Introduction to Deep Learning

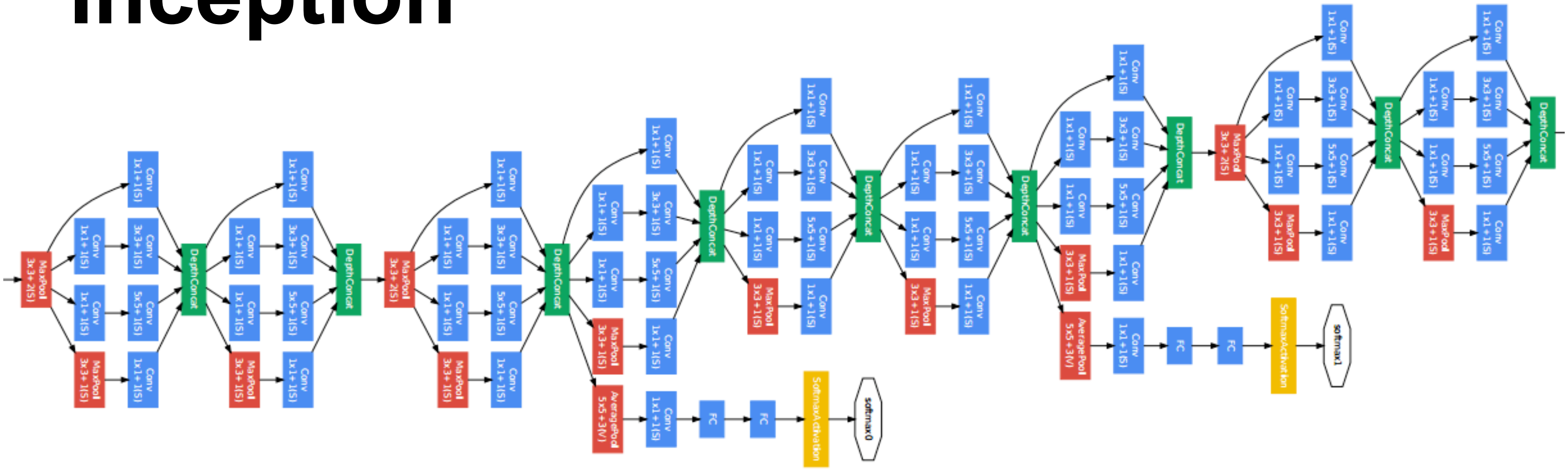
13. Inception, Batch Normalization, ResNet, DenseNet

STAT 157, Spring 2019, UC Berkeley

Alex Smola and Mu Li

courses.d2l.ai/berkeley-stat-157

Inception



Picking the best convolution …

1x1

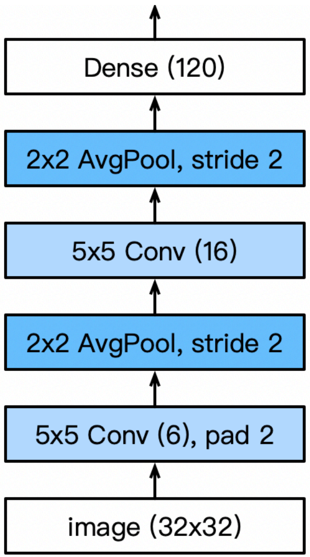
3x3

5x5

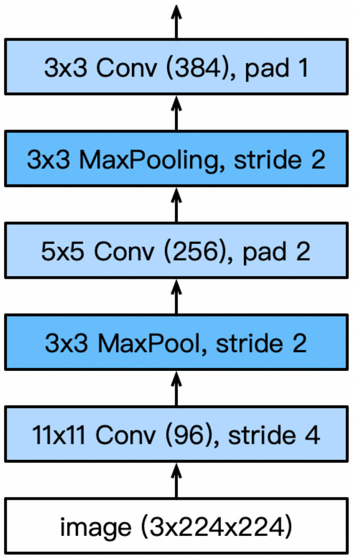
Max pooling

Multiple 1x1

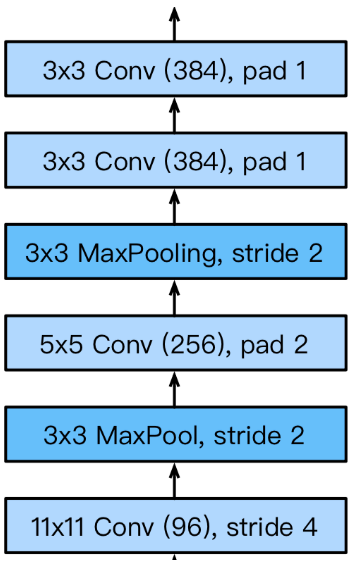
LeNet



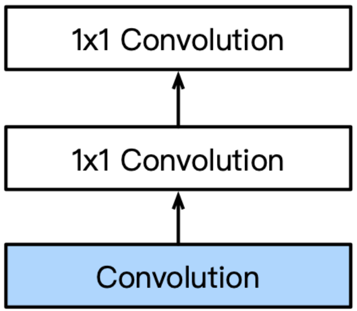
AlexNet



VGG



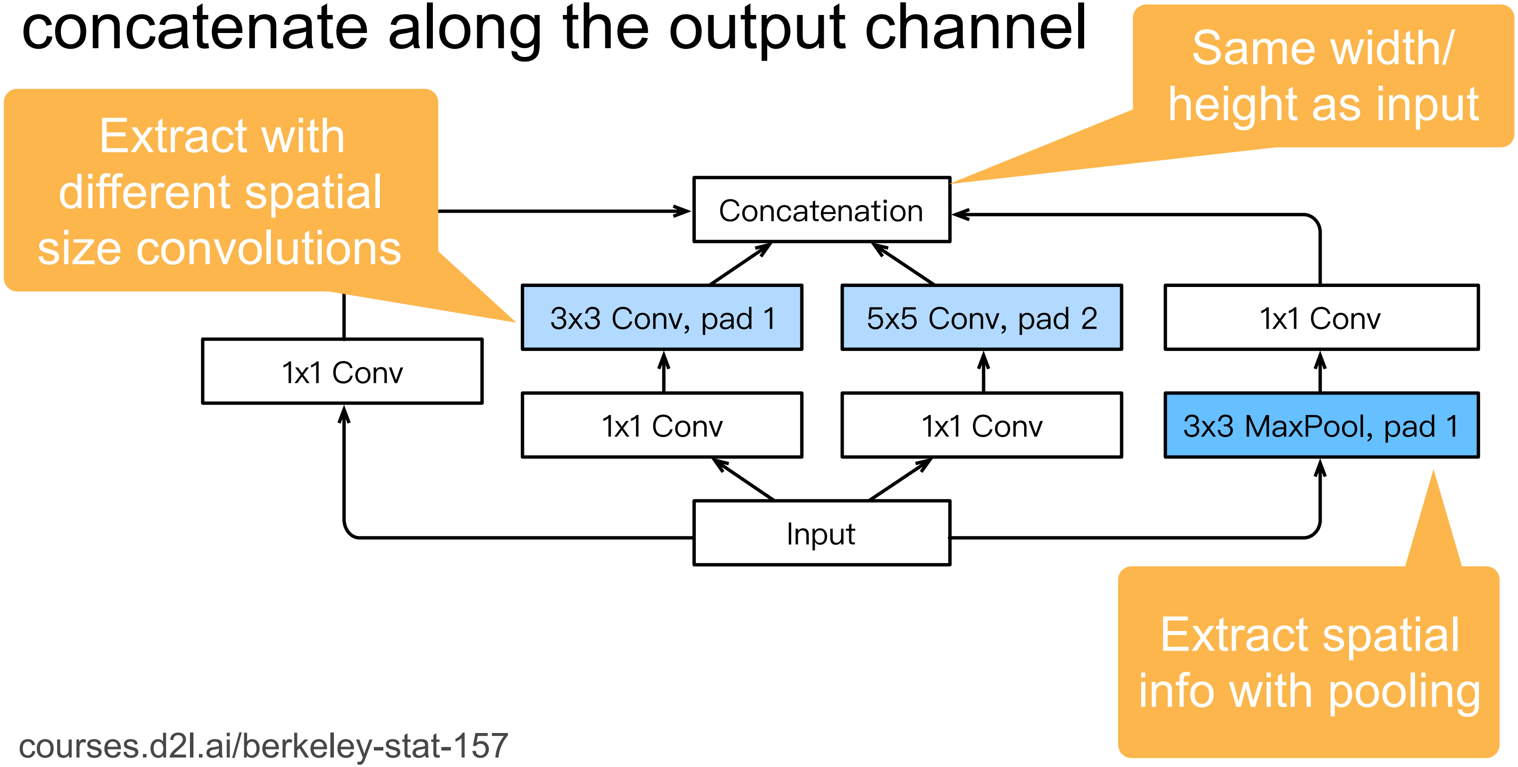
NiN



Why choose? Just pick them all.

