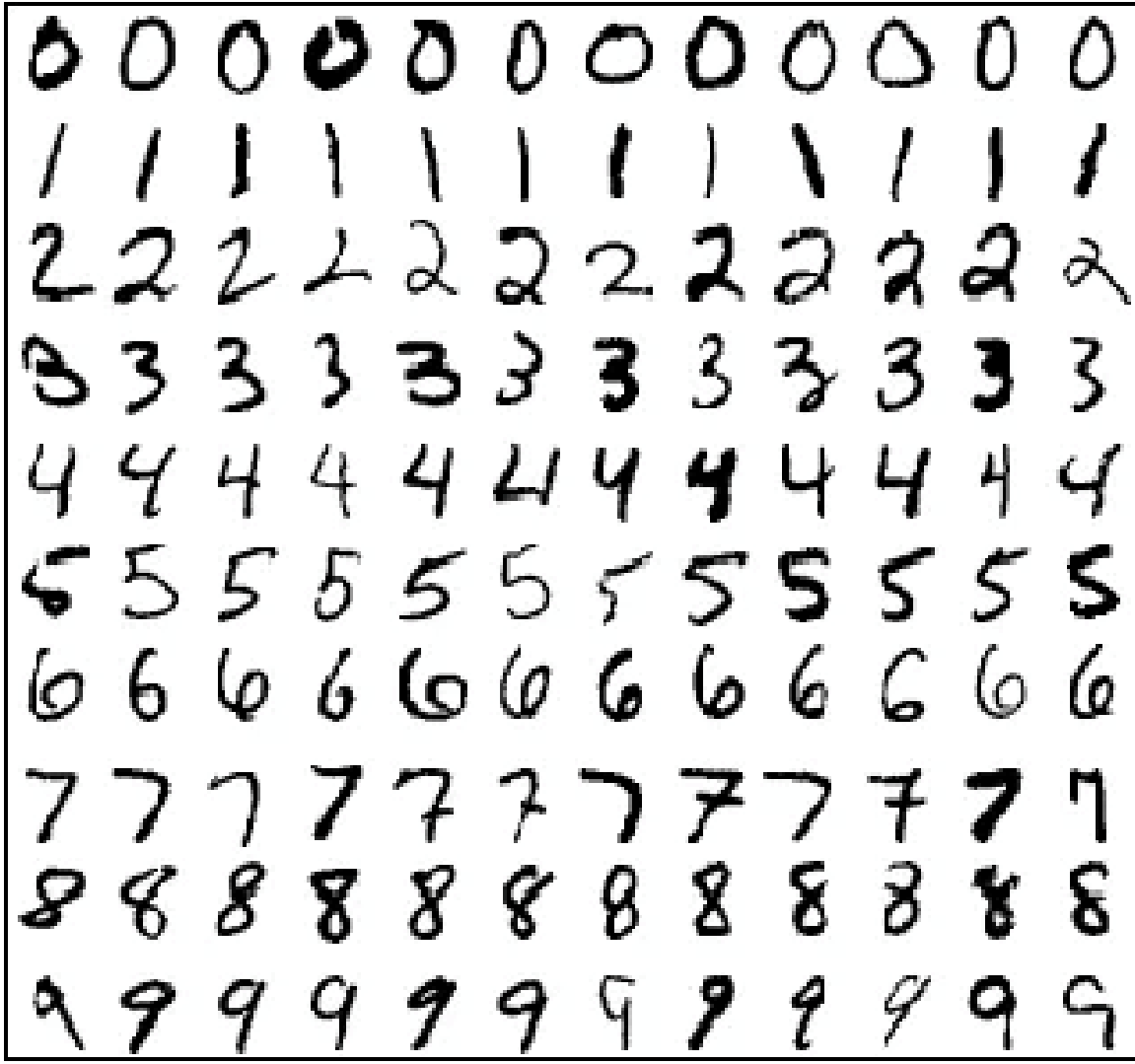
Advanced Convolutional Neural Networks for Image Classification

LeNet Architecture

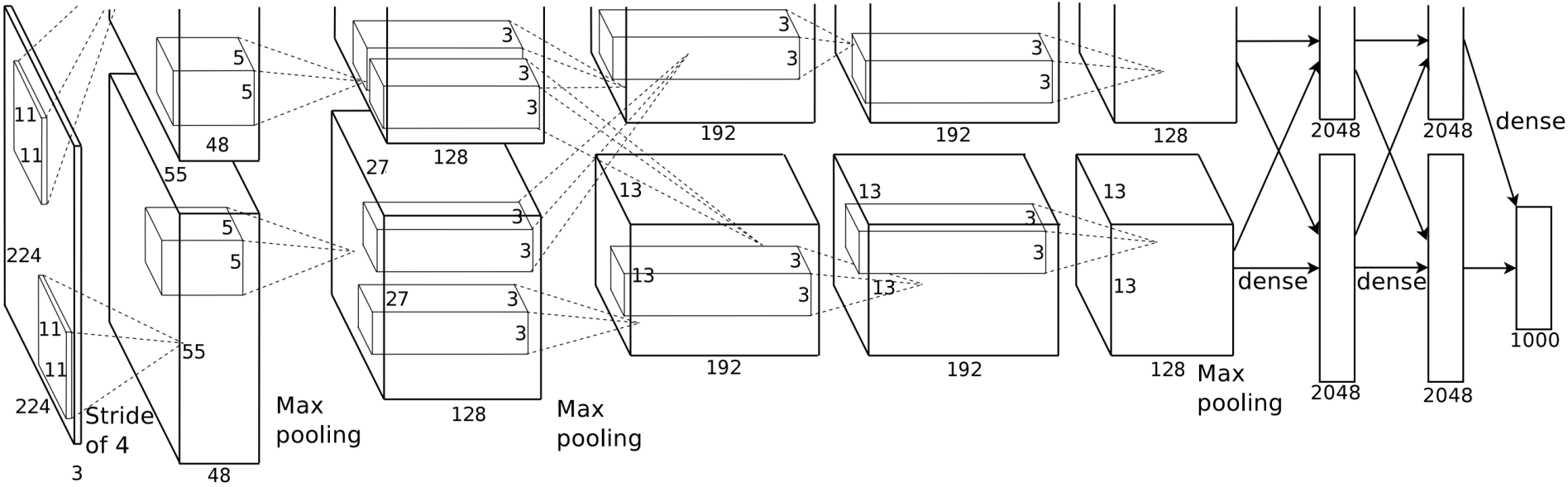


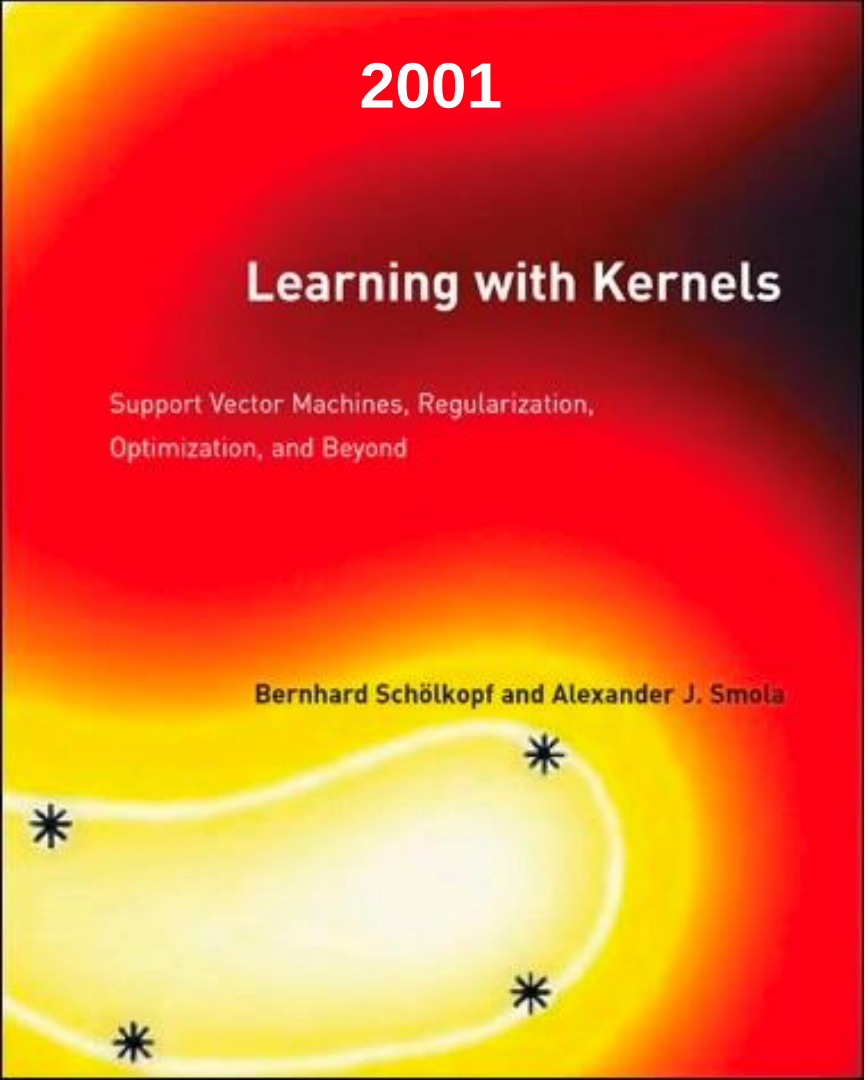
MNIST

- Centered and scaled
- 60,000 training data
- 10,000 test data
- 28 x 28 images
- 10 classes



AlexNet

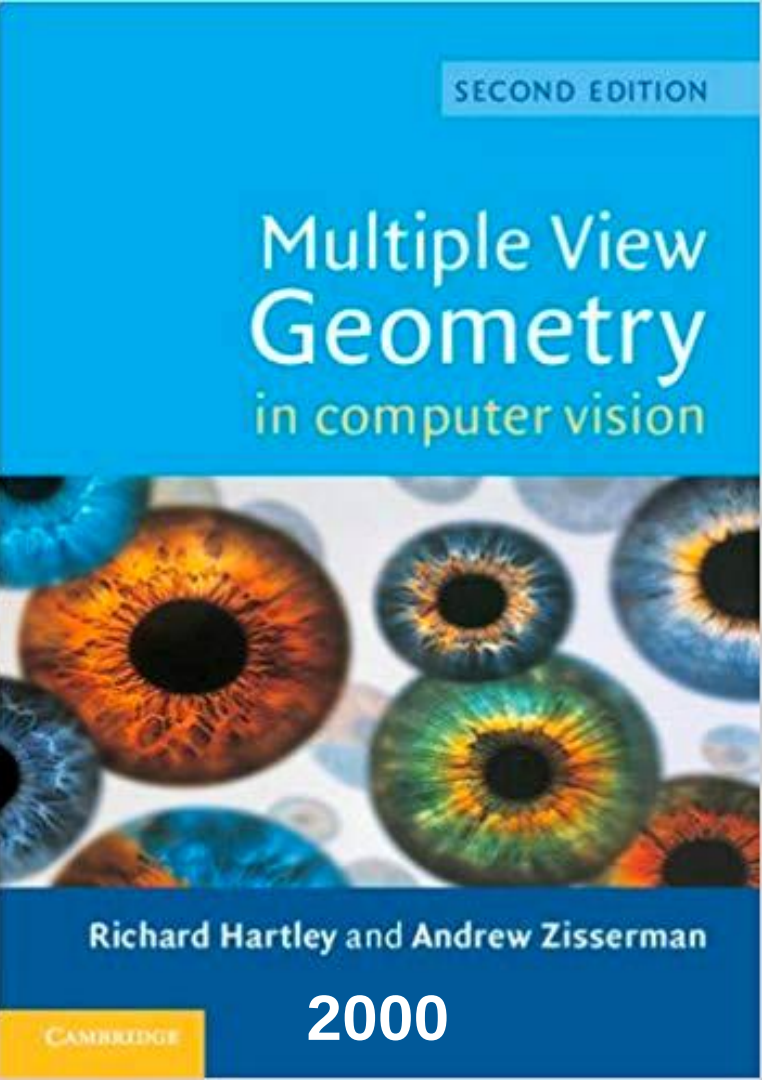

224
224
3
11
11
11
11
Stride
of 4
55
55
48
48
5
5
5
5
Max
pooling
27
27
128
128
3
3
3
3
3
3
Max
pooling
192
192
128
13
13
13
13
13
13
3
3
3
3
3
3
3
3
3
3
3
3
dense
dense
dense
2048
2048
2048
2048
1000
Max
pooling



Function Classes

In the 1990s, a new type of learning algorithm was developed, based on results from statistical learning theory: the Support Vector Machine (SVM). This gave rise to a new class of theoretically elegant learning machines that use a central concept of SVMs – -kernels – for a number of

- Extract features
- Pick kernel for similarity
- **Convex** optimization problem
- Many beautiful theorems …



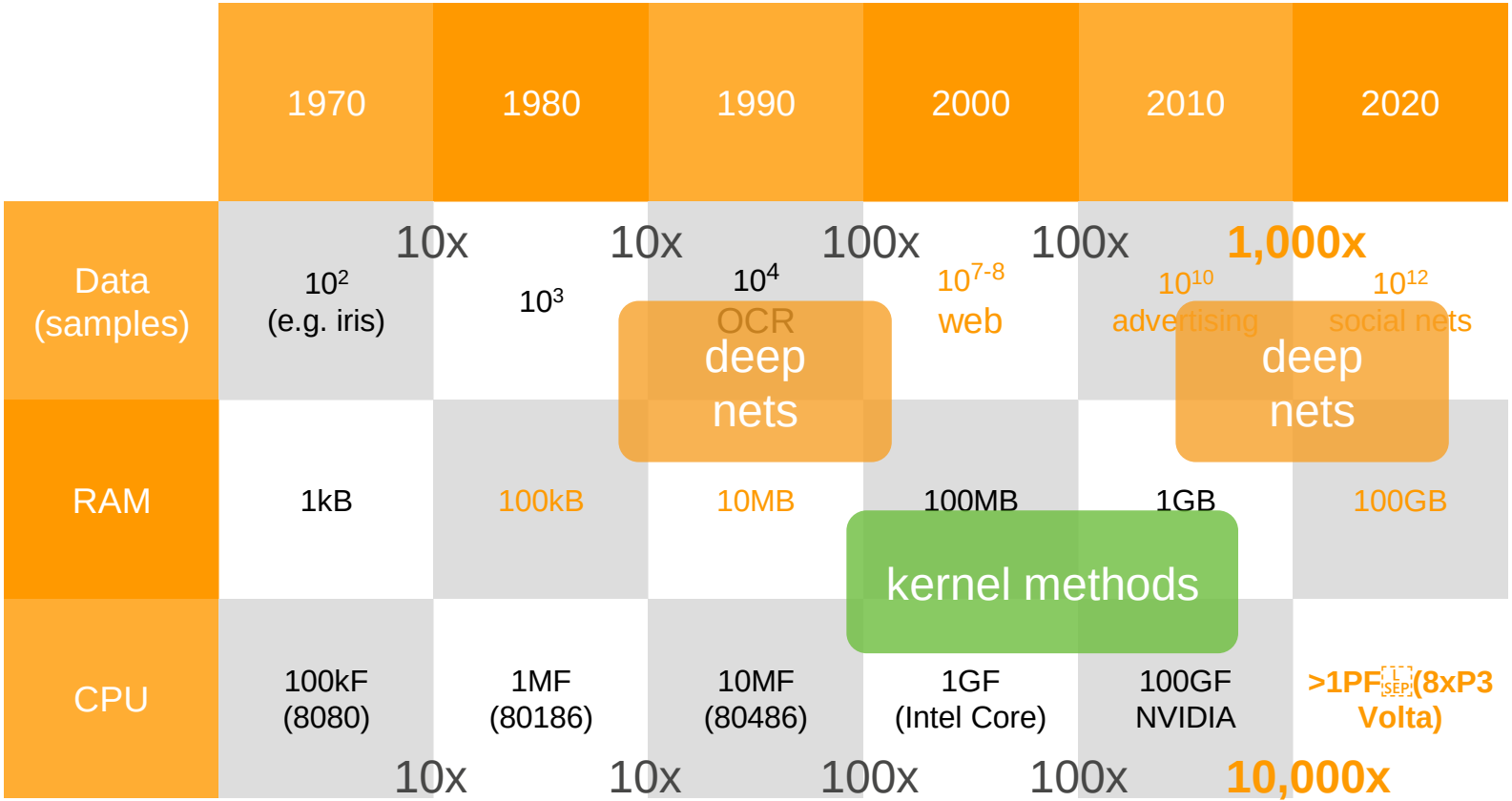
Geometry

- 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

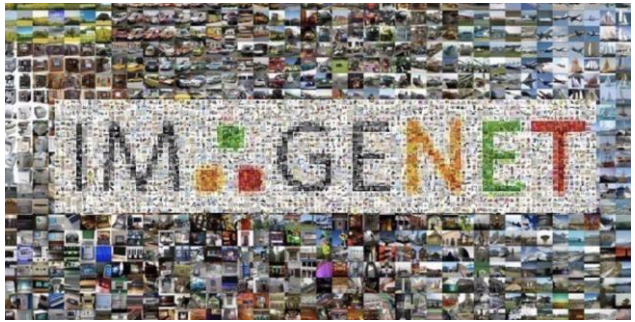
Feature Engineering

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

Hardware


1970
1980
1990
2000
2010
2020
Data (samples)
RAM
CPU
10x
10x
100x
100x
1,000x
10^2 (e.g. iris)
10^3
10^4
OCR
10^{7-8} web
10^{10} advertising
10^{12} social nets
deep nets
deep nets
1kB
100kB
10MB
100MB
1GB
100GB
kernel methods
100kF (8080)
1MF (80186)
10MF (80486)
1GF (Intel Core)
100GF NVIDIA
>1PF (8xP3 Volta)
10x
10x
100x
100x
10,000x