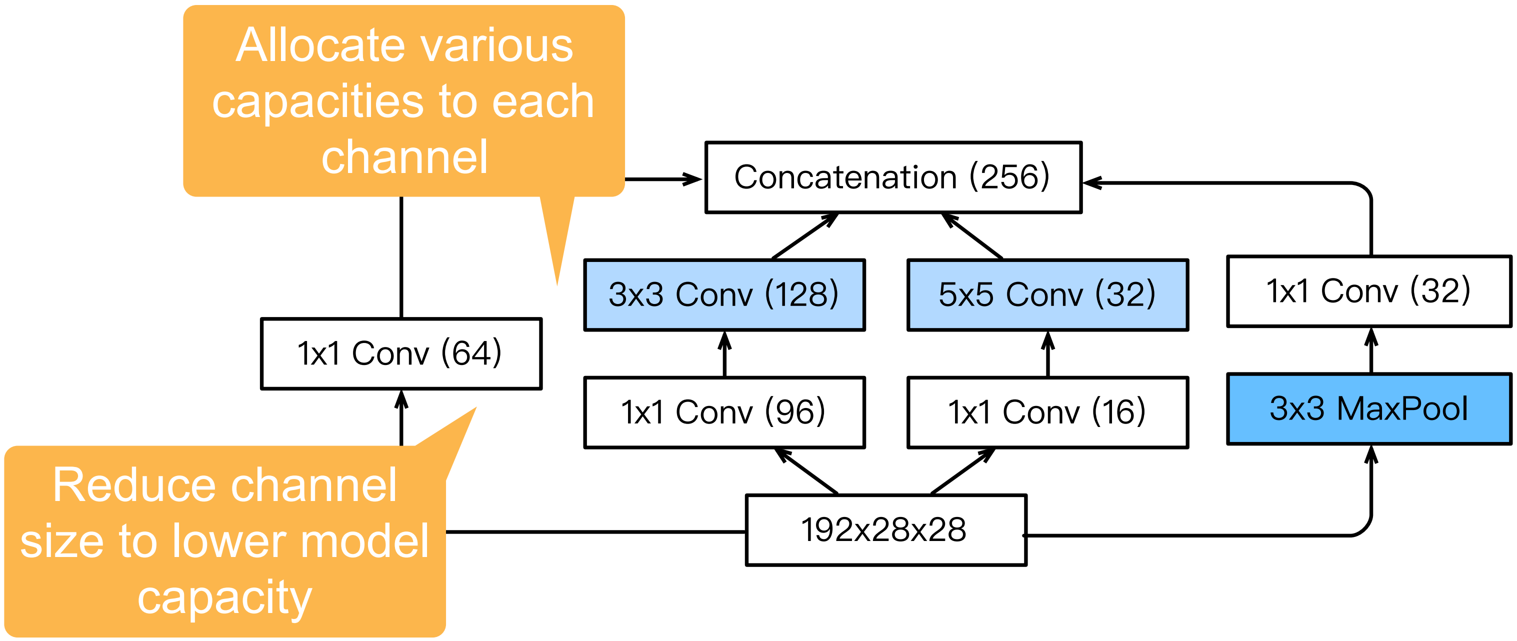
Inception Blocks

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



Inception Blocks

The first inception block with channel sizes specified

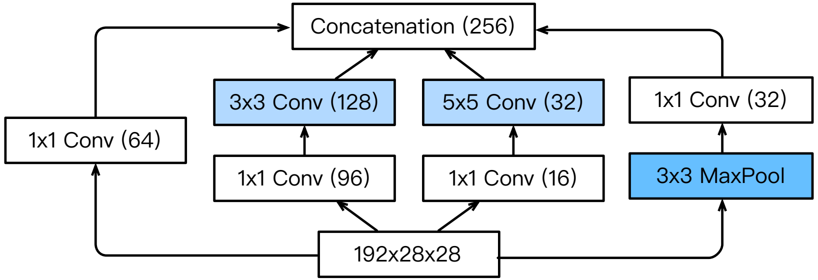


Inception Blocks

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

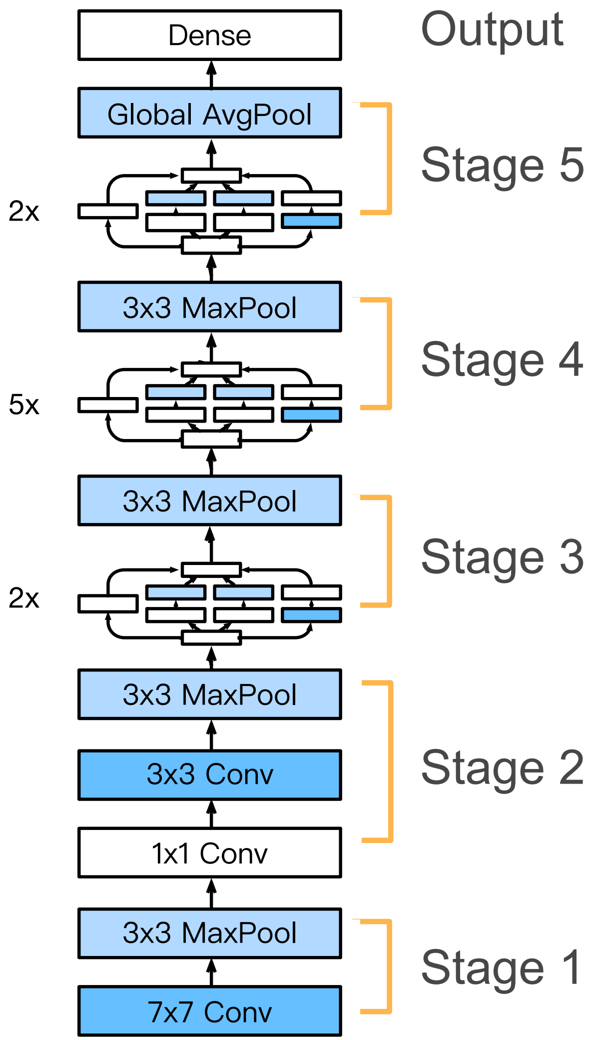
- Mix of different functions (powerful function class)
- Memory and compute efficiency (good generalization)