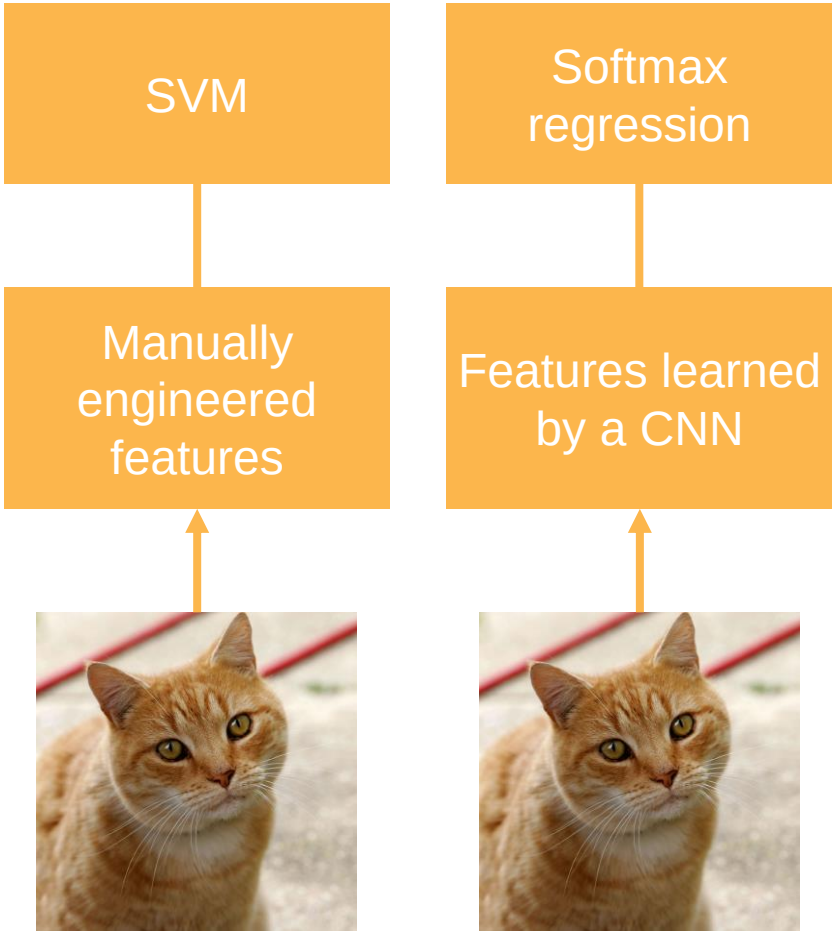
ImageNet (2010)



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

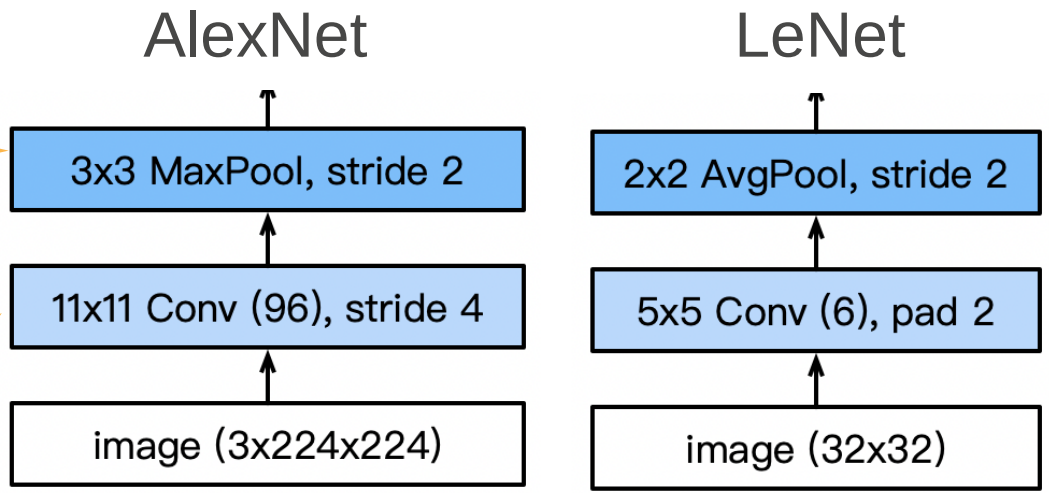
- AlexNet won ImageNet competition in 2012
- Deeper and bigger LeNet
- Key modifications
  - Dropout (regularization)
  - ReLU (training)
  - MaxPooling
- Paradigm shift for computer vision



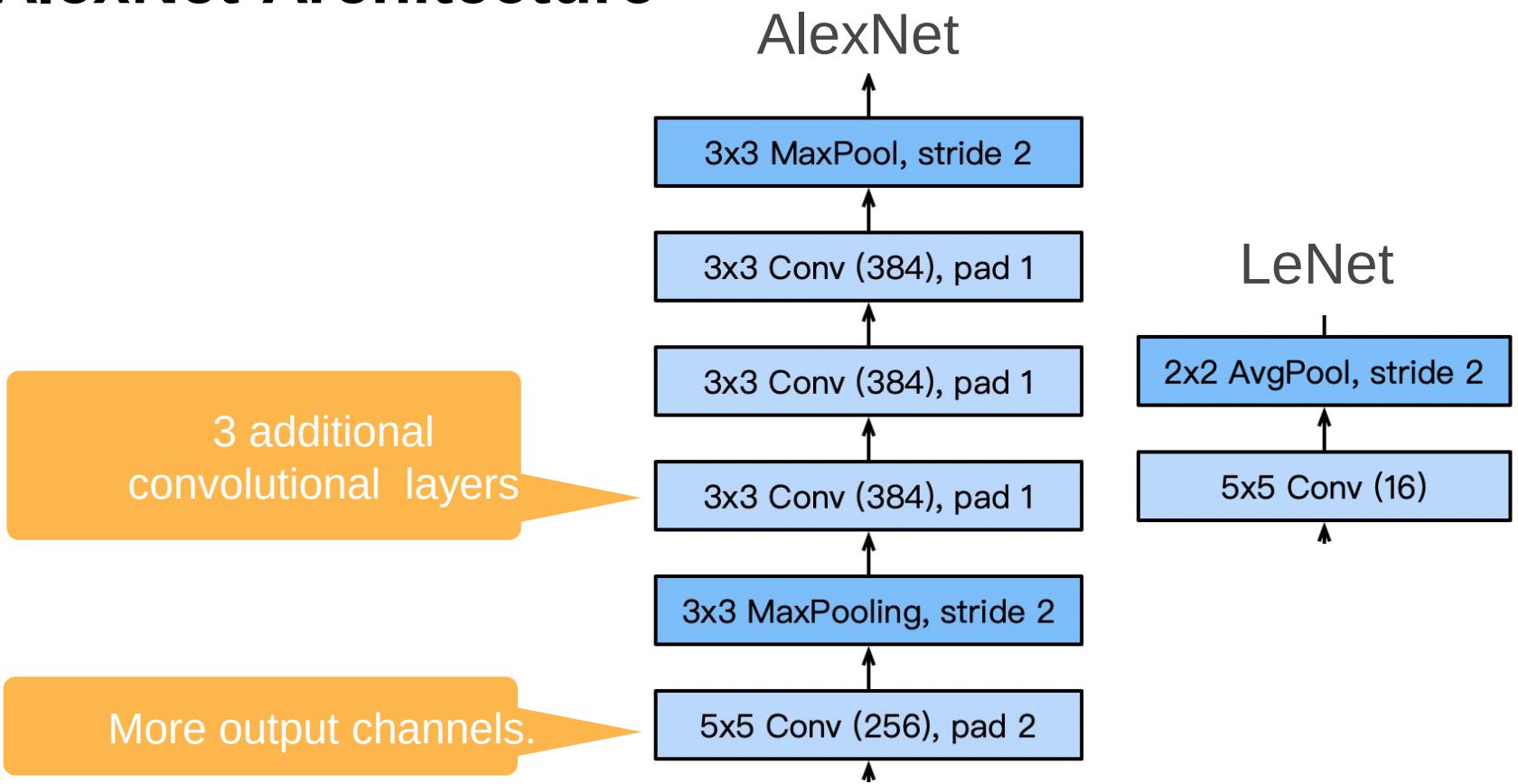
AlexNet Architecture

Larger pool size, change to max pooling

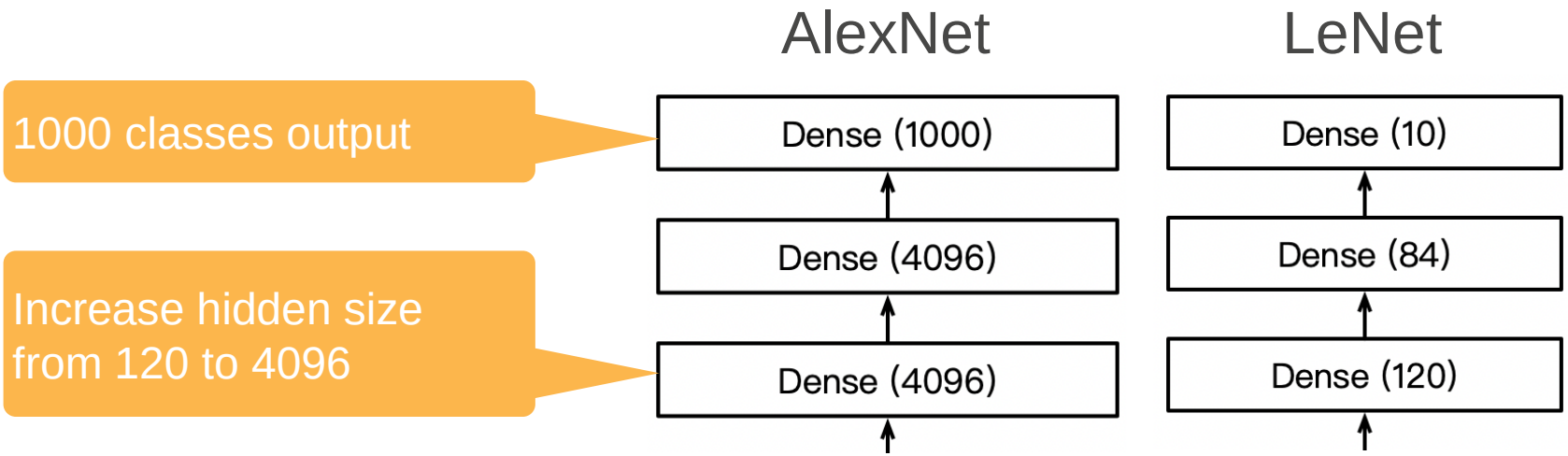
Larger kernel size, stride because of the increased image size, and more output channels.



AlexNet Architecture

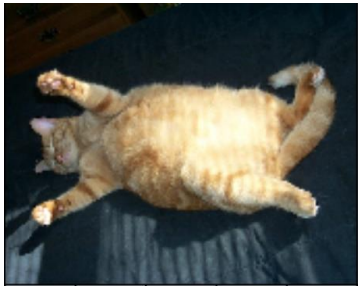
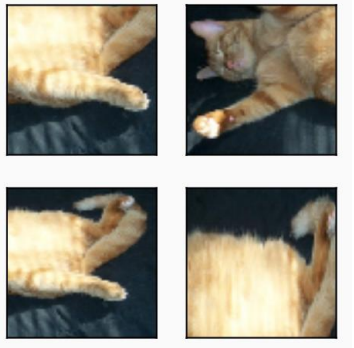
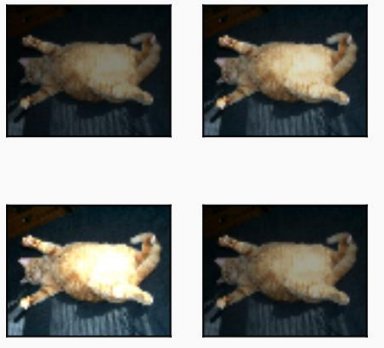
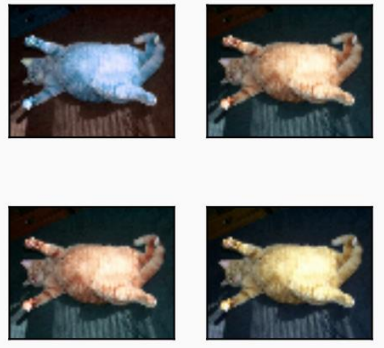


AlexNet Architecture



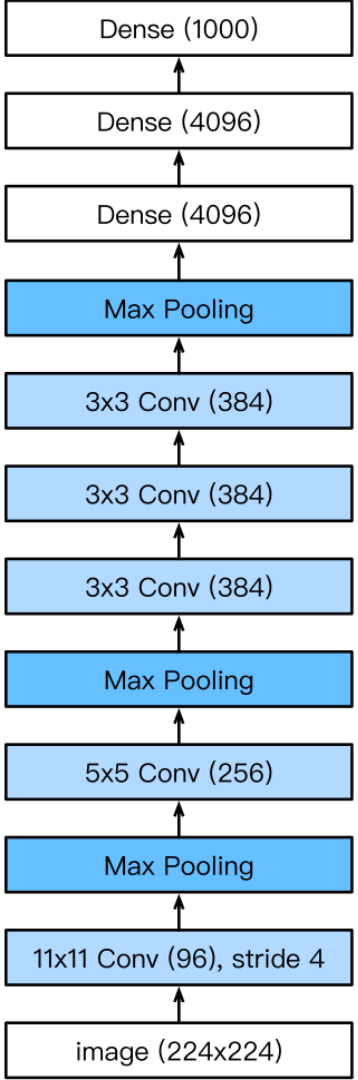
More Tricks

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

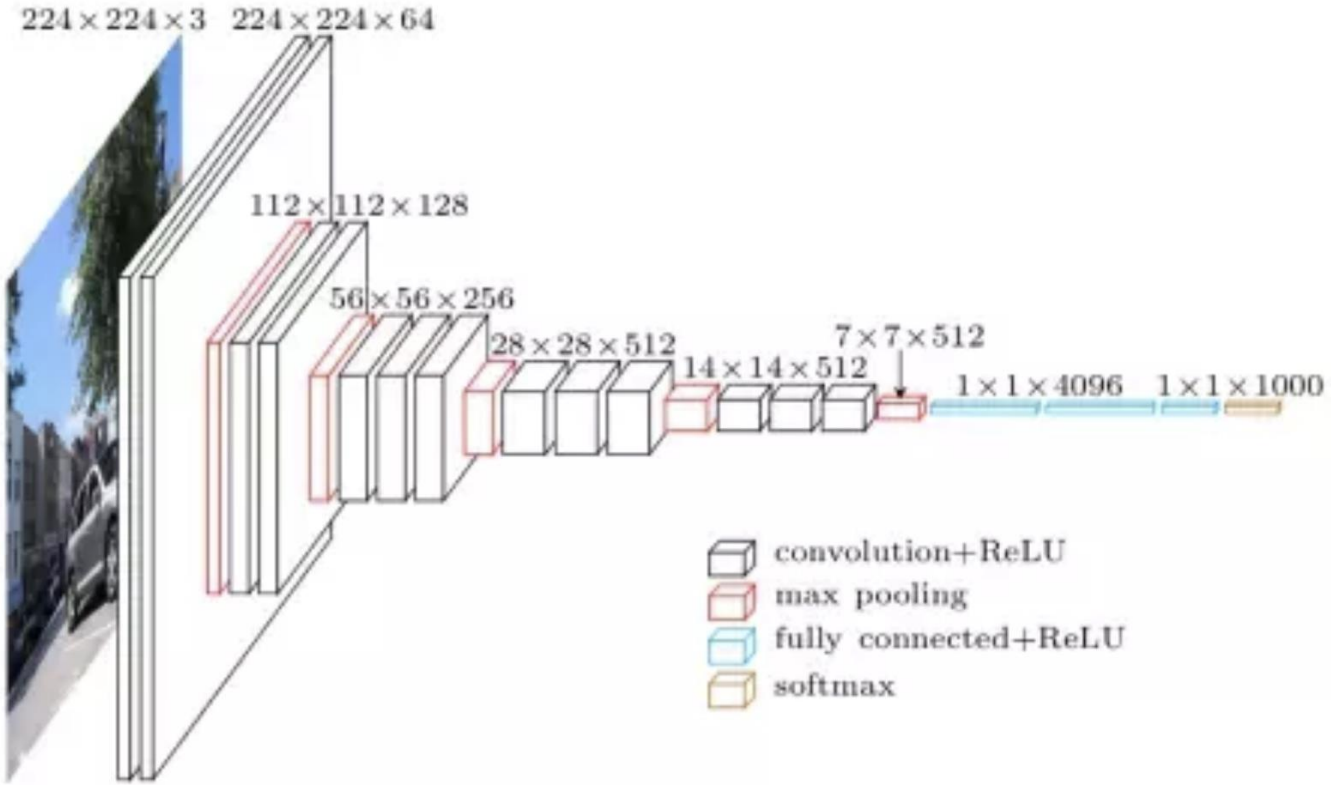





Complexity

	#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	76x	1x	250x	1x

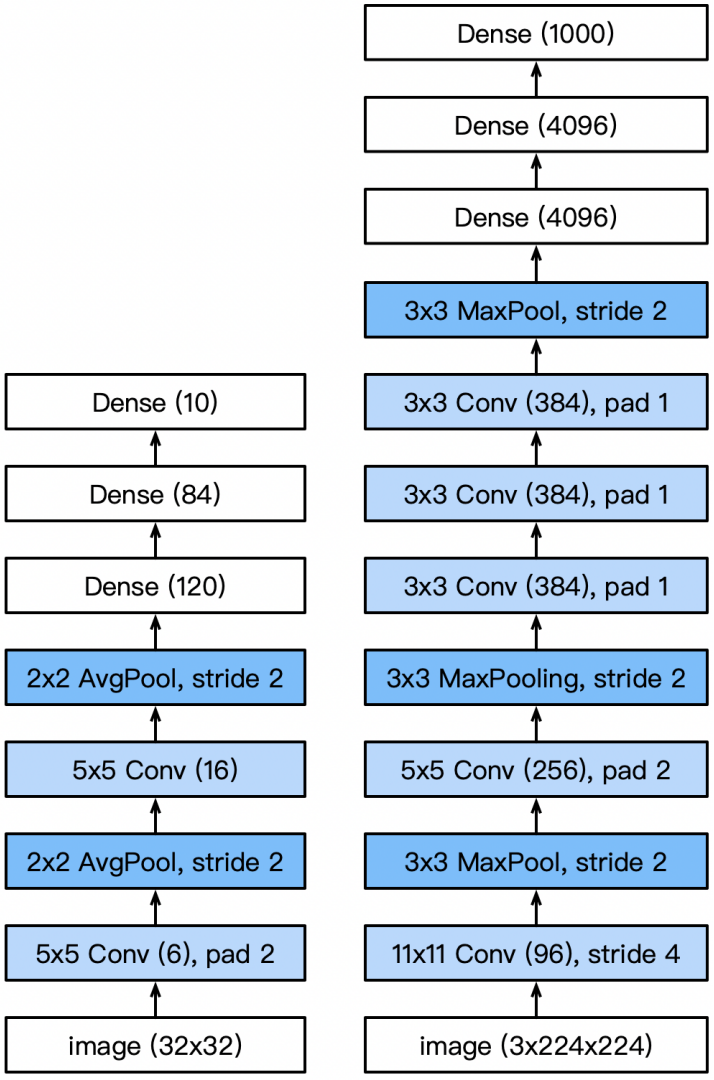


VGG


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

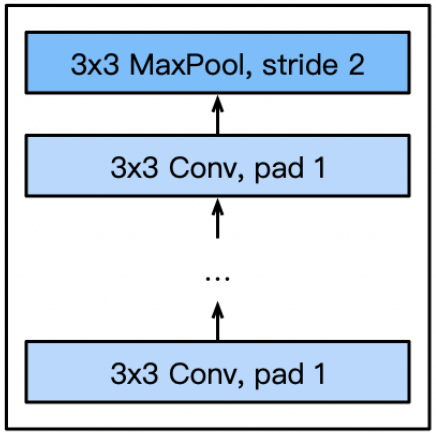
- AlexNet is deeper and bigger than LeNet to get performance
- Go even bigger & deeper?
- Options
  - More dense layers (too expensive)
  - **More** convolutions
  - Group into **blocks**