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



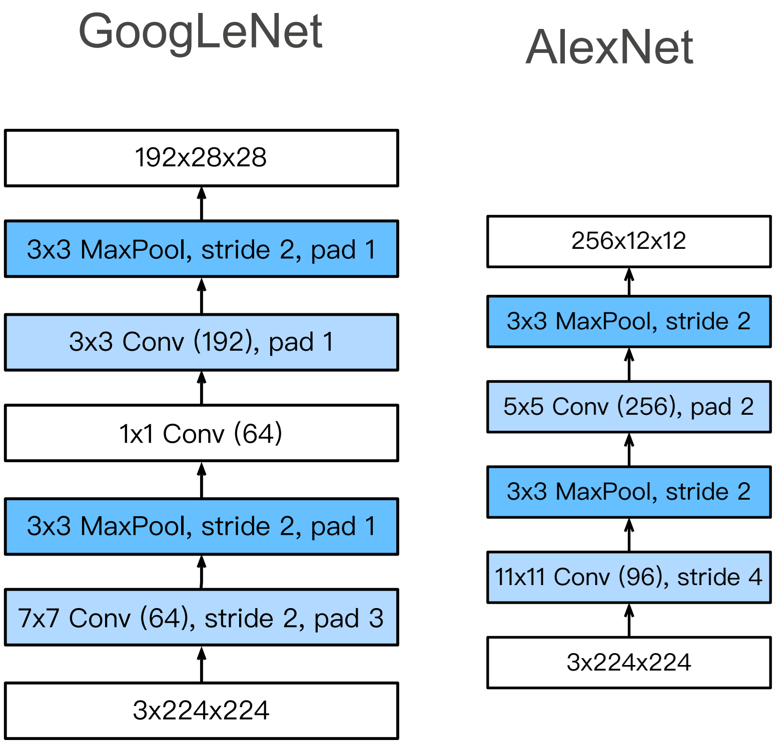
GoogLeNet

- 5 stages with 9 inceptions blocks

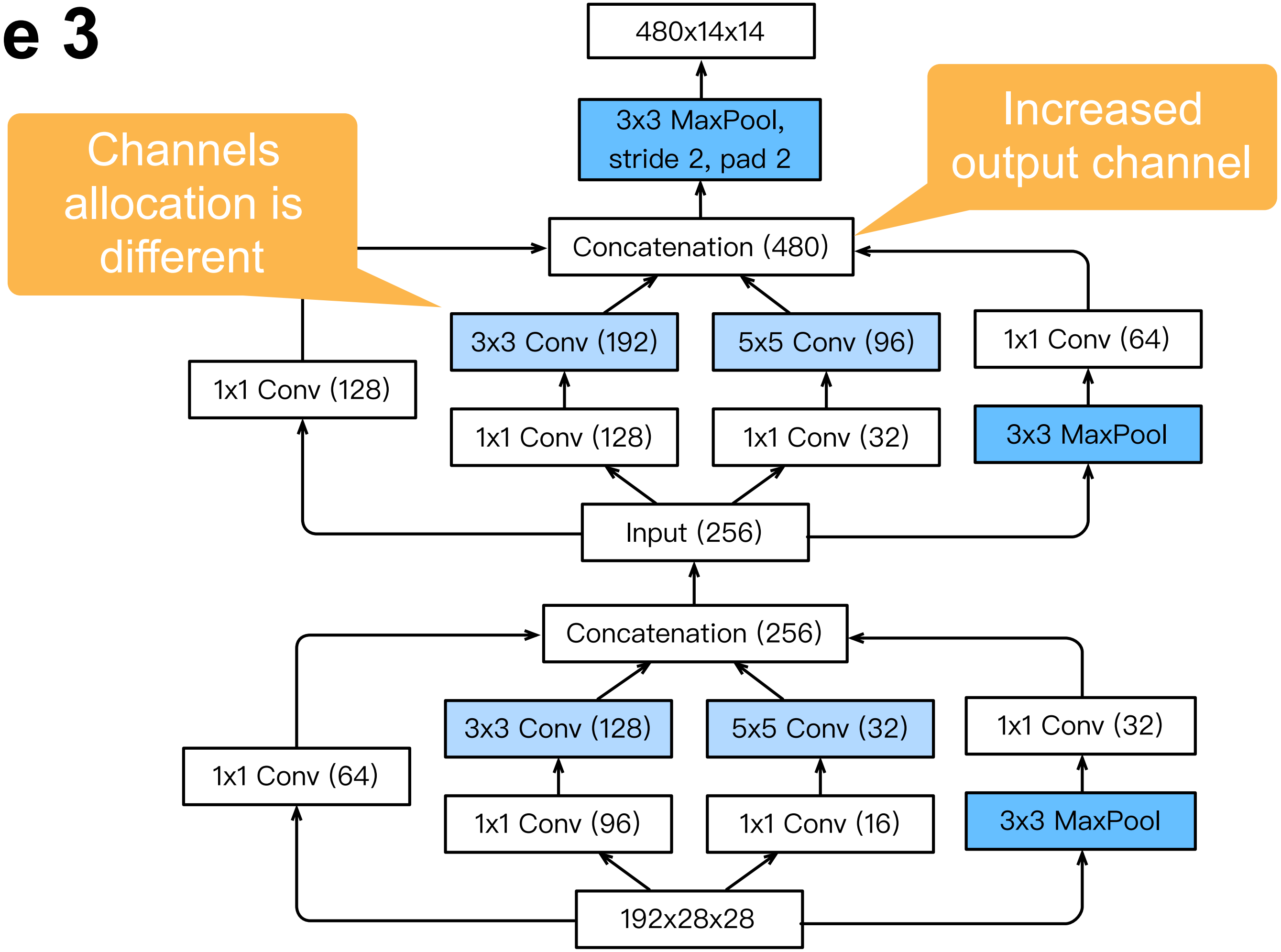


Stage 1 & 2

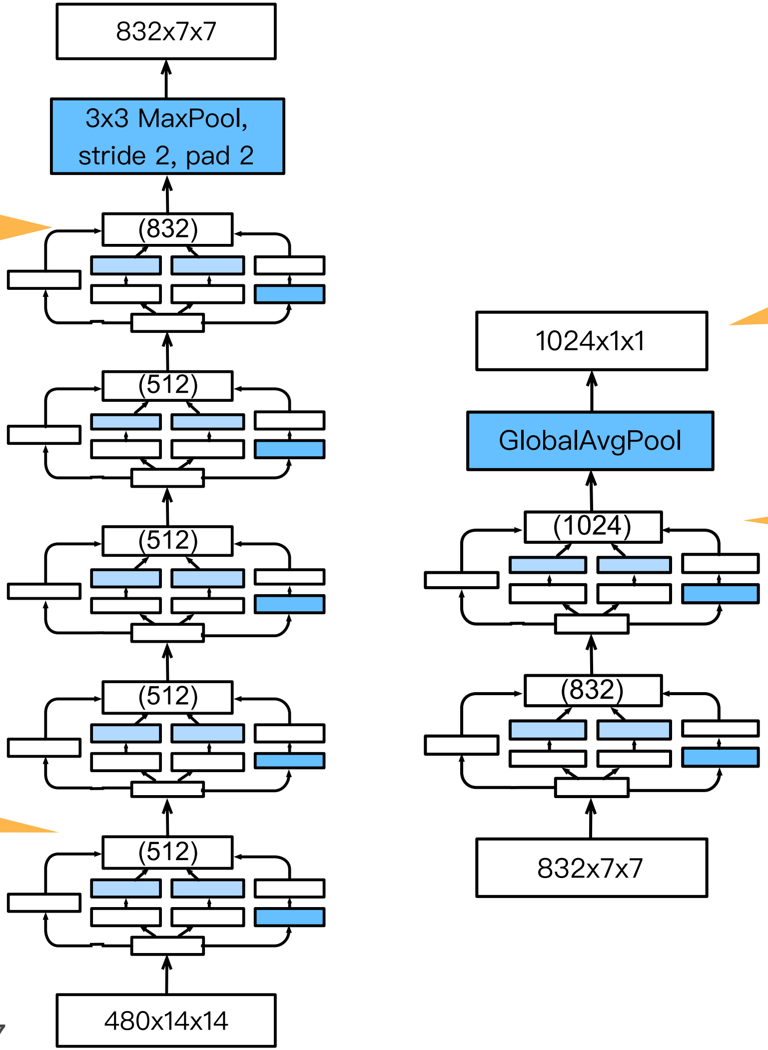
- Smaller kernel size and output channels due to more layers