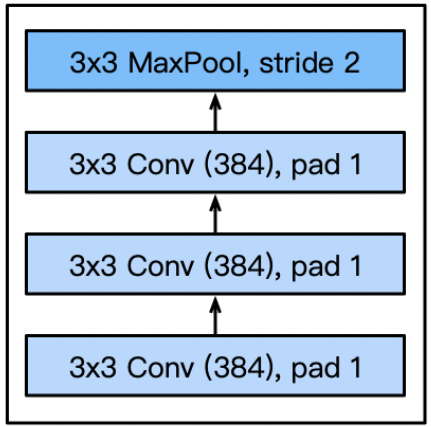
VGG Blocks

- 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 block

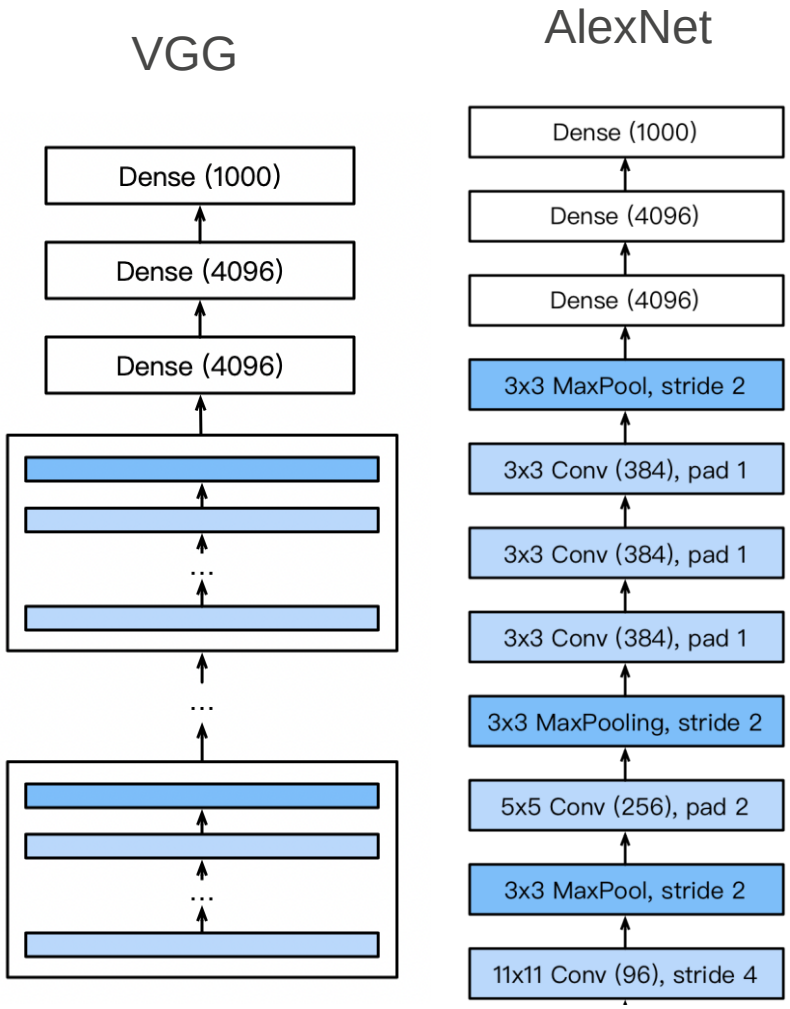


Part of AlexNet



VGG Architecture

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

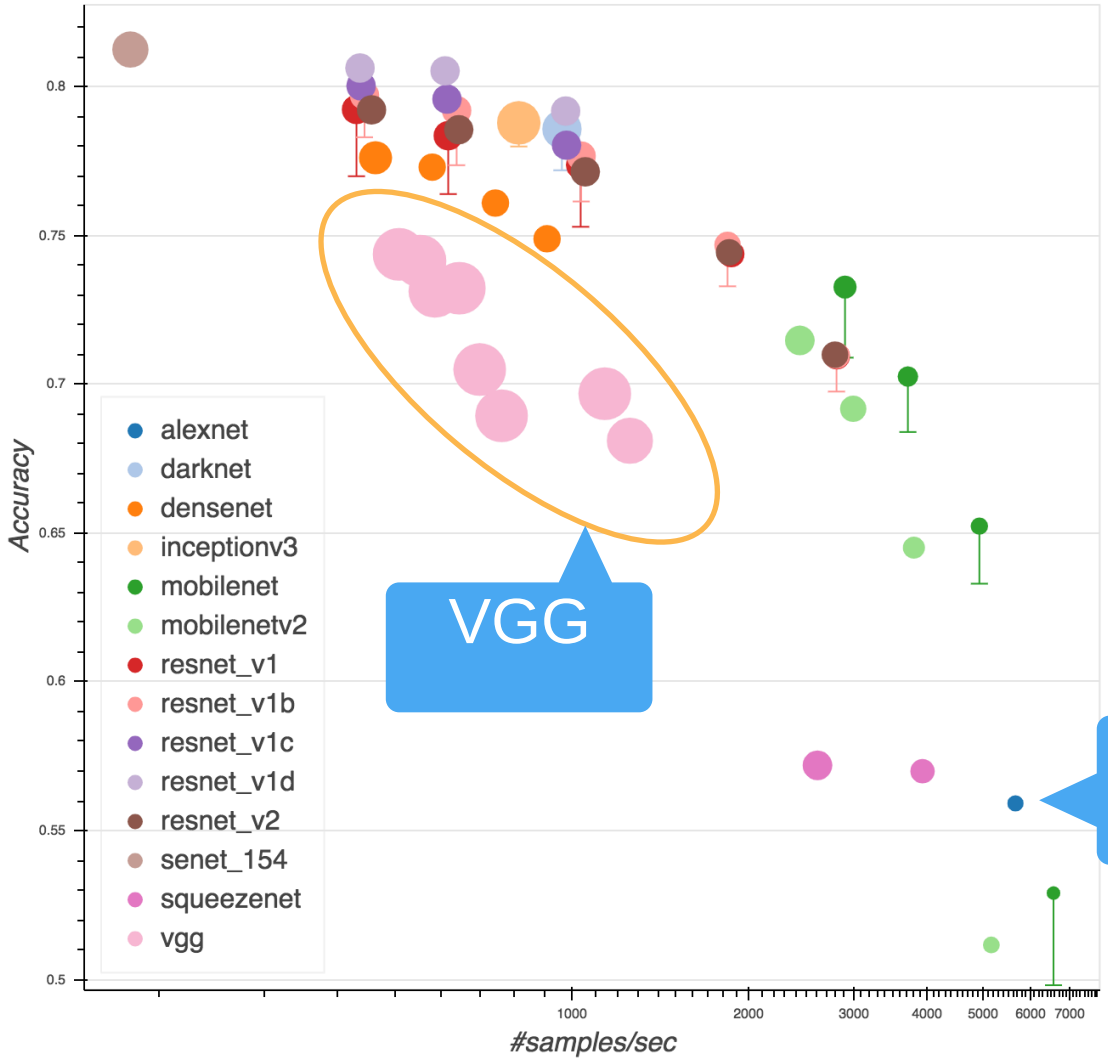


Progress

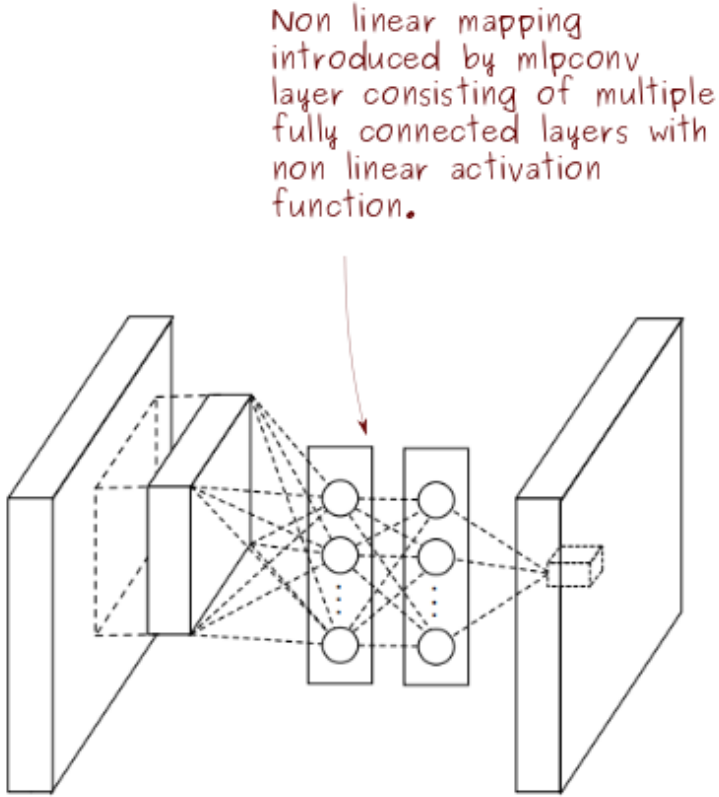
- LeNet (1995)
  - 2 convolution + pooling layers
  - 2 hidden dense layers
- AlexNet
  - Bigger and deeper LeNet
  - ReLu, Dropout, preprocessing
- VGG
  - Bigger and deeper AlexNet (repeated VGG blocks)

GluonCV Model Zoo

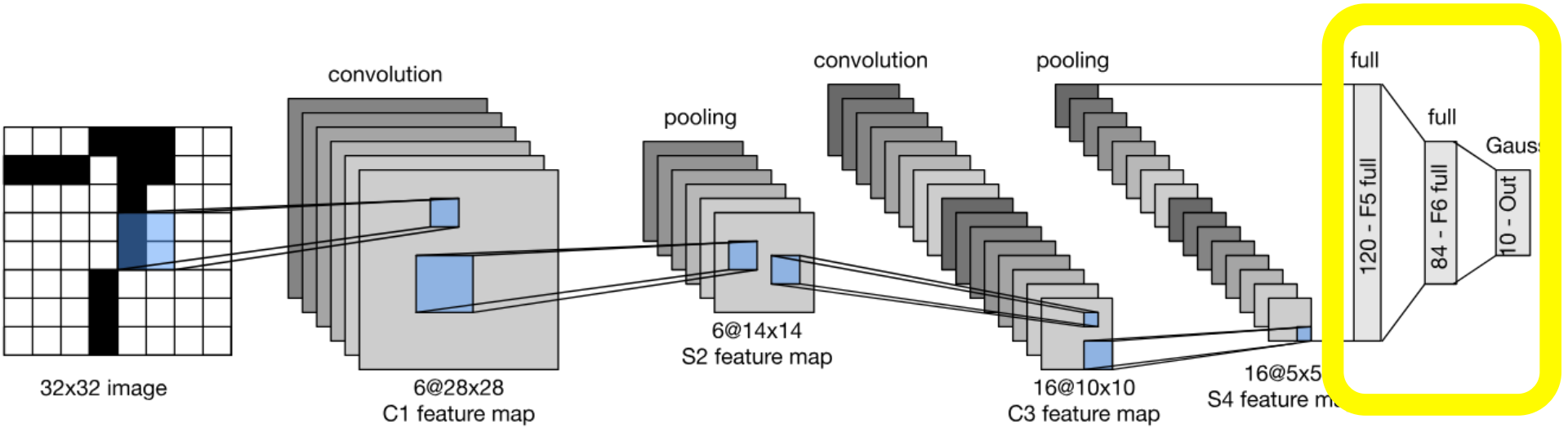
gluon-cv.mxnet.io/model_zoo/classification.html



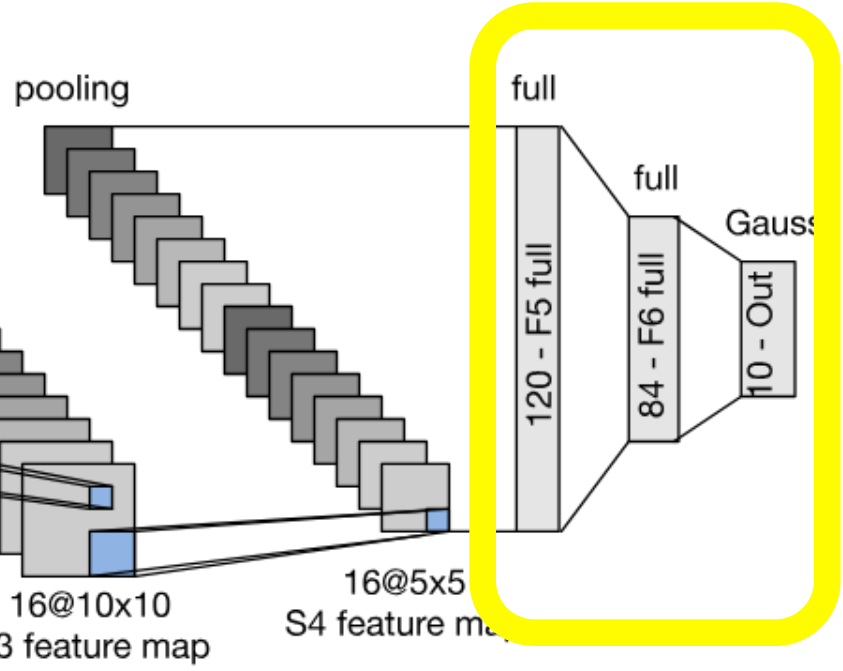
Network in Network



The Curse of the Last Layer(s)



The Last Layer(s)



- Convolution layers need relatively few parameters

$$c_i \times c_o \times k^2$$

- Last layer needs many parameters for n classes

$$c \times m_w \times m_h \times n$$

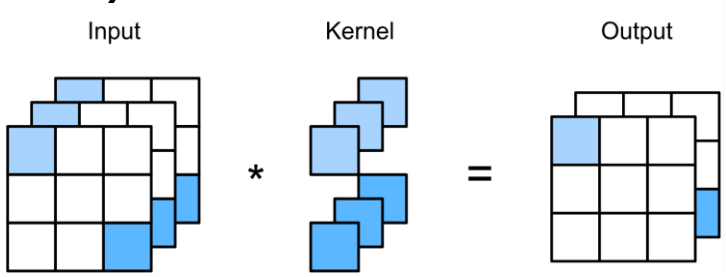
- LeNet 16x5x5x120 = 48k
- AlexNet 256x5x5x4096 = 26M
- VGG 512x7x7x4096 = 102M

VGG parameters

sequential1 output shape: (1, 64, 112, 112)
sequential2 output shape: (1, 128, 56, 56)
sequential3 output shape: (1, 256, 28, 28)
sequential4 output shape: (1, 512, 14, 14)
sequential5 output shape: (1, 512, 7, 7)
dense0 output shape: (1, 4096)
dropout0 output shape: (1, 4096)
dense1 output shape: (1, 4096)
dropout1 output shape: (1, 4096)
dense2 output shape: (1, 10)

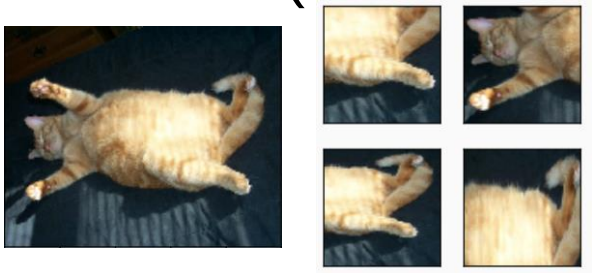
Breaking the Curse of the Last Layer

- Key Idea
  - **Get rid of the fully connected last layer(s)**
  - Convolutions and pooling reduce resolution (e.g. stride of 2 reduces resolution 4x)
- Implementation details
  - Reduce resolution progressively
  - Increase number of channels
  - Use **1x1 convolutions** (they only act per pixel)
- **Global average pooling in the end**

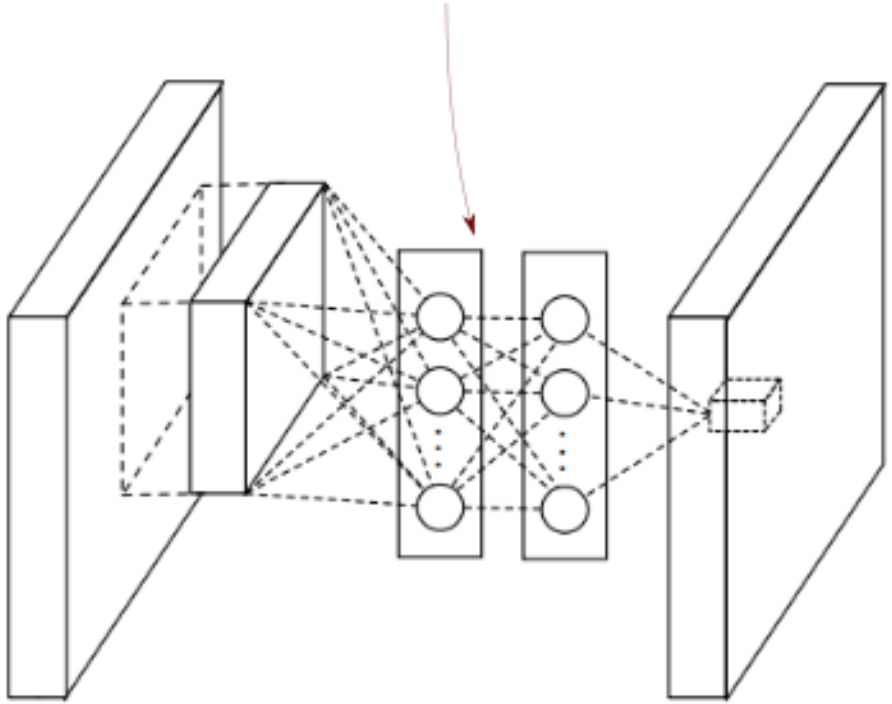


What's a 1x1 convolution anyway?