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

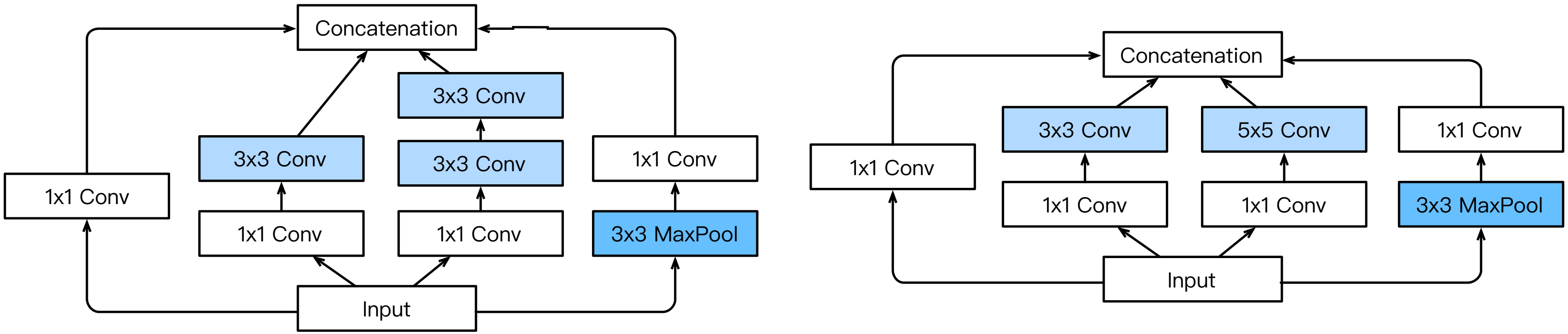
1024-dim feature to output layer

Increased output channel

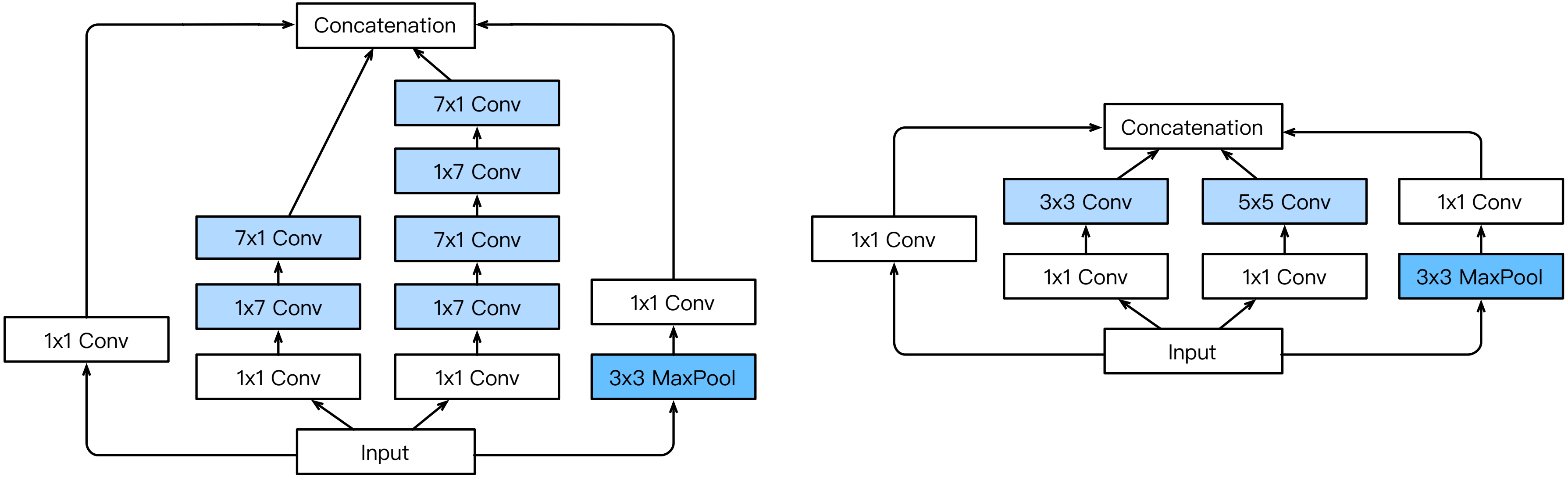
The many flavors of Inception Networks

- 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)