- Extreme case
  1x1 image with n channels
- Equivalent to MLP
- Pooling allows for translation invariance of detection (e.g. 5x5)

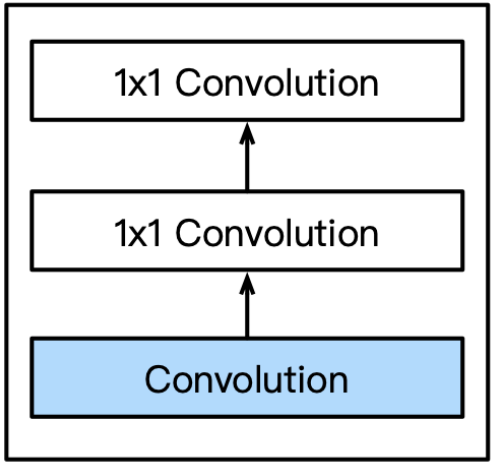


Non linear mapping introduced by mlpconv layer consisting of multiple fully connected layers with non linear activation function.

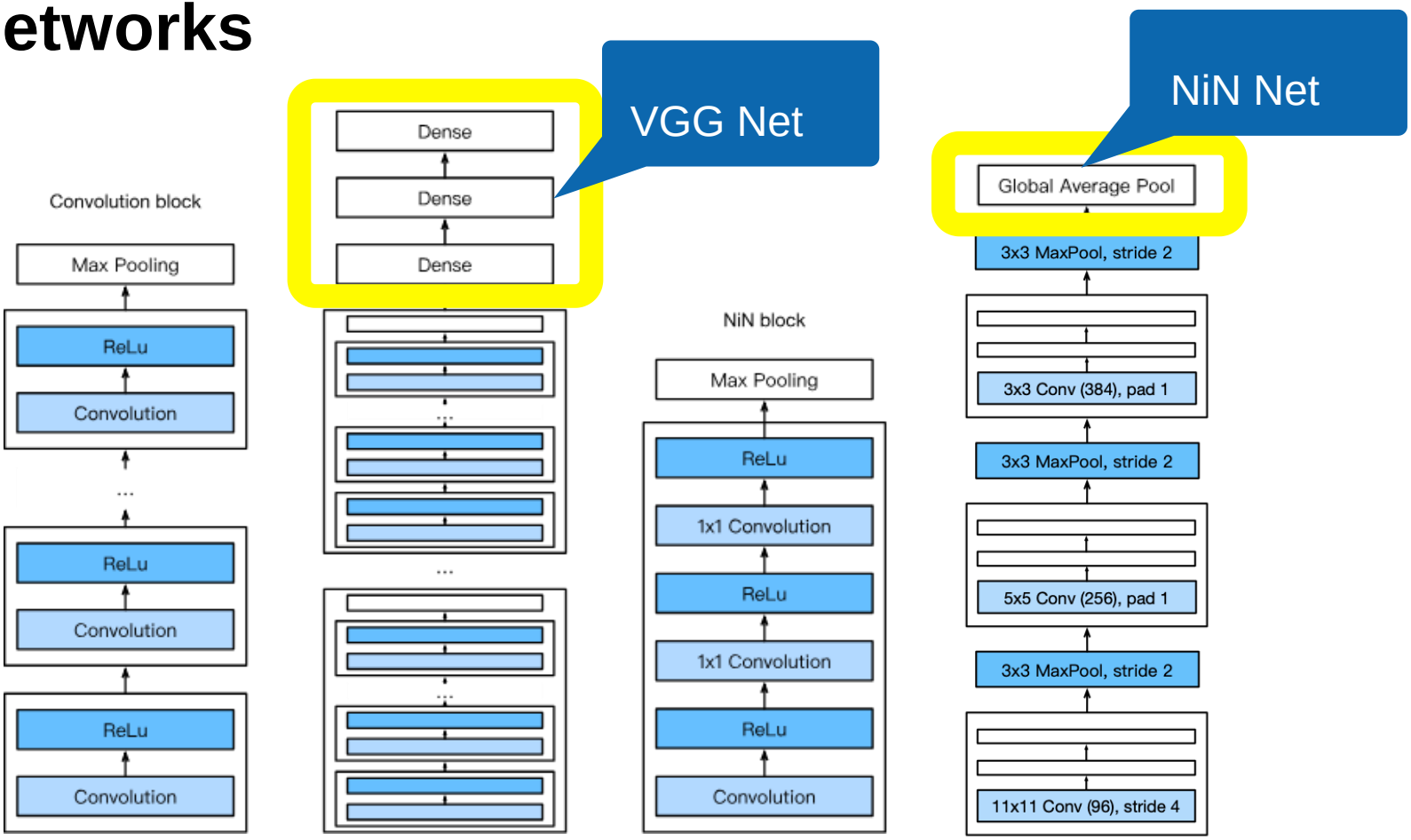


NiN Block

- A convolutional layer
  - kernel size, stride, and padding are hyper-parameters
- Following by two 1x1 convolutions
  - 1 stride and no padding, share the same output channels as first layer
  - Act as dense layers

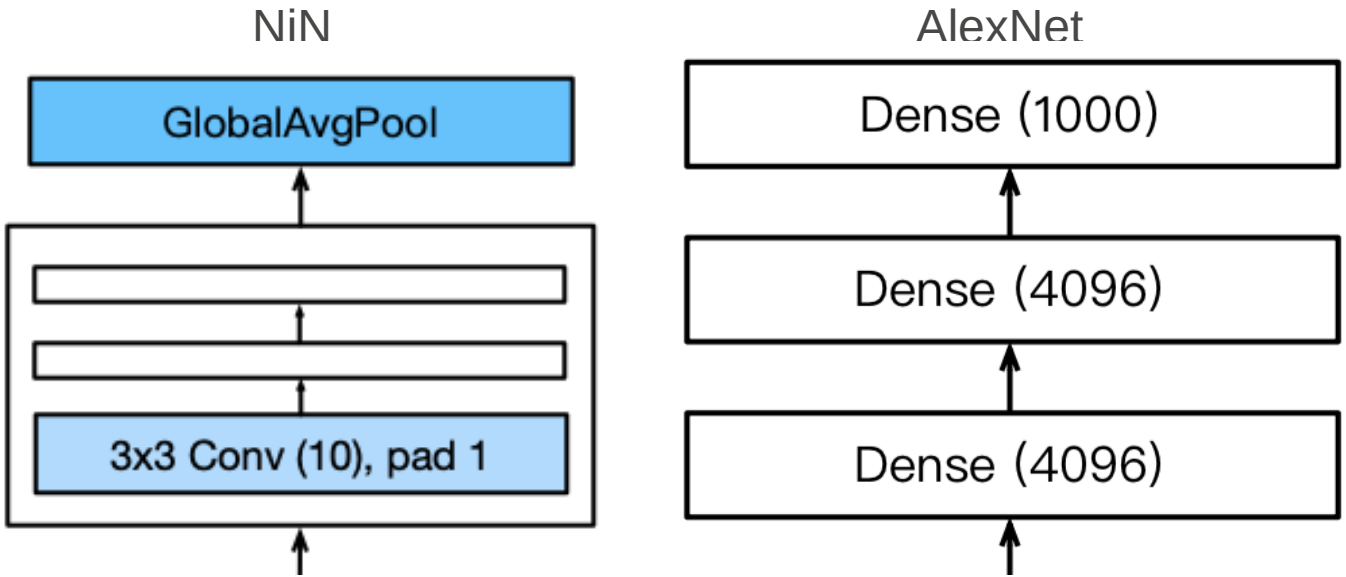


NiN Networks



NiN Last Layers

- Replaced AlexNet's dense layers with a NiN block
- Global average pooling layer to combine outputs



Summary

- Reduce image resolution progressively
- Increase number of channels
- Global average pooling for given number of classes

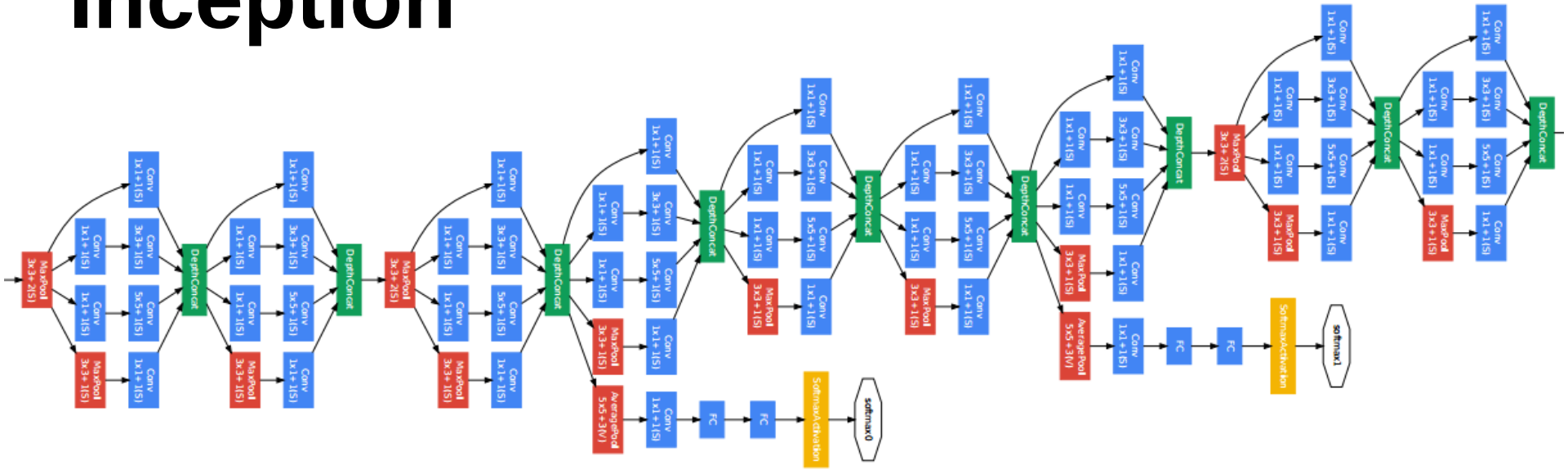
```
sequential1 output shape: (96, 54, 54)
pool0 output shape:       (96, 26, 26)
sequential2 output shape: (256, 26, 26)
pool1 output shape:       (256, 12, 12)
sequential3 output shape: (384, 12, 12)
pool2 output shape:       (384, 5, 5)
dropout0 output shape:    (384, 5, 5)
sequential4 output shape: (10, 5, 5)
pool3 output shape:       (10, 1, 1)
flatten0 output shape:    (10)
```

NIN dimension reduction

Summary

- **LeNet** (the first convolutional neural network)
- **AlexNet**
  - More of everything
  - ReLu, Dropout, Invariances
- **VGG**
  - Even more of everything (narrower and deeper)
  - Repeated blocks
- **NiN**
  - 1x1 convolutions + global pooling instead of dense

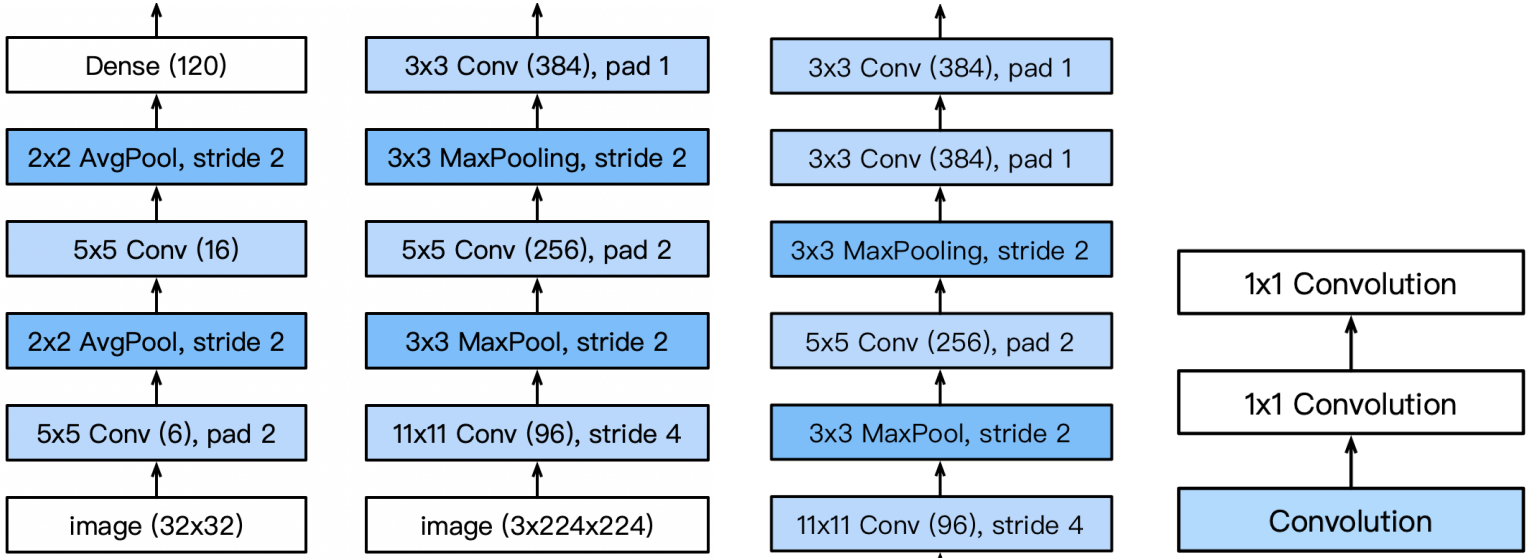
Inception


MaxPool 3x3+2(S)
Conv 1x1+1(S)
Conv 3x3+1(S)
Conv 5x5+1(S)
DepthConcat
AveragePool 5x5+3(V)
FC
SoftmaxActivation
softmax0
softmax1

Picking the best convolution …

1x1 | 3x3 | 5x5 | Max pooling | Multiple 1x1

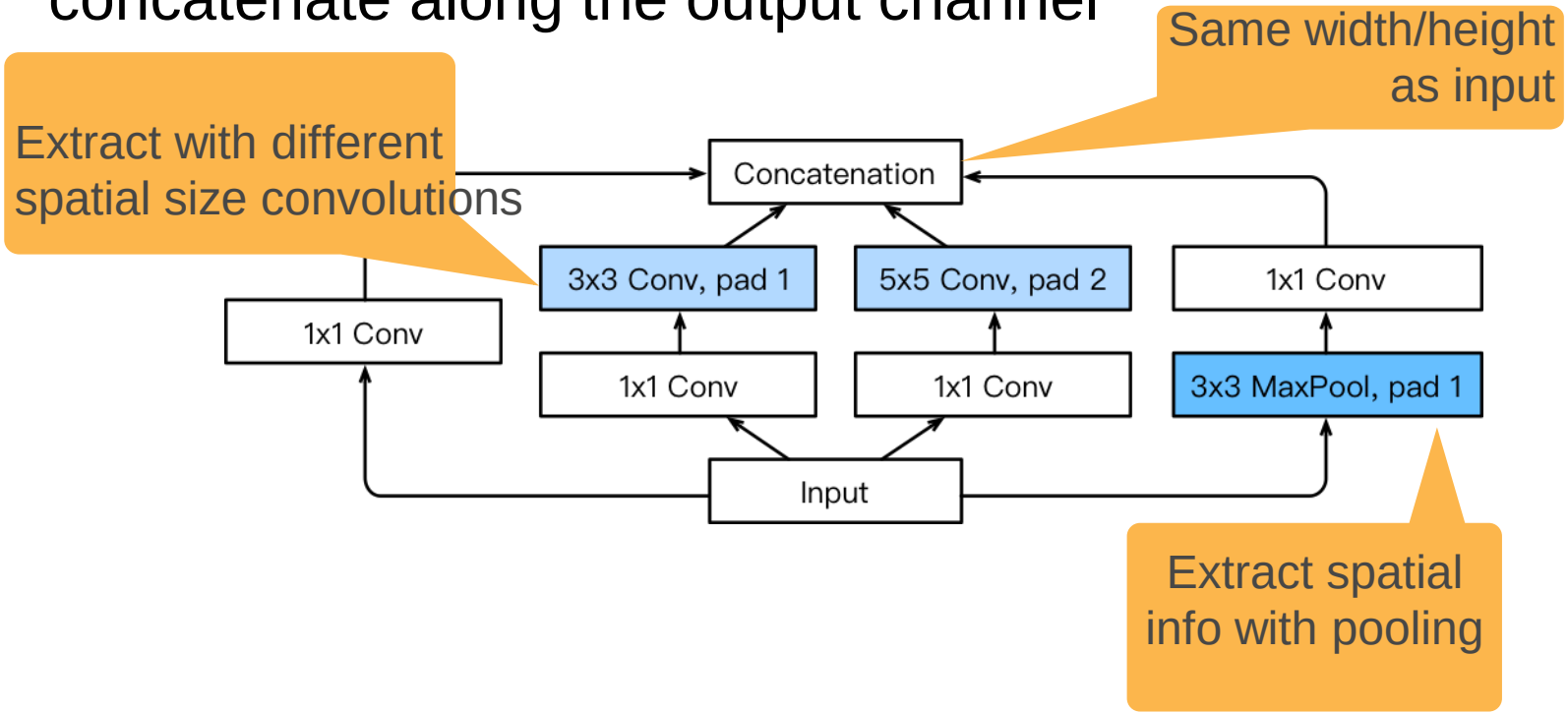
LeNet | AlexNet | VGG | NiN



Why choose? Just pick them all.

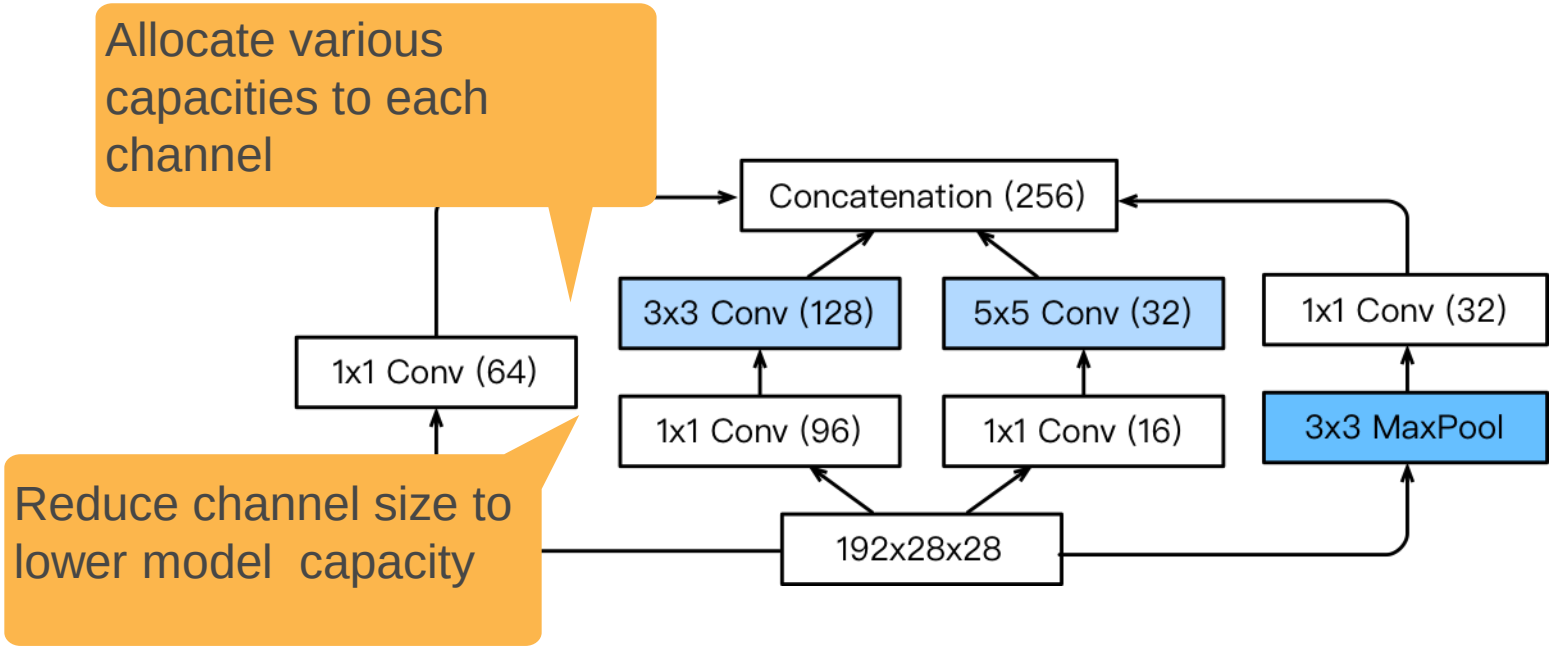
Inception Blocks

4 paths extract information from different aspects, then concatenate along the output channel



Inception Blocks

The first inception block with channel sizes specified

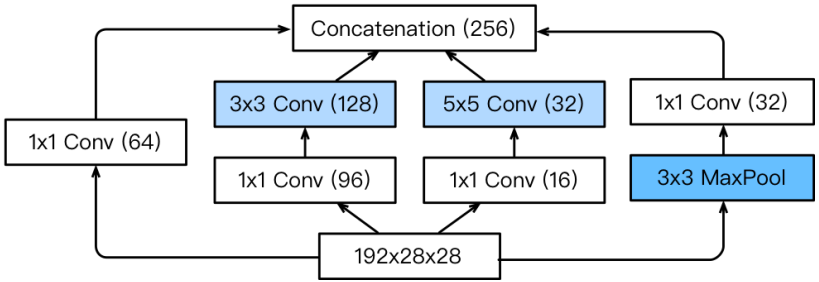


Inception Blocks

Inception blocks have fewer parameters and less computation complexity than a single 3x3 or 5x5 convolutional layer

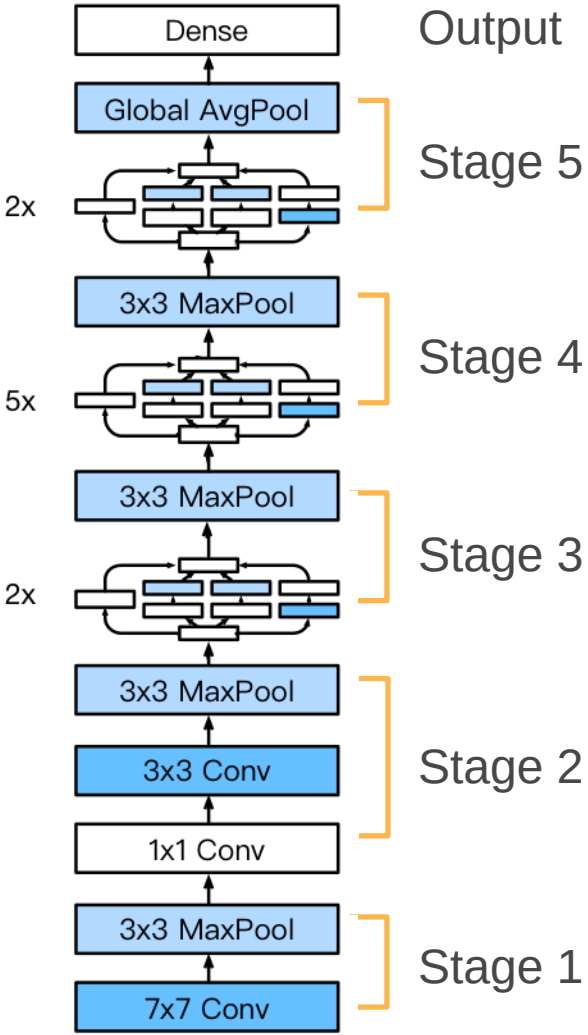
- Mix of different functions (powerful function class)
- Memory and compute efficiency (good generalization)

	#parameters	FLOPS
Inception	0.16 M	128 M
3x3 Conv	0.44 M	346 M
5x5 Conv	1.22 M	963 M



GoogLeNet

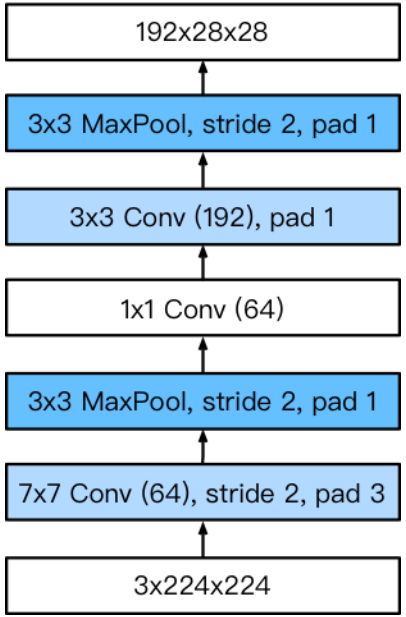
- 5 stages with 9 inceptions blocks



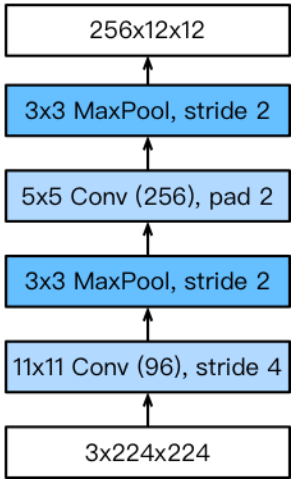
Stage 1 & 2

- Smaller kernel size and output channels due to more layers

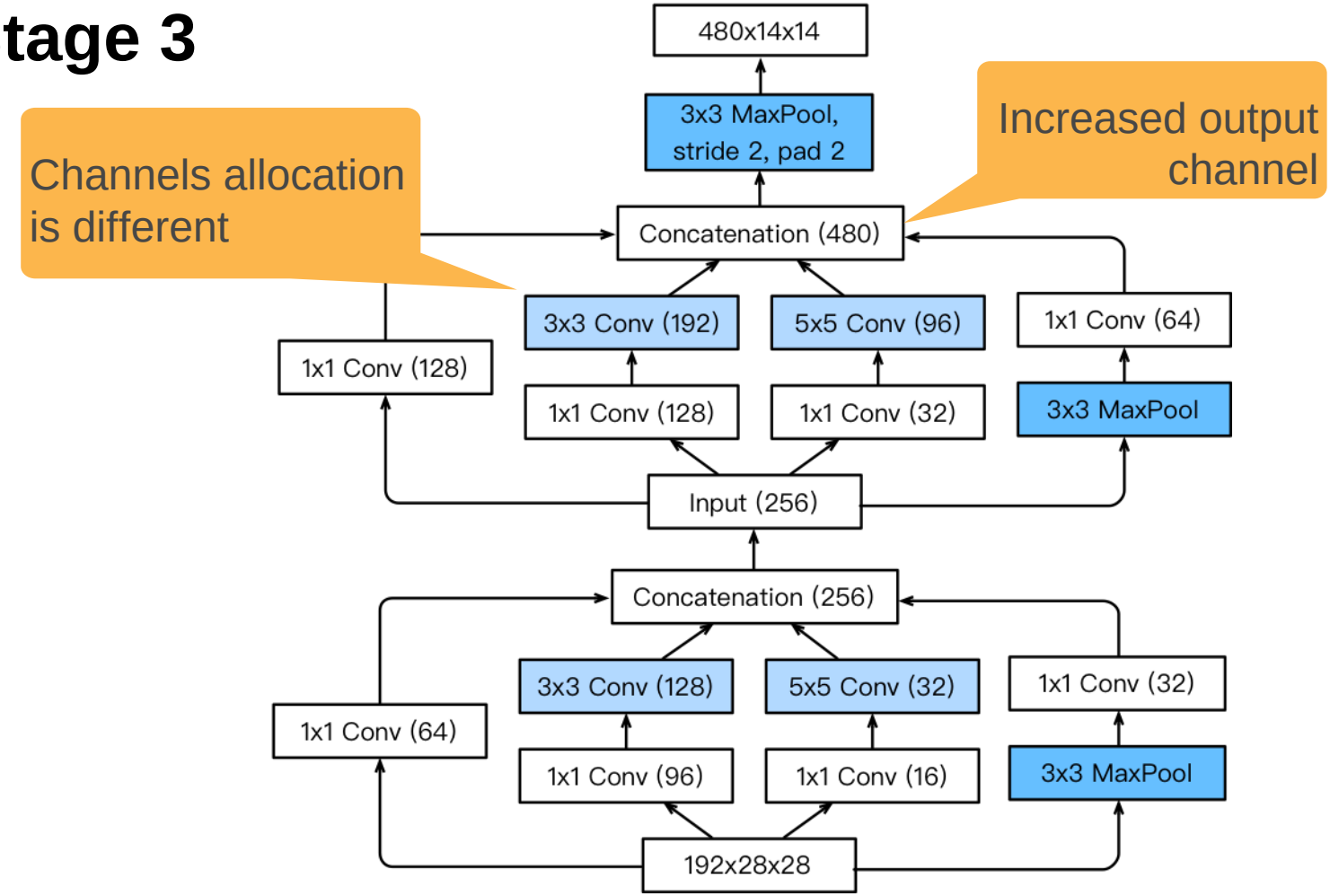
GoogLeNet



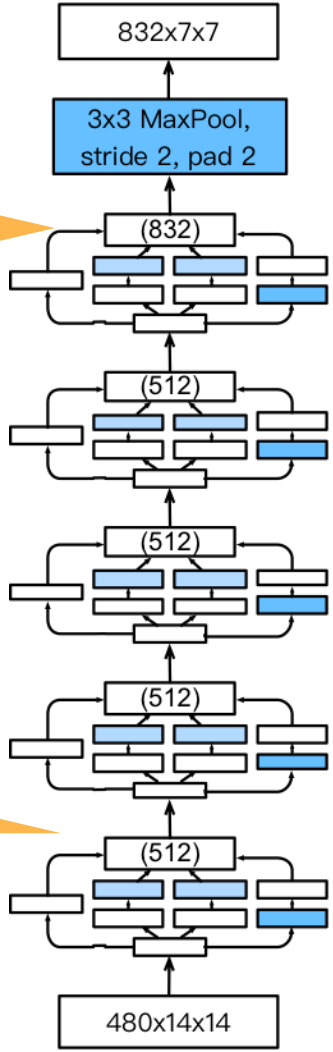
AlexNet



Stage 3

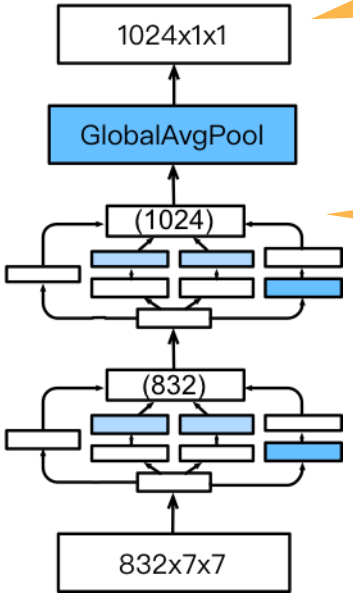

480x14x14
3x3 MaxPool, stride 2, pad 2
Increased output channel
Channels allocation is different
Concatenation (480)
3x3 Conv (192)
5x5 Conv (96)
1x1 Conv (64)
1x1 Conv (128)
1x1 Conv (128)
1x1 Conv (32)
3x3 MaxPool
Input (256)
Concatenation (256)
3x3 Conv (128)
5x5 Conv (32)
1x1 Conv (32)
1x1 Conv (64)
1x1 Conv (96)
1x1 Conv (16)
3x3 MaxPool
192x28x28

Stage 4 & 5



Increased output channel

Increased output channel



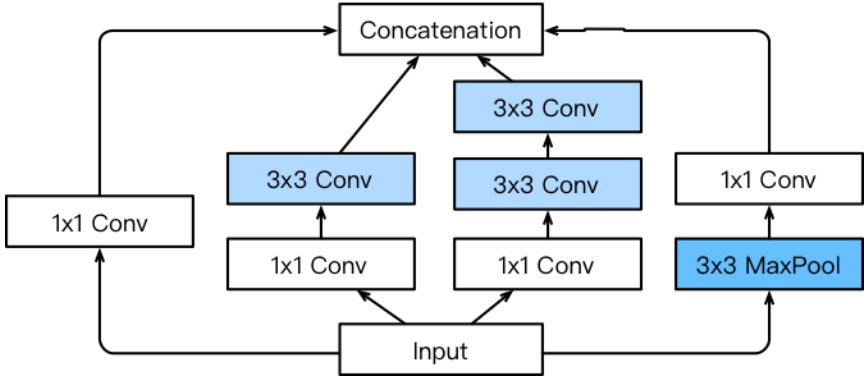
1024-dim feature to output layer

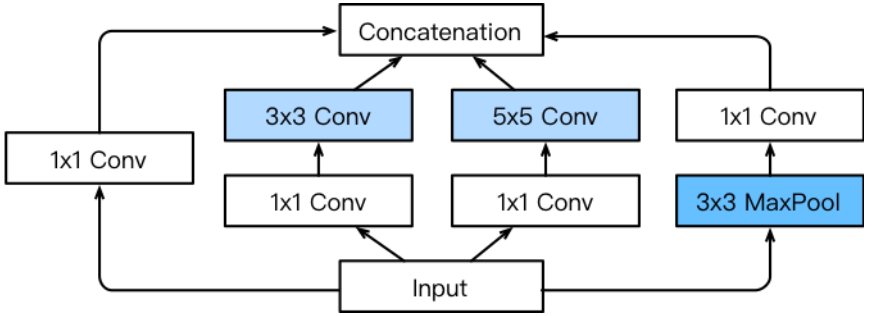
Increased output channel

The many flavors of Inception Networks

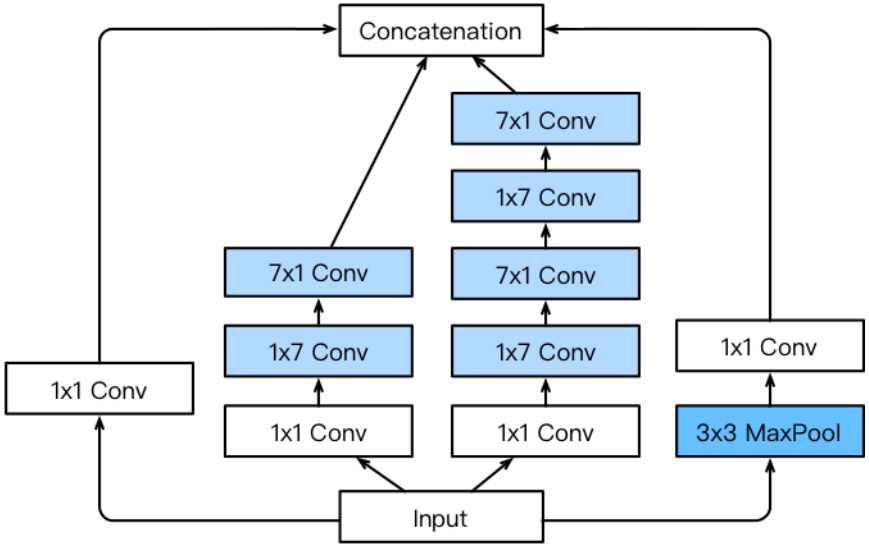
- Inception-BN (v2) - Add batch normalization
- Inception-V3 - Modified the inception block
  - Replace 5x5 by multiple 3x3 convolutions
  - Replace 5x5 by 1x7 and 7x1 convolutions
  - Replace 3x3 by 1x3 and 3x1 convolutions
  - Generally deeper stack
- Inception-V4 - Add residual connections (more later)