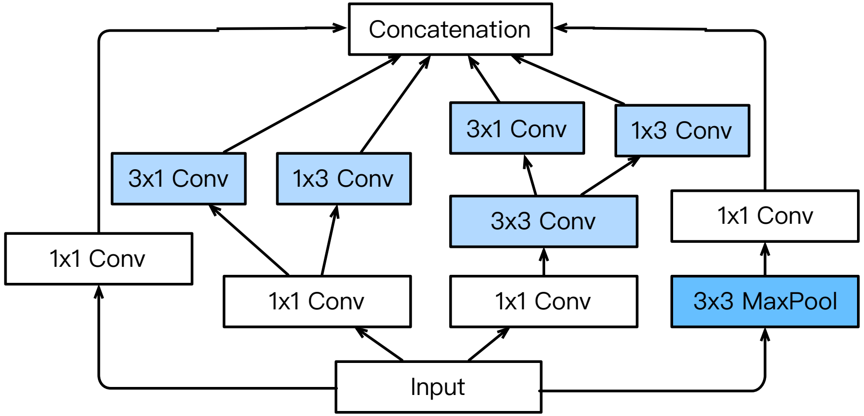
Inception V3 Block for Stage 3

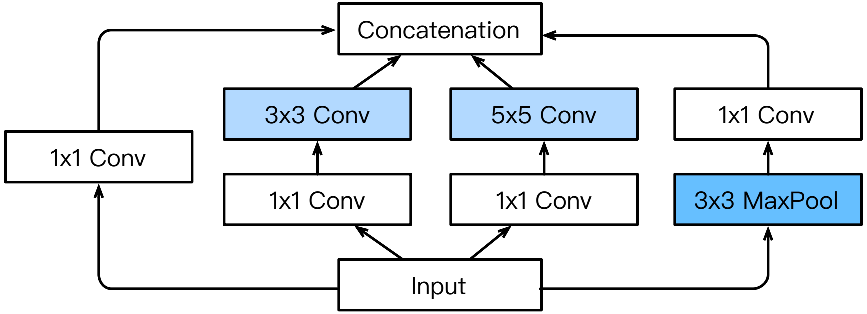


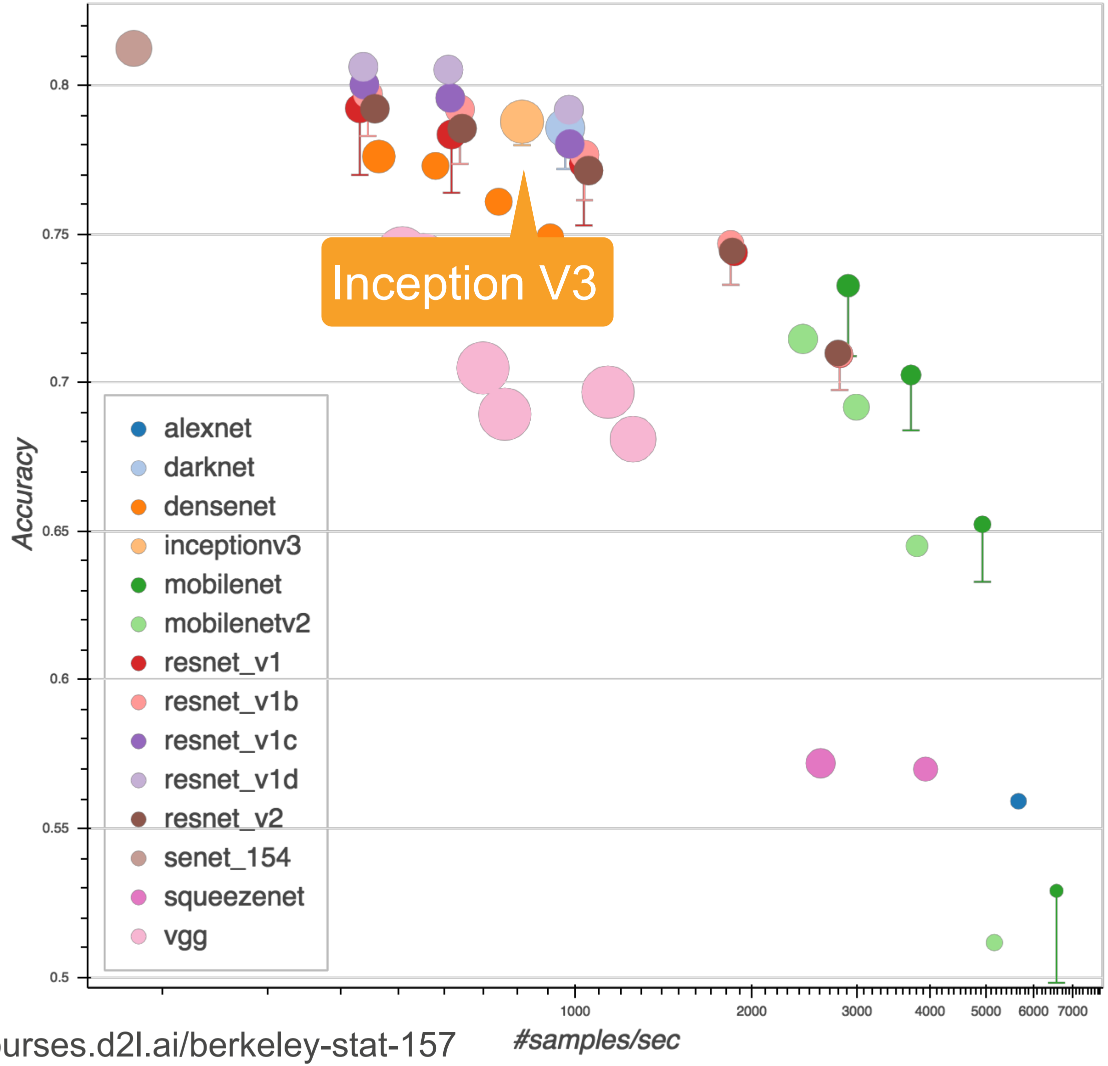
Inception V3 Block for Stage 4



Inception V3 Block for Stage 5







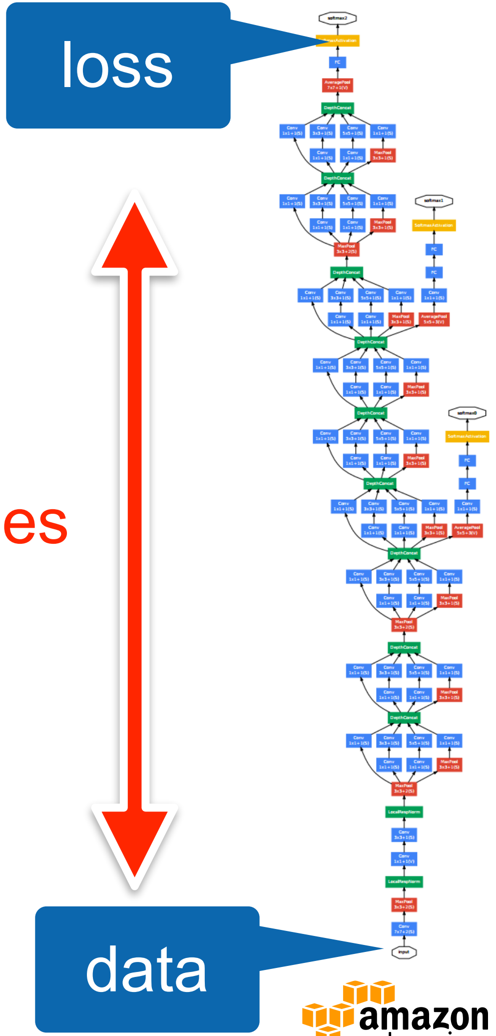
GluonCV Model Zoo

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

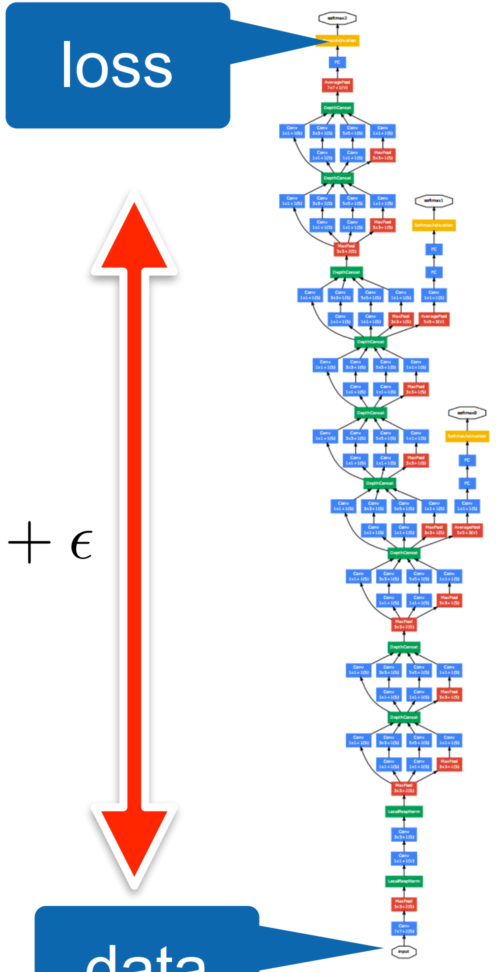

SMALL BATCH NORM
• PACIFICA, CALIFORNIA •
Batch # 22716-06
Hand Made All Natural
BLACKBERRY JAM
NET WT. 12 OZ (340 G)

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

mean

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

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

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

Random offset

Random scale

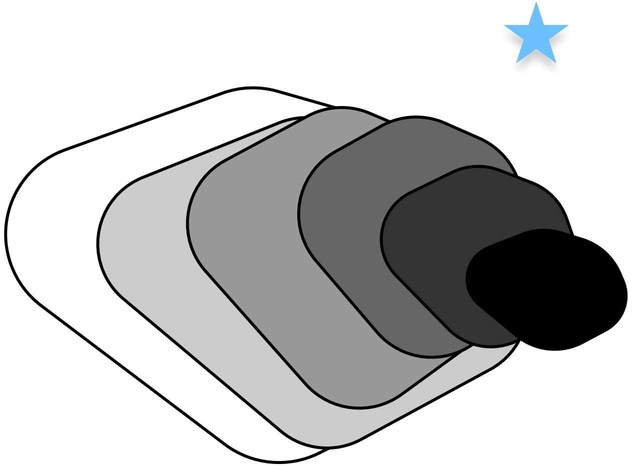
  - Random shift per minibatch
  - Random scale per minibatch
- No need to mix with dropout (both are capacity control)
- Ideal minibatch size of 64 to 256

Details

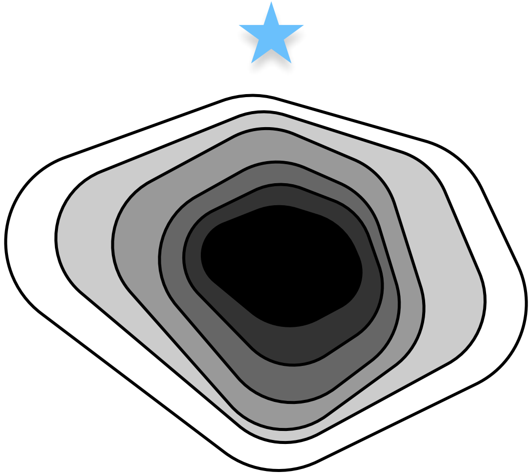
`gluon.nn.BatchNorm(…)`

- **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

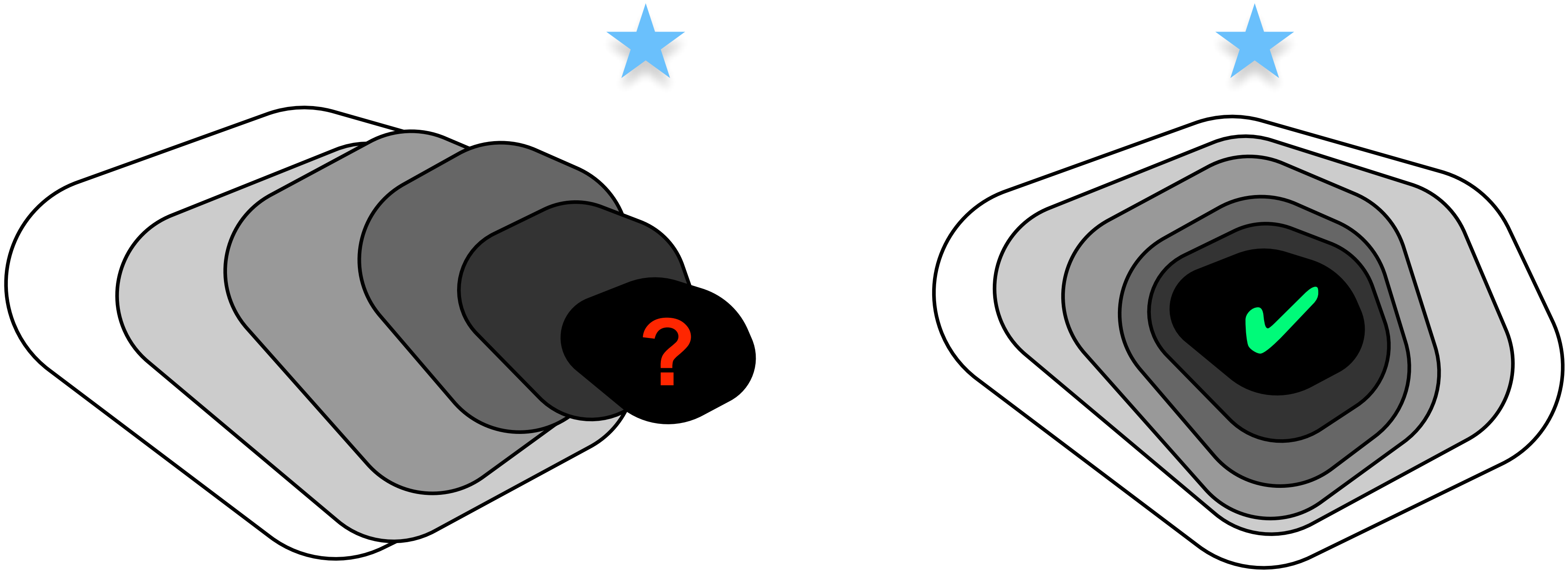


generic function classes



nested function classes

Does adding layers improve accuracy?