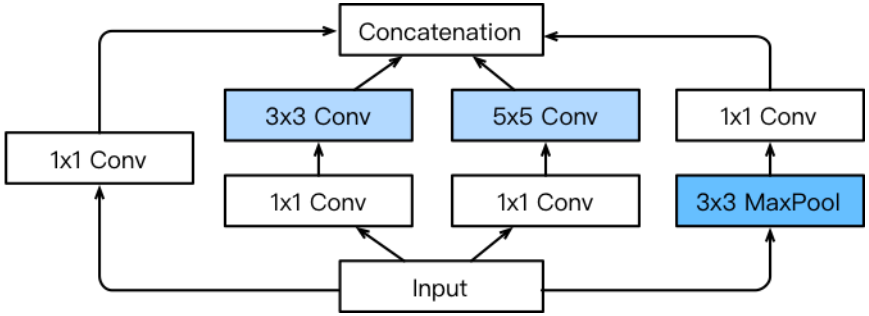
Inception V3 Block for Stage 3



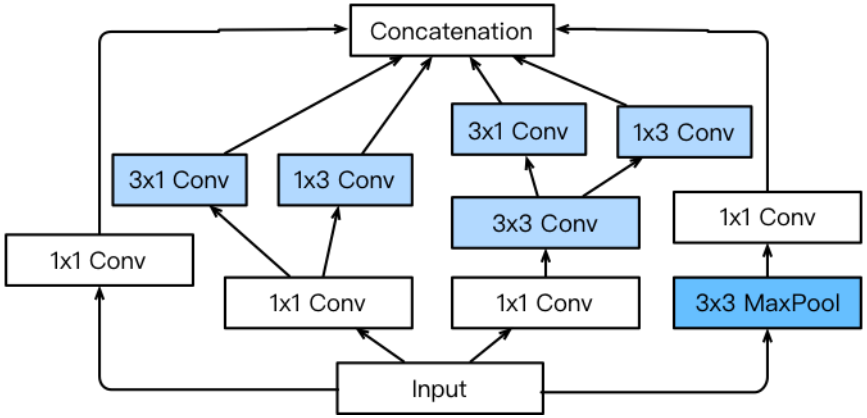


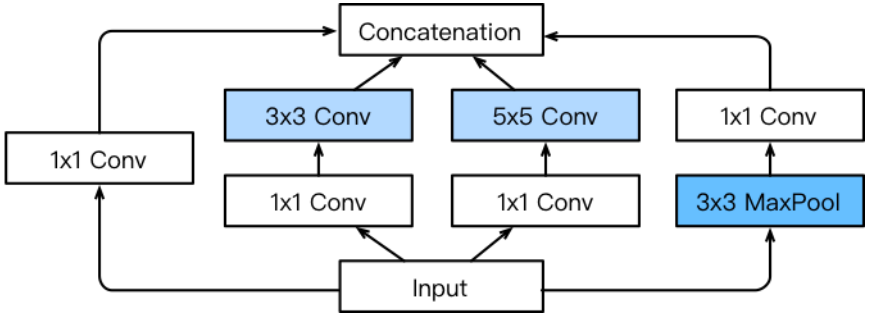
Inception V3 Block for Stage 4

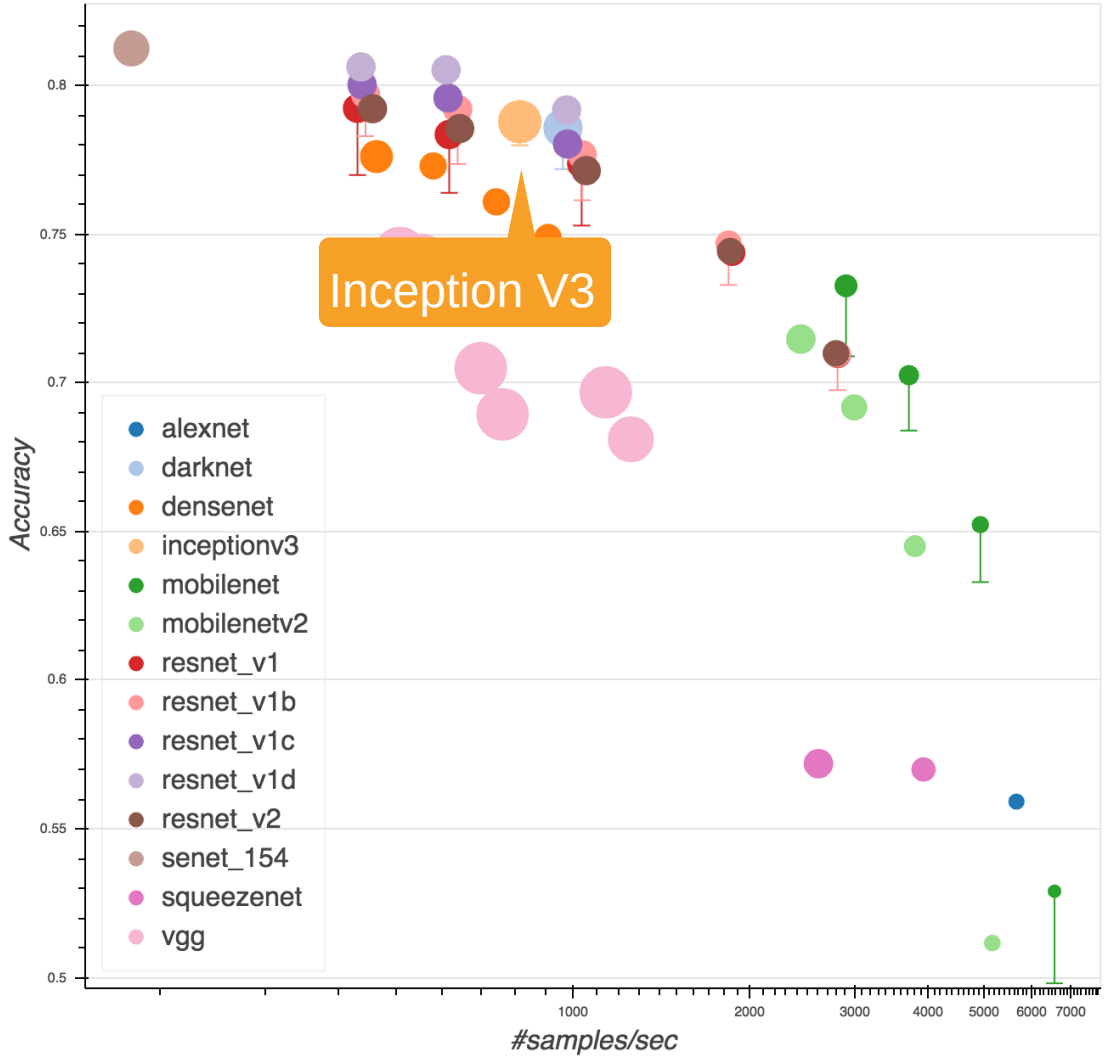




Inception V3 Block for Stage 5



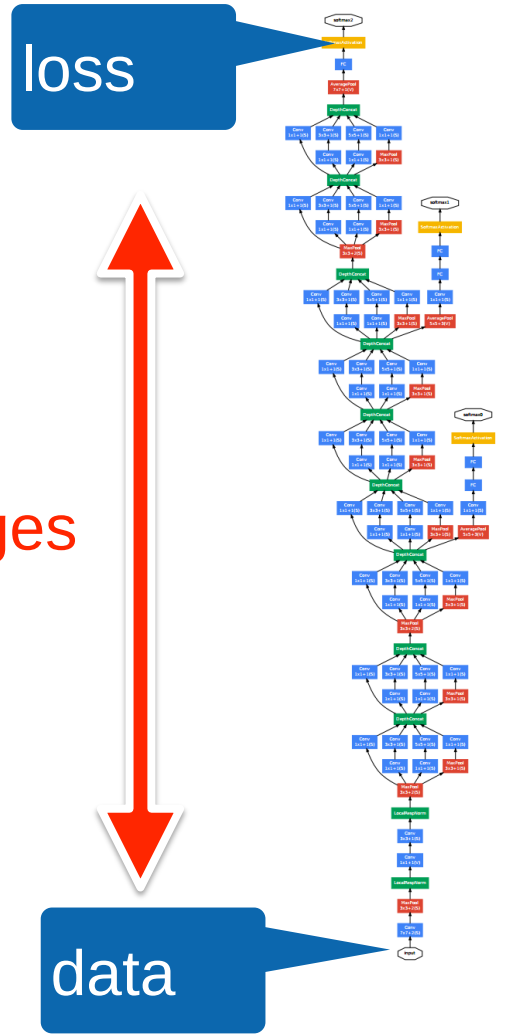



Inception V3
alexnet
darknet
densenet
inceptionv3
mobilenet
mobilenetv2
resnet_v1
resnet_v1b
resnet_v1c
resnet_v1d
resnet_v2
senet_154
squeezenet
vgg
Accuracy
0.8
0.75
0.7
0.65
0.6
0.55
0.5
1000
2000
3000
4000
5000
6000
7000
#samples/sec

Batch Normalization

- Loss occurs at last layer
  - Last layers learn quickly
- Data is inserted at bottom layer
  - Bottom layers change - **everything** changes
  - Last layers need to relearn many times
  - Slow convergence
- This is like covariate shift

  Can we avoid changing last layers while learning first layers?



Batch Normalization

- Can we avoid changing last layers while learning first layers?
- Fix mean and variance

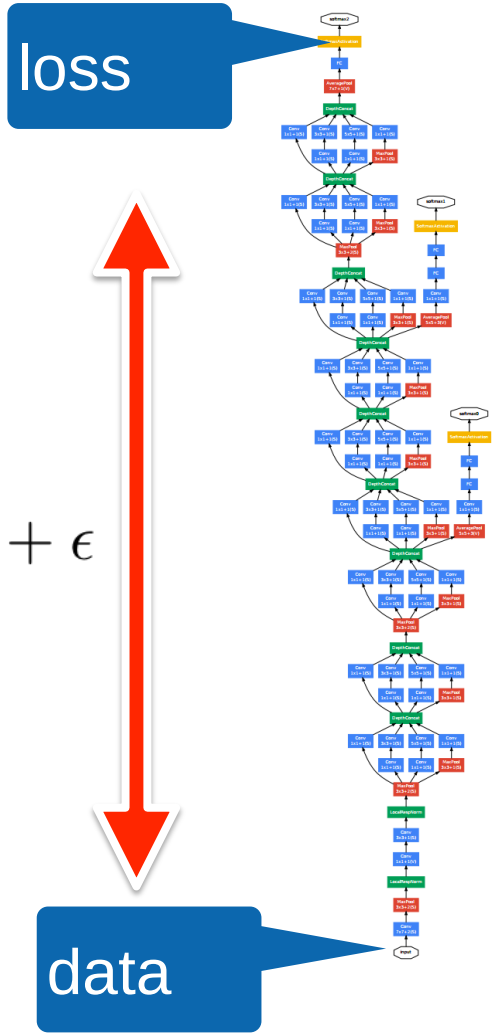
$$\mu_B = \frac{1}{|B|}\sum_{i\in B} x_i \text{ and } \sigma_B^2 = \frac{1}{|B|}\sum_{i\in B}(x_i - \mu_B)^2 + \epsilon$$

and adjust it separately

$$x_{i+1} = \gamma\frac{x_i - \mu_B}{\sigma_B} + \beta$$

mean

variance



This was the original motivation …

What Batch Norms really do

- Doesn't really reduce covariate shift (Lipton et al., 2018)
- Regularization by noise injection

$$x_{i+1} = \gamma \frac{x_i - \hat{\mu}_B}{\hat{\sigma}_B} + \beta$$

Empirical mean

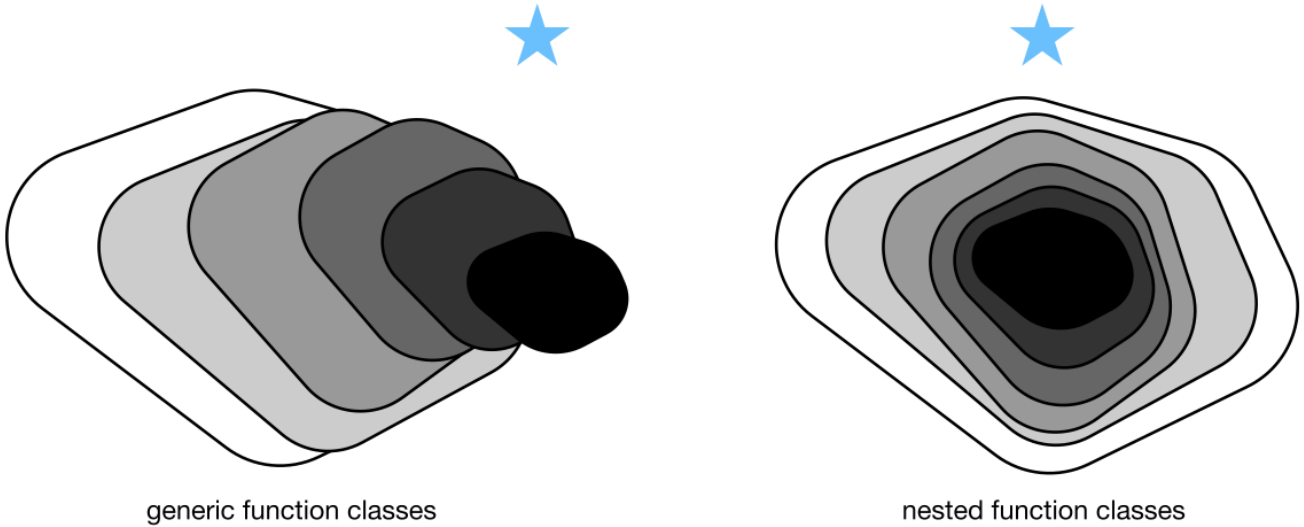
Empirical variance

  - Random shift per minibatch
  - Random scale per minibatch
- No need to mix with dropout (both are capacity control)
- Ideal minibatch size of 64 to 256

Details

- **Dense Layer**
  One normalization for all
- **Convolution**
  One normalization per channel
- Compute **new mean and variance** for every minibatch
  - Effectively acts as regularization
  - Optimal minibatch size is ~128
    (watch out for parallel training with many machines)

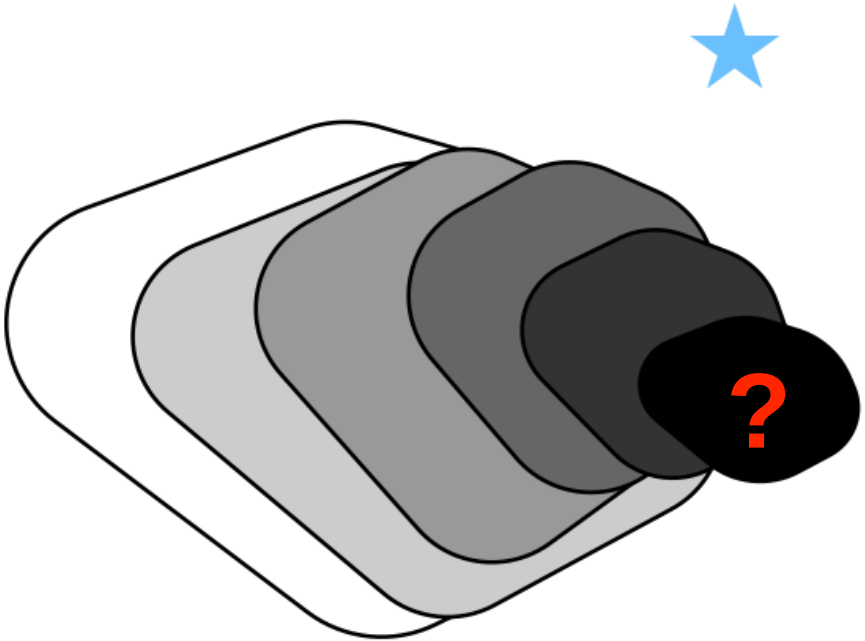
Residual Networks



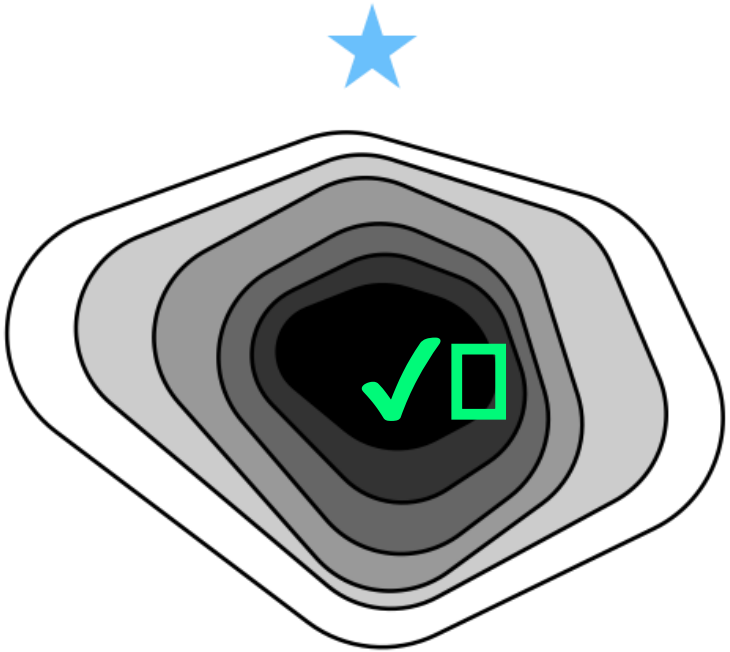
generic function classes

nested function classes

Does adding layers improve accuracy?



generic function classes

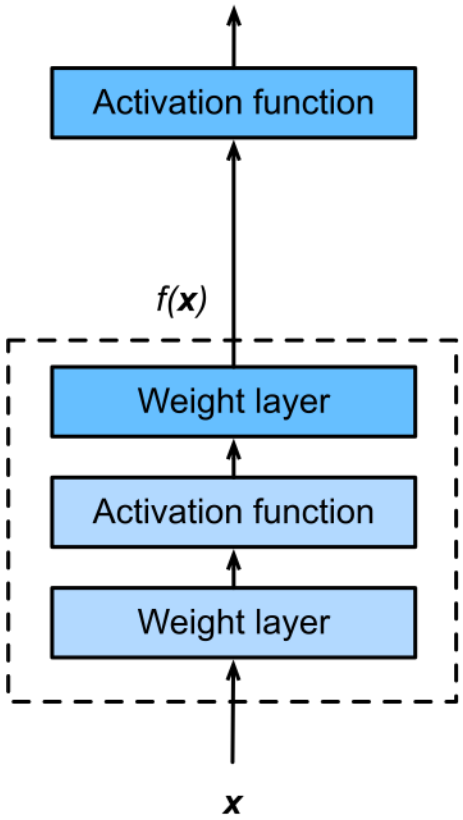


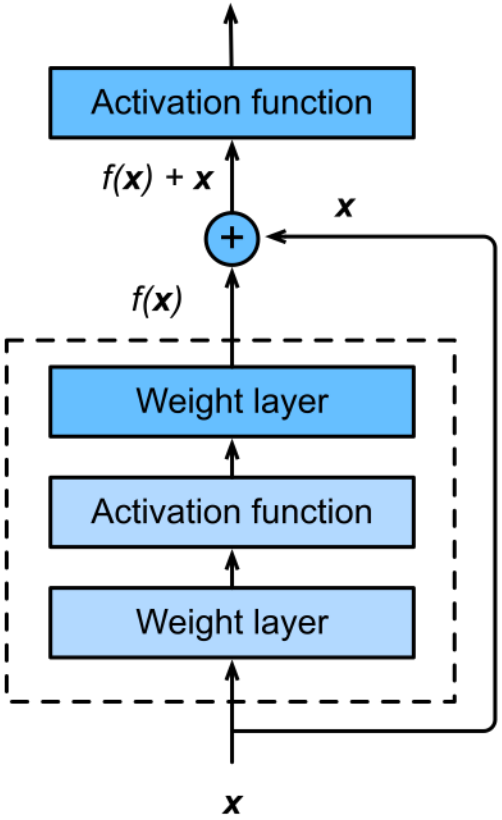
nested function classes

Residual Networks

- Adding a layer **changes** function class
- We want to **add to** the function class
- 'Taylor expansion' style parametrization

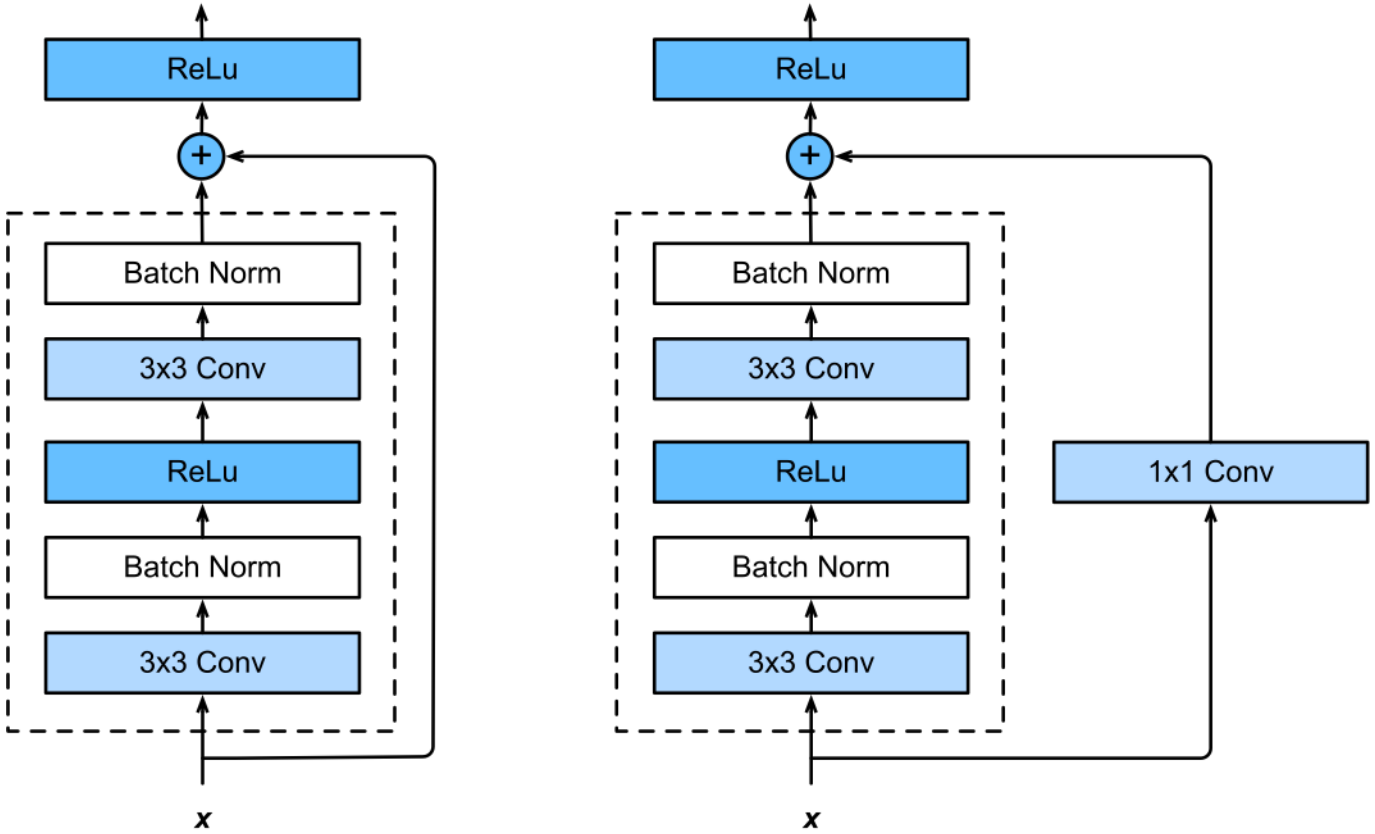
$$f(x) + x$$



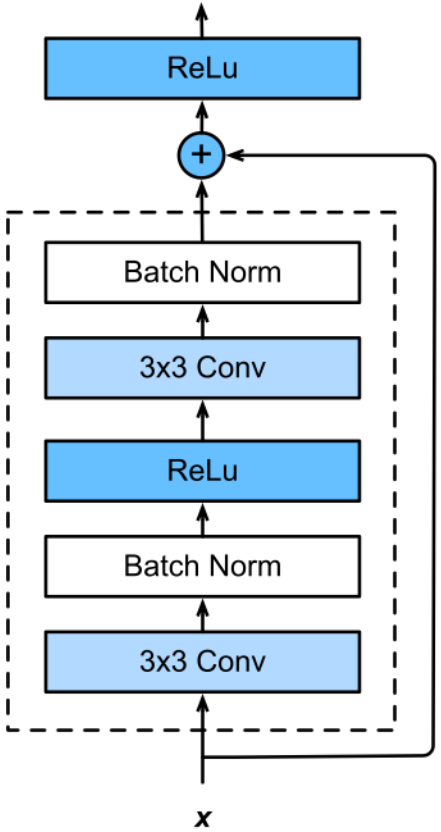


He et al., 2015

ResNet Block in detail

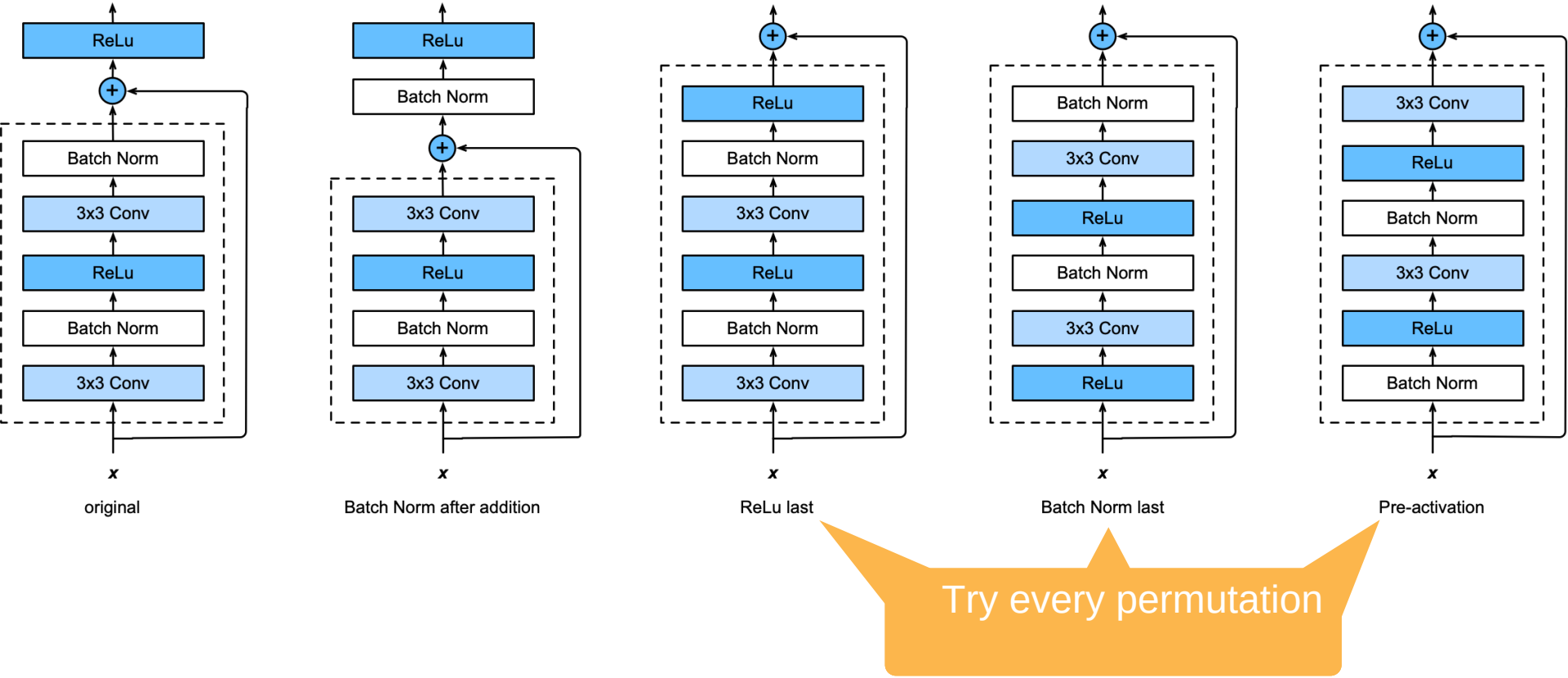


In code

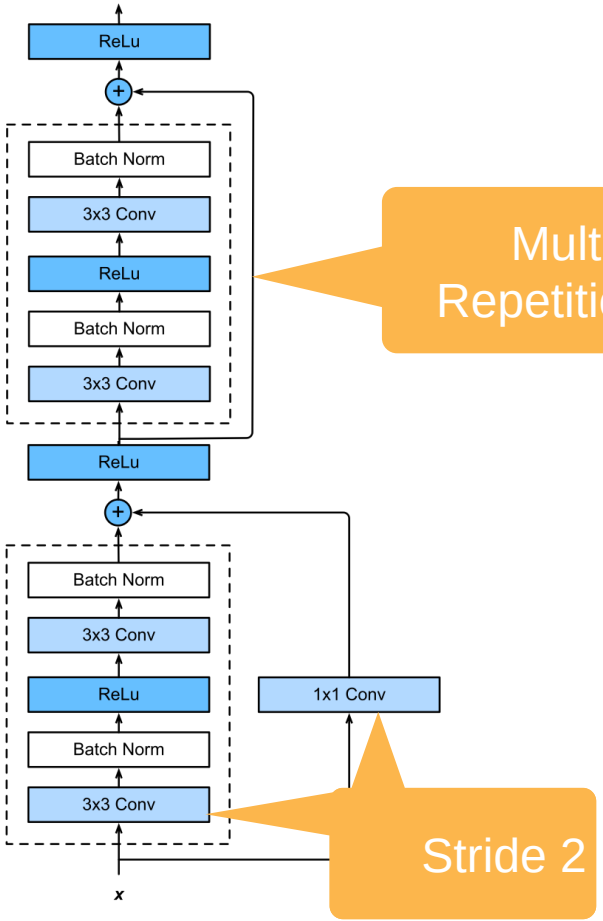



```
def forward(self, X):
    Y = self.bn1(self.conv1(X))
    Y = nd.relu(Y)
    Y = self.bn2(self.conv2(Y))
    if self.conv3:
        X = self.conv3(X)
    return nd.relu(Y + X)
```

The many flavors of ResNet blocks



ResNet Module



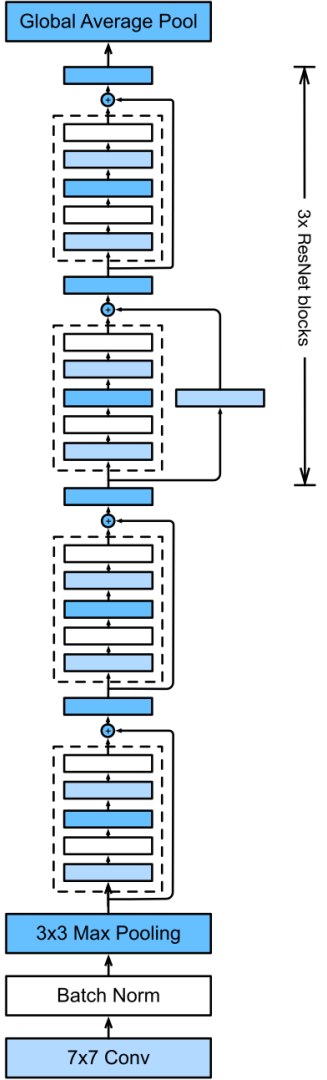
- Downsample per module (stride=2)
- Enforce some nontrivial nonlinearity per module (via 1x1 convolution)
- Stack up in blocks

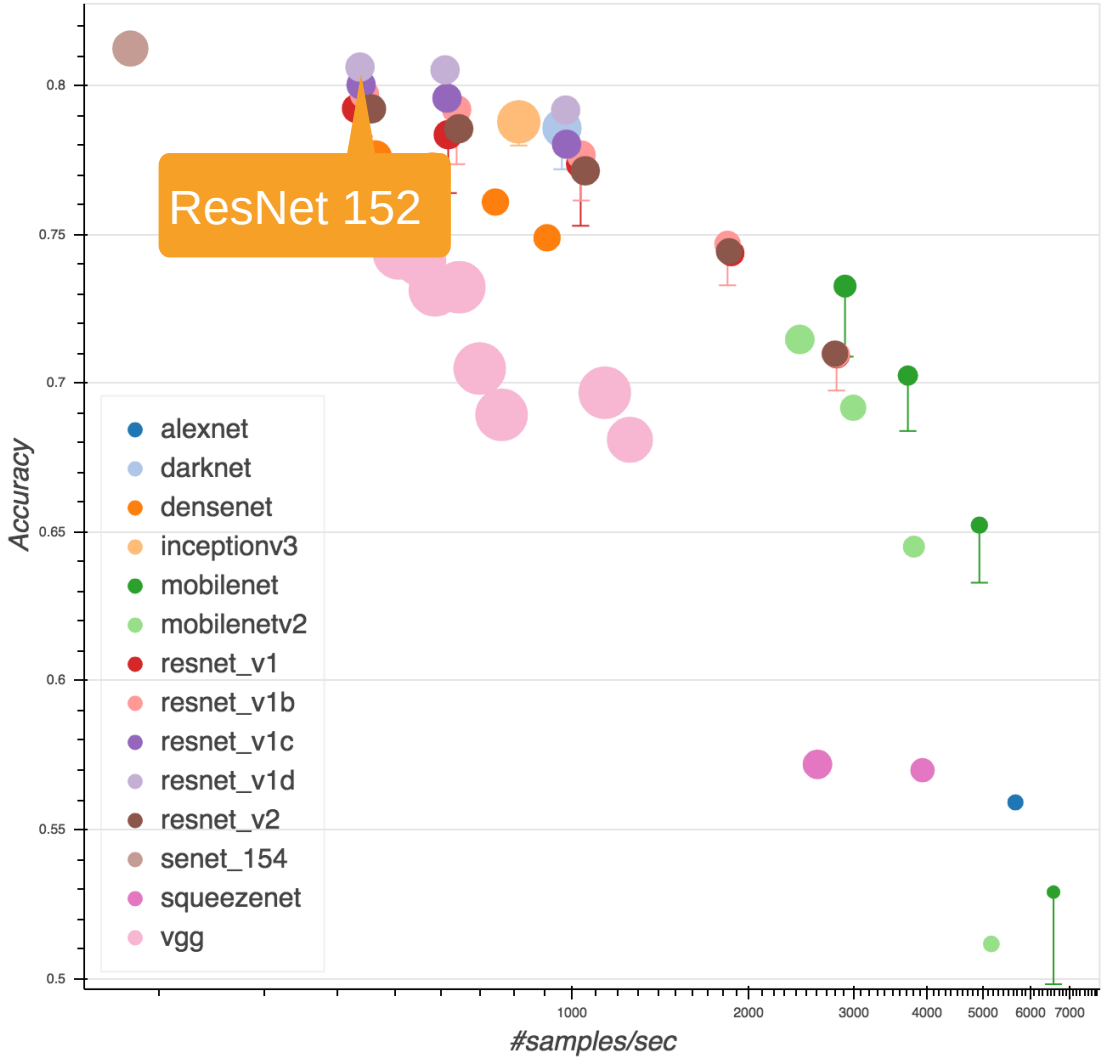
```
blk = nn.Sequential()
for i in range(num_residuals):
    if i == 0 and not first_block:
        blk.add(Residual(num_channels,
            use_1x1conv=True, strides=2))
    else:
        blk.add(Residual(num_channels))
```