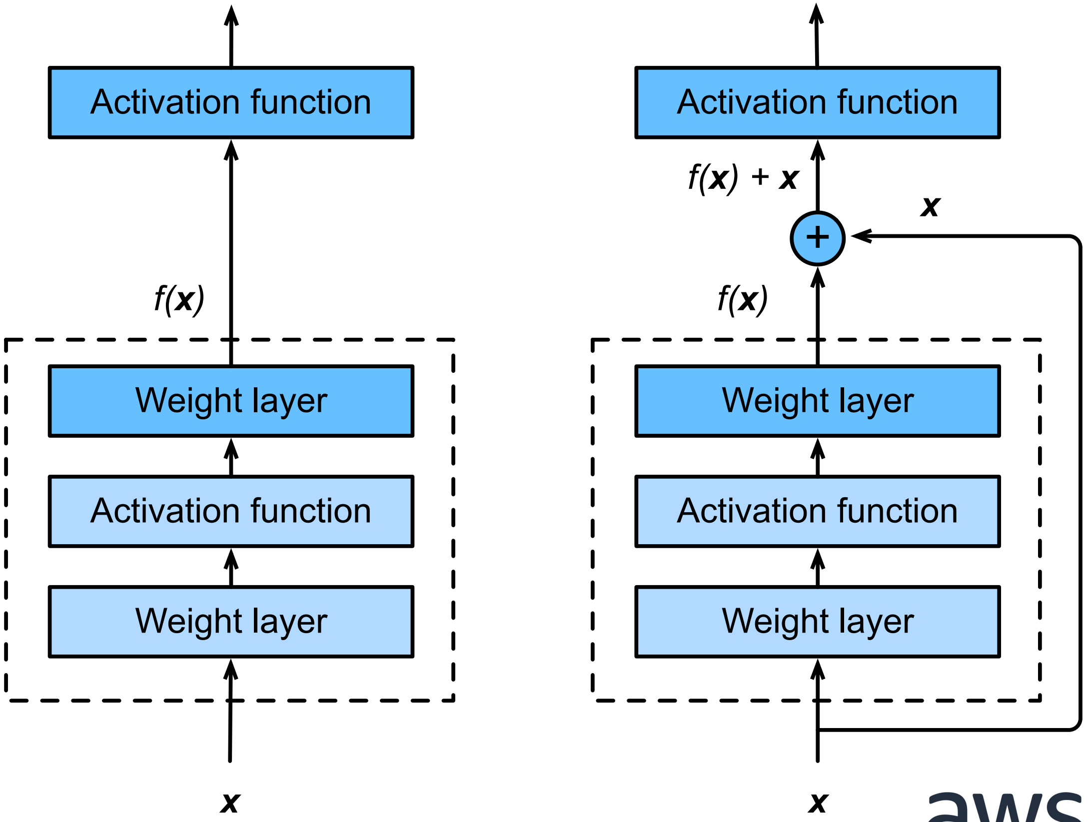
generic function classes

nested function classes

Residual Networks

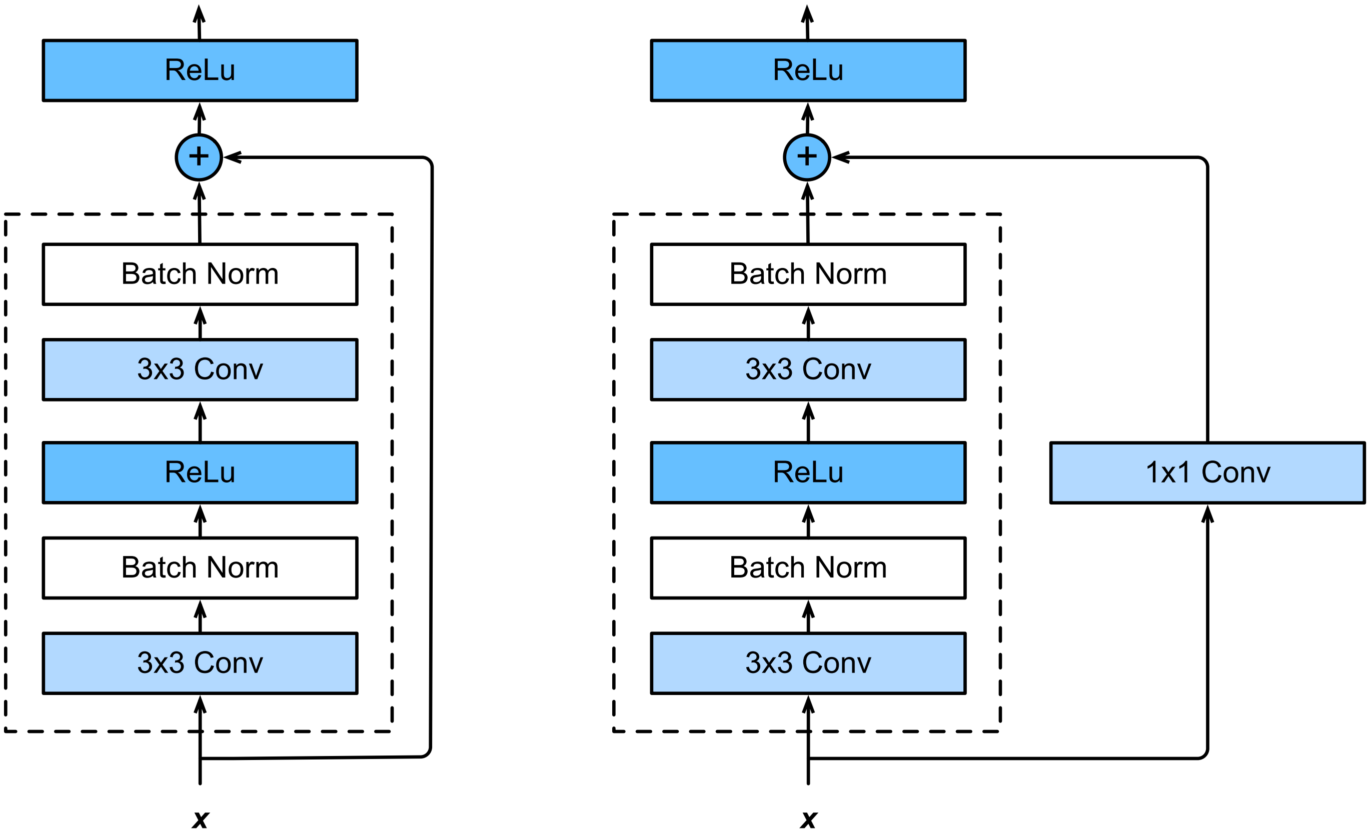
- Adding a layer **changes** function class
- We want to **add to** the function class
- 'Taylor expansion' style parametrization

$$f(x) = x + g(x)$$

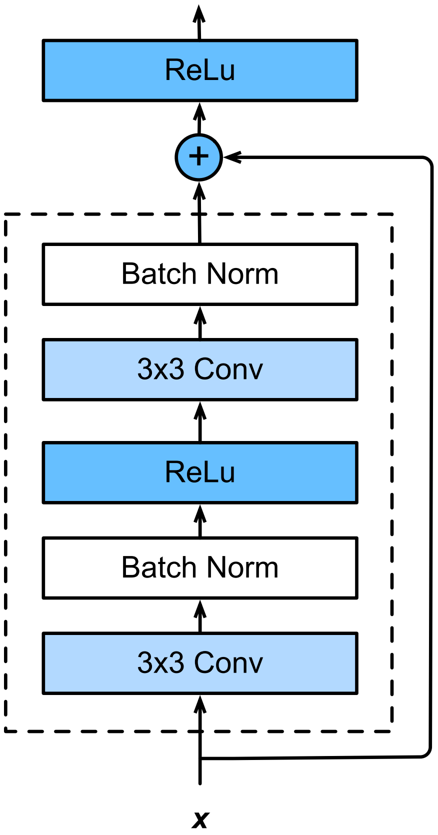


He et al., 2015

ResNet Block in detail

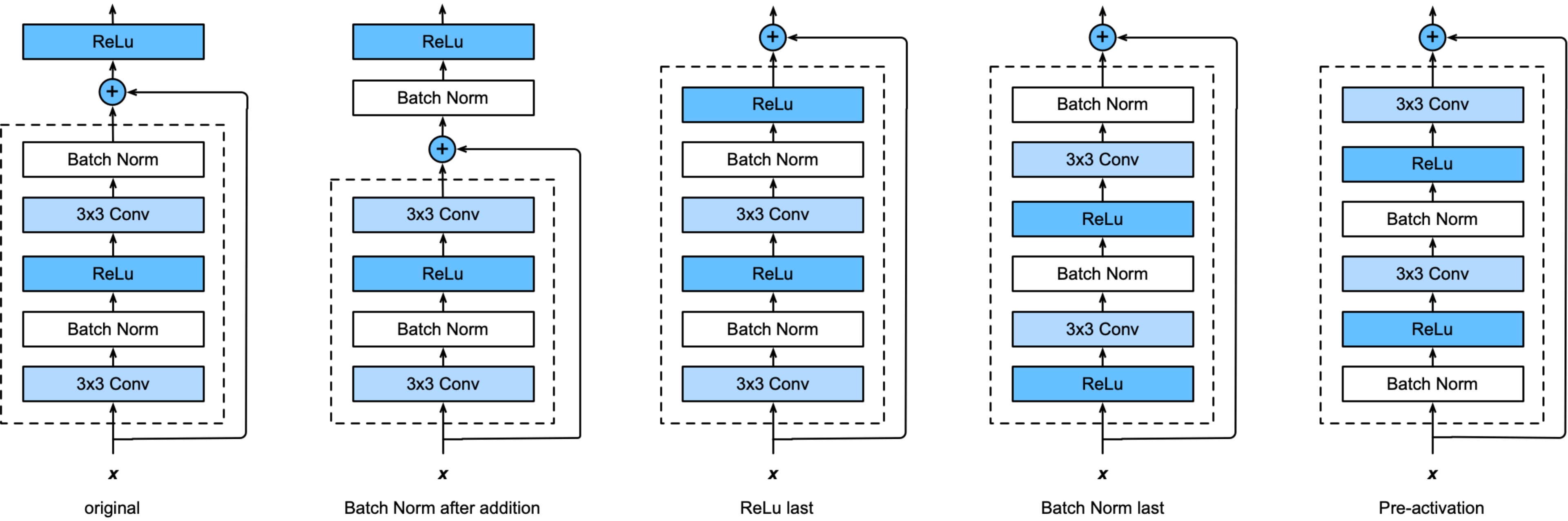


In code



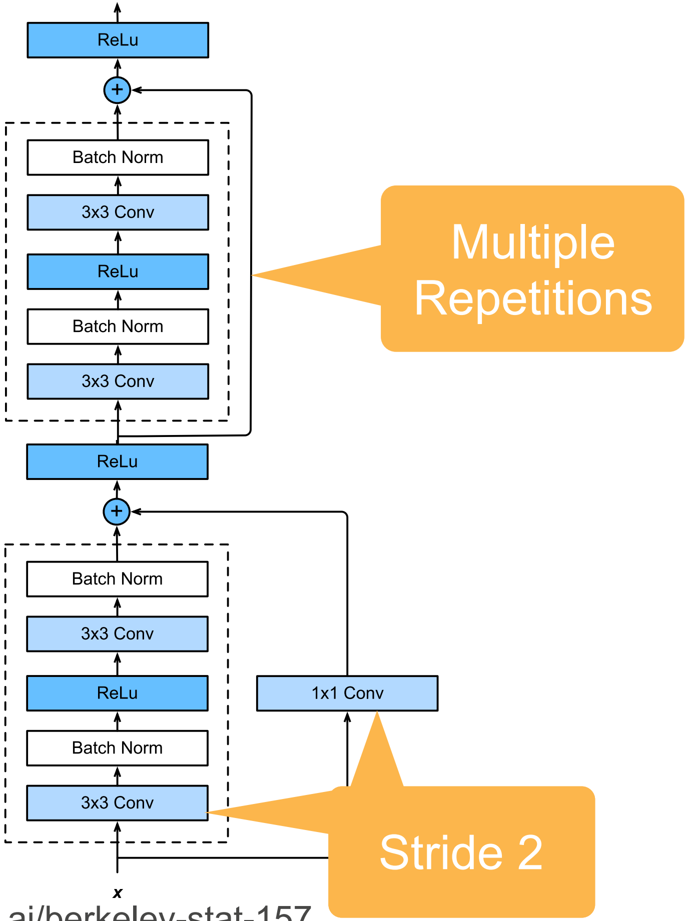

```
def forward(self, X):
    Y = self.bn1(self.conv1(X))
    Y = nd.relu(Y)
    Y = self.bn2(self.conv2(Y))
    if self.conv3:
        X = self.conv3(X)
    return nd.relu(Y + X)
```

The many flavors of ResNet blocks



Try every permutation

ResNet Module



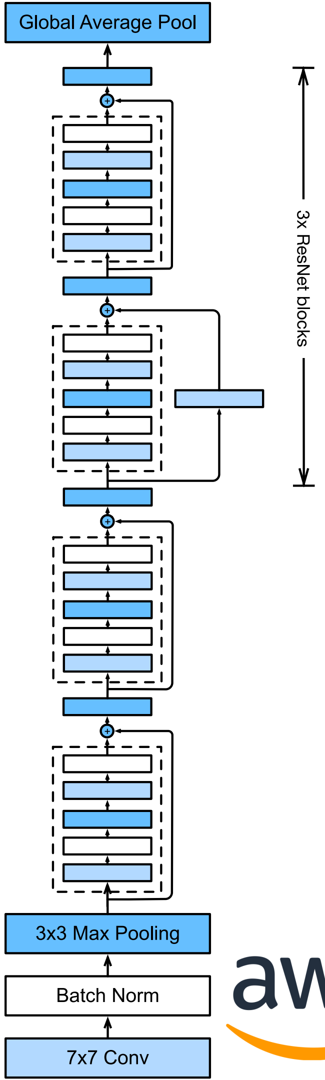
- Downsample per module (stride=2)
- Enforce some nontrivial nonlinearity per module (via 1x1 convolution)
- Stack up in blocks

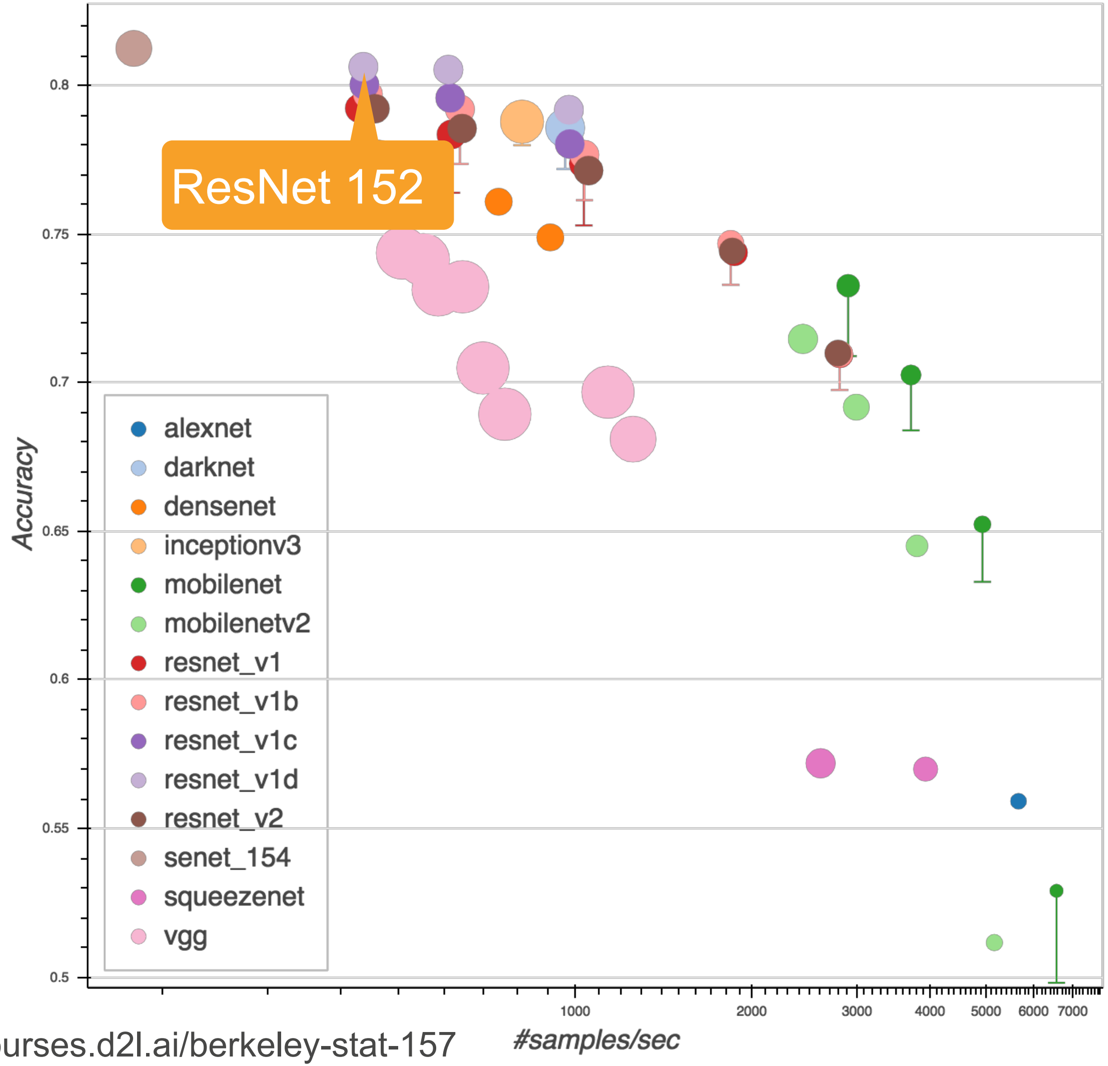
```
blk = nn.Sequential()
for i in range(num_residuals):
    if i == 0 and not first_block:
        blk.add(Residual(num_channels,
          use_1x1conv=True, strides=2))
    else:
        blk.add(Residual(num_channels))
```