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 …

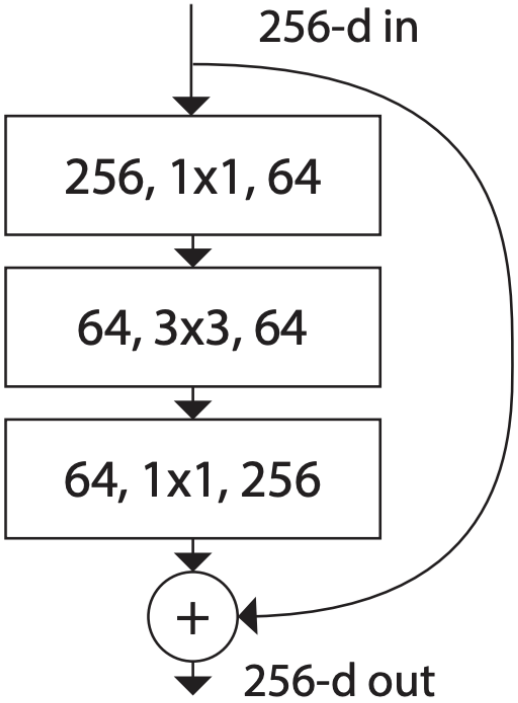


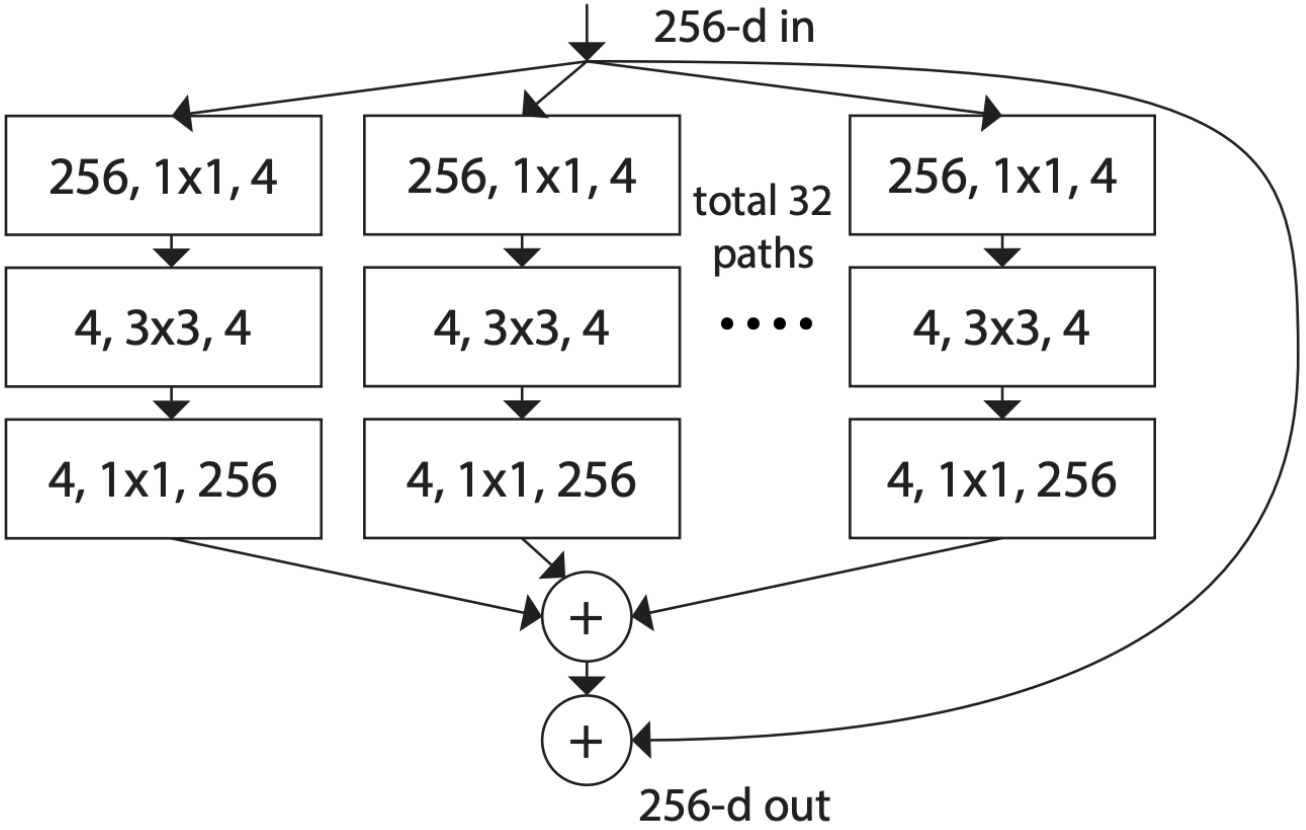


GluonCV Model Zoo

https://gluon-cv.mxnet.io/model_zoo/classification.html

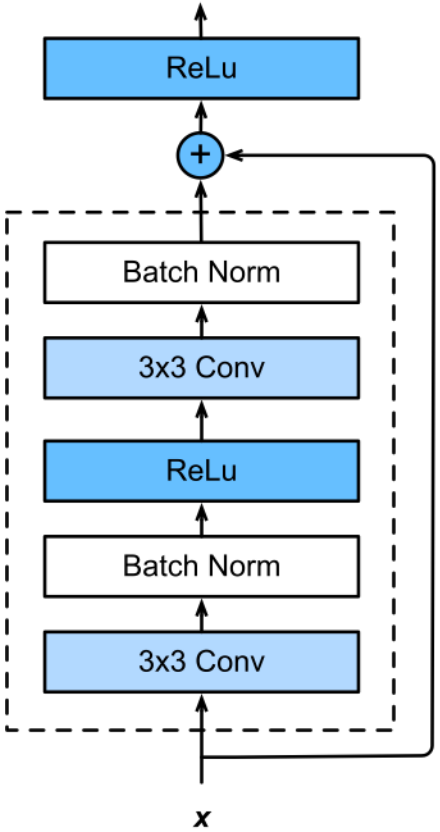
ResNext





Xie et al., 2016

Reducing the cost of Convolutions



- **Parameters**

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

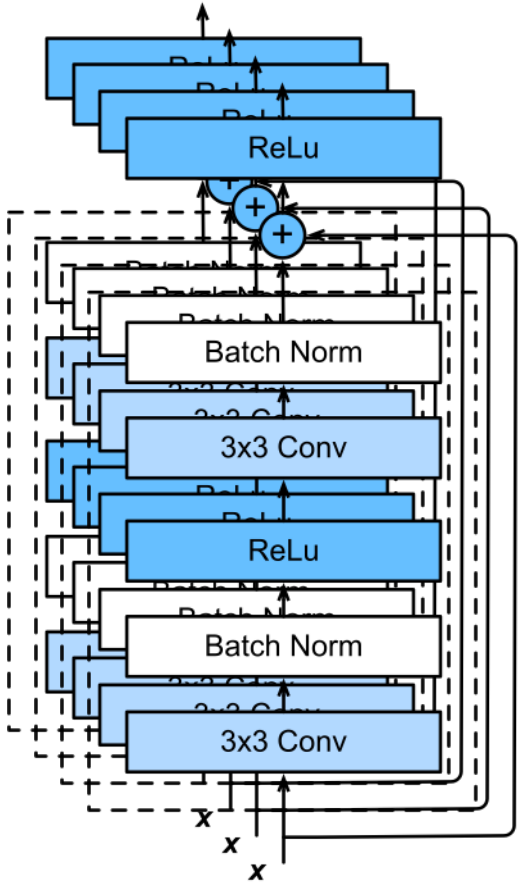
- **Computation**

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

- **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)

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

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**×10^6	**25.0**×10^6
FLOPs		**4.1**×10^9	**4.2**×10^9

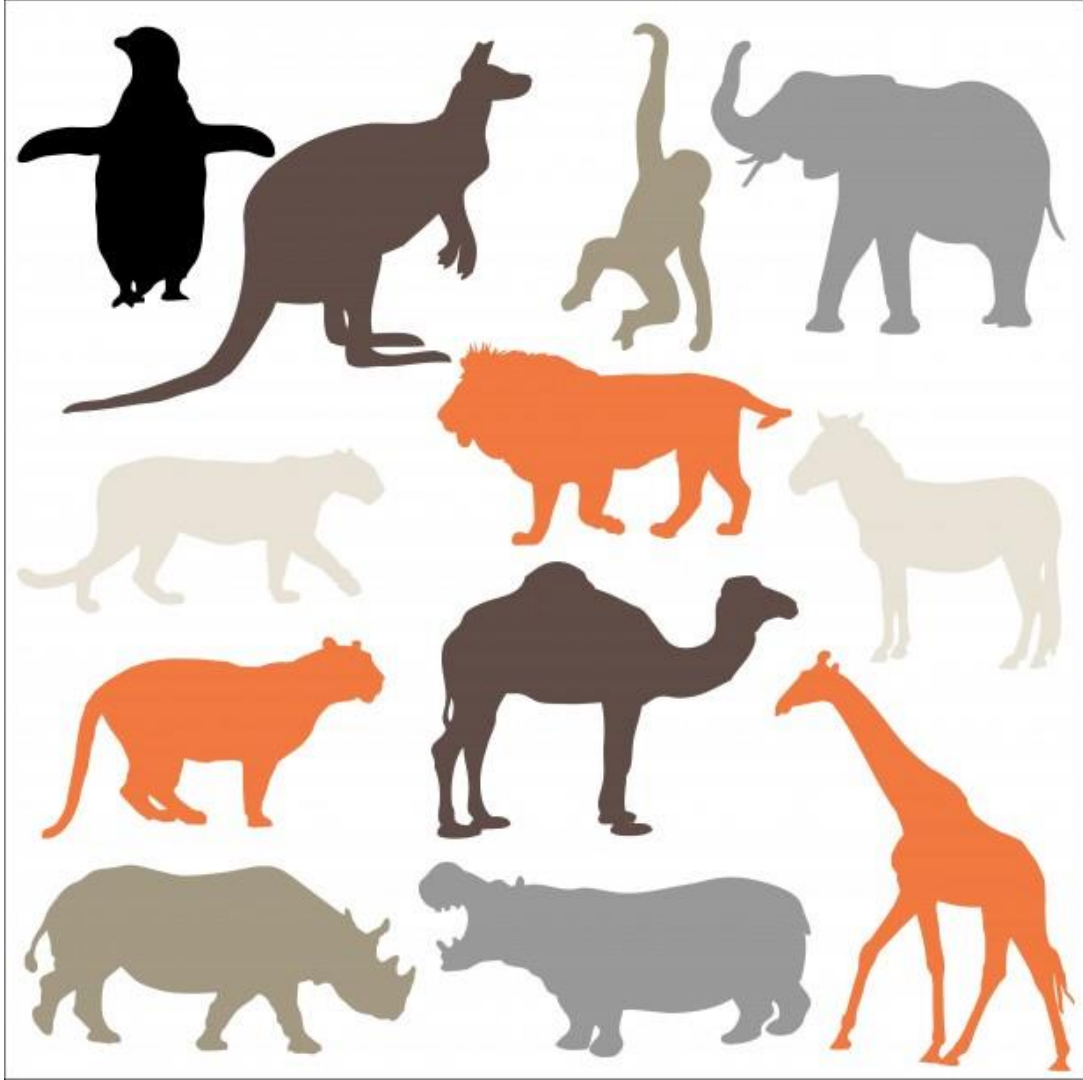
RexNext budget

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

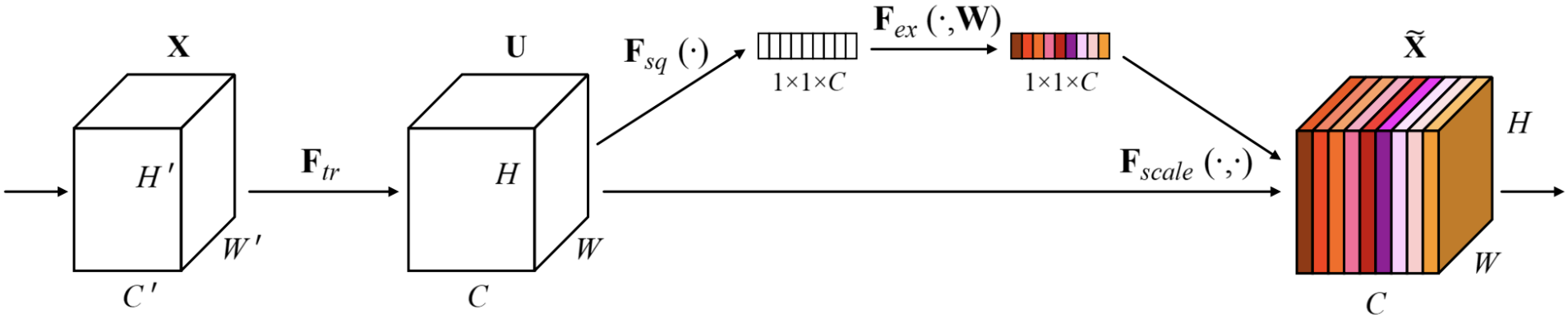
```
nn.Conv2D(group_width=width,
      kernel_size=3,
      groups=cardinality)
```

Xie et al., 2016

More Ideas

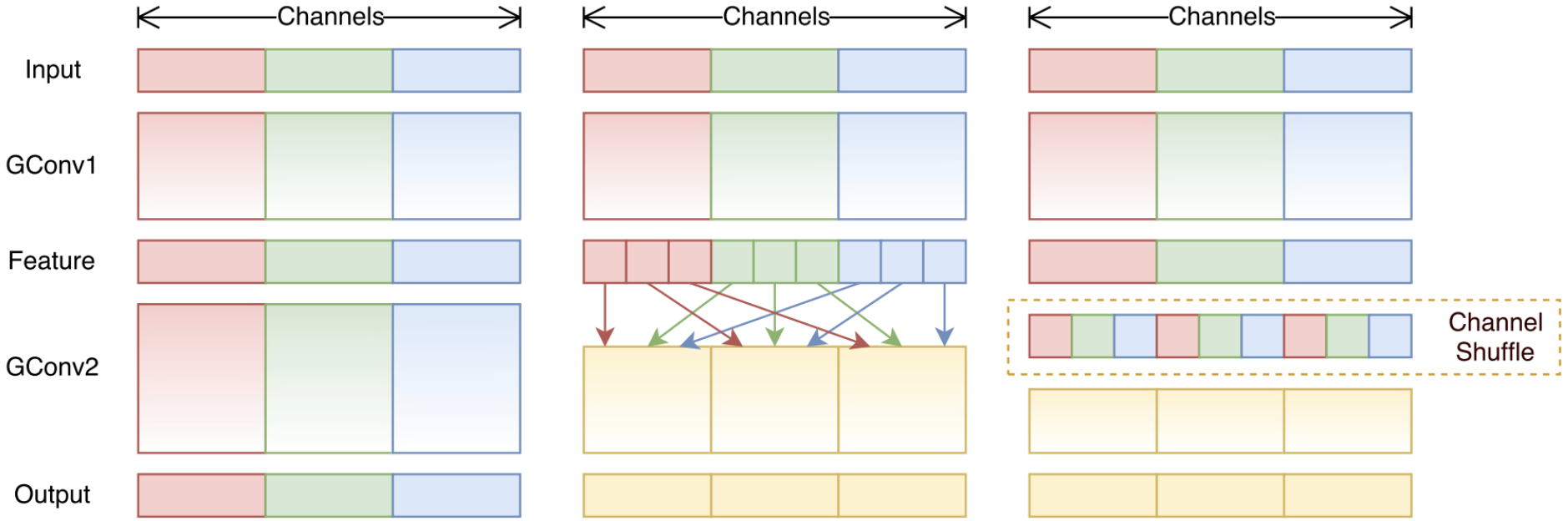


Squeeze-Excite Net (Hu et al., 2017)



- Learn global weighting function per channel
- Allows for fast information transfer between pixels in different locations of the image

ShuffleNet (Zhang et al., 2018)

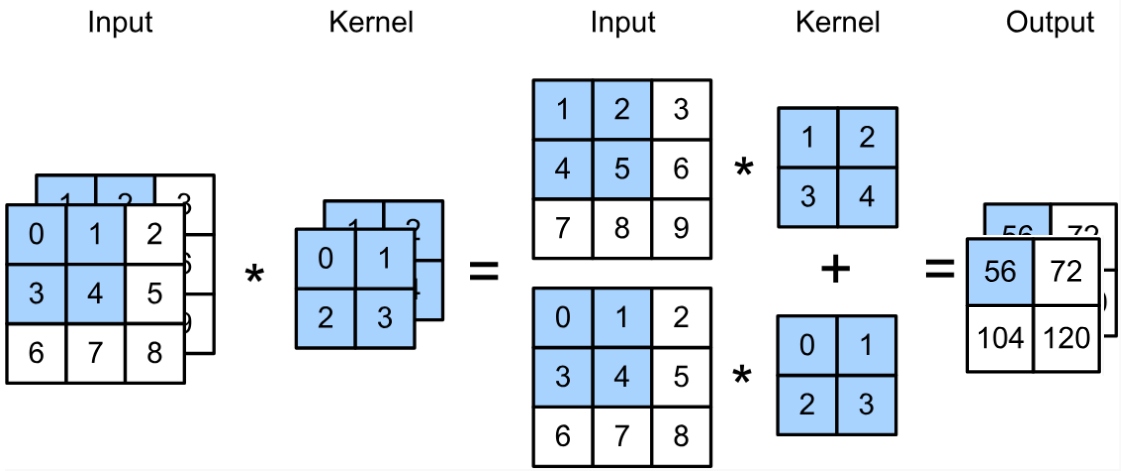


- ResNext breaks convolution into channels
- ShuffleNet mixes by grouping (very efficient for mobile)

Separable Convolutions - all channels separate

- **Parameters** $k_h \cdot k_w \cdot c_i \cdot c_o$
- **Computation** $m_h \cdot m_w \cdot k_h \cdot k_w \cdot c_i \cdot c_o$
- **Break up channels** to the extreme
  No mixing between channels

$m_h \cdot m_w \cdot k_h \cdot k_w \cdot c$



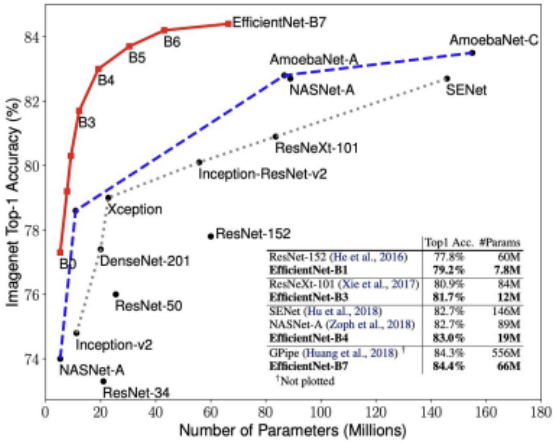
EfficientNet

EfficientNet: Smart Compound Scaling

[Tan and Le. 2019]

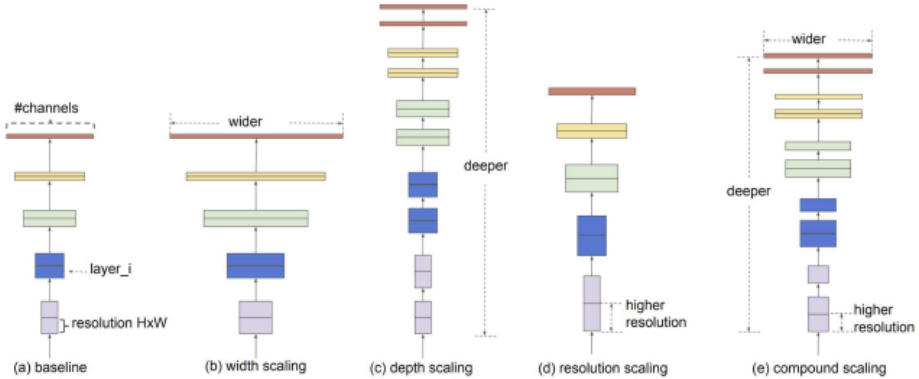
- Increase network capacity by scaling width, depth, and resolution, while balancing accuracy and efficiency.
- Search for optimal set of compound scaling factors given a compute budget (target memory & flops).
- Scale up using smart heuristic rules

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



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**79.2%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.7%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**83.0%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

†Not plotted



Credit: CS231n @ Stanford

Summary

- **Inception**
  - Inhomogeneous mix of convolutions (varying depth)
  - Batch norm regularization
- **ResNet**
  - Taylor expansion of functions
  - **ResNext** decomposes convolutions
- **Zoo**
  DenseNet, ShuffleNet, Separable Convolutions, EfficientNet