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 …

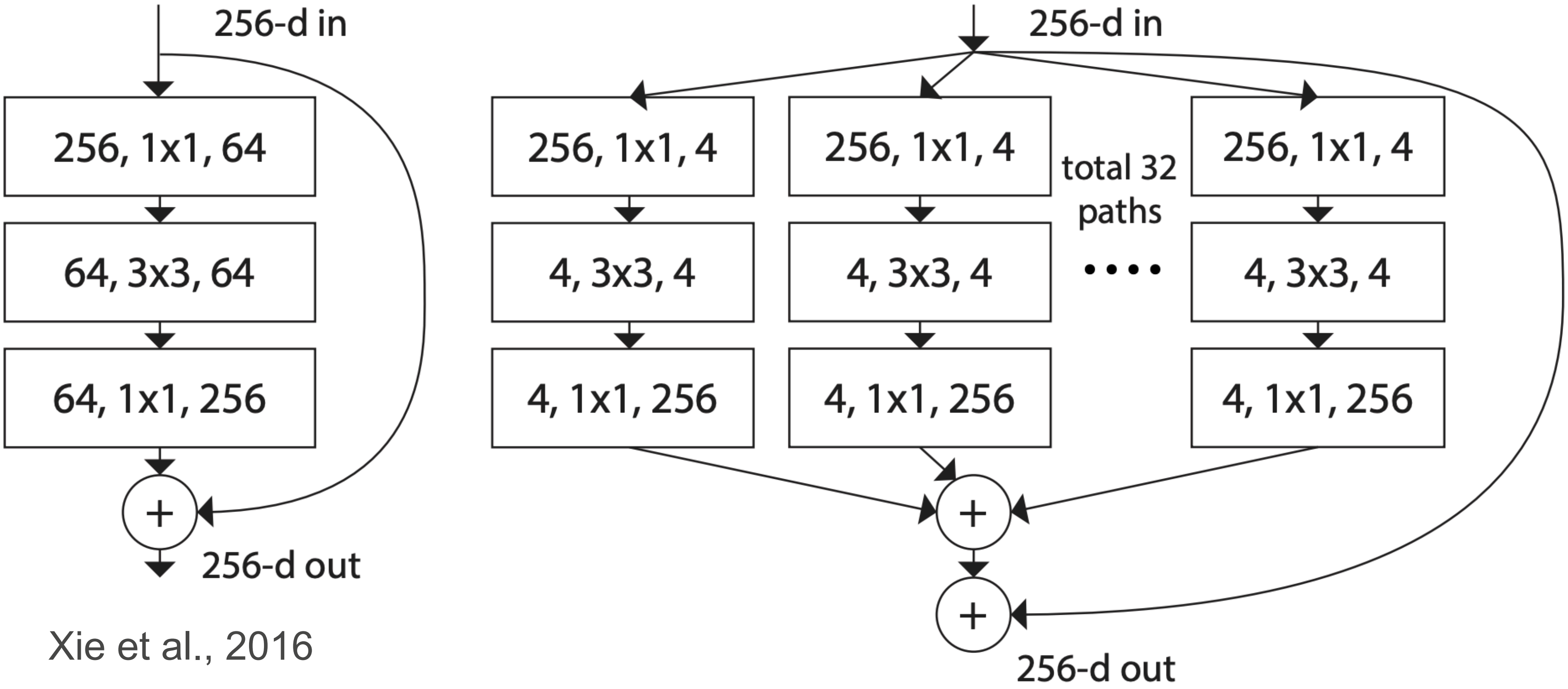




GluonCV Model Zoo

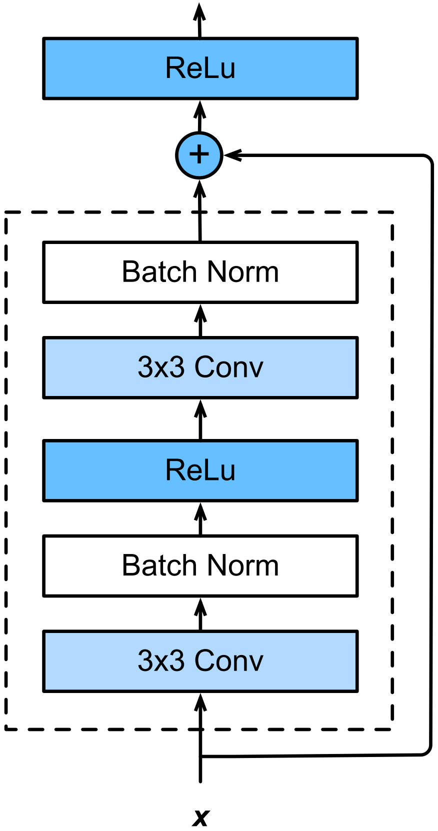
https://gluon-cv.mxnet.io/model_zoo/classification.html

ResNext



Xie et al., 2016

Reducing the cost of Convolutions



- **Parameters**

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

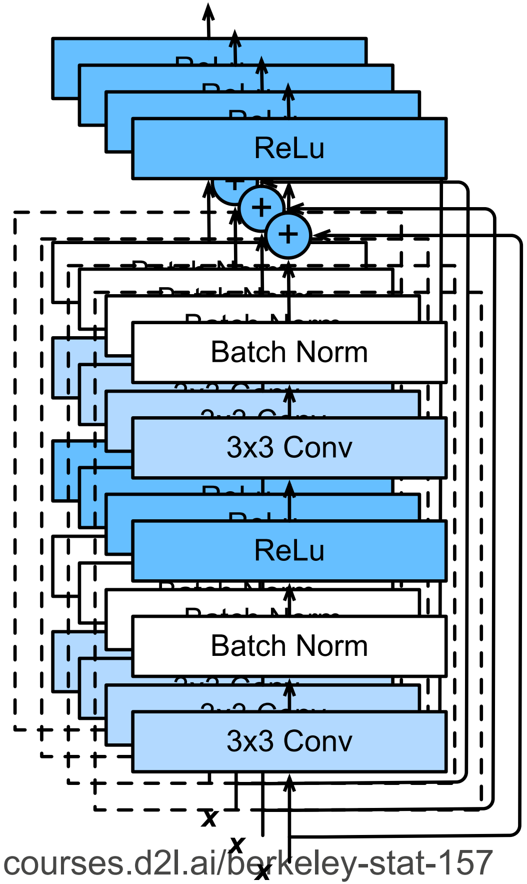
- **Computation**

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

- **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
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C{=}32 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C{=}32 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C{=}32 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C{=}32 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**$\times10^6$	**25.0**$\times10^6$
FLOPs		**4.1**$\times10^9$	**4.2**$\times10^9$

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

DenseNet (Huang et al., 2016)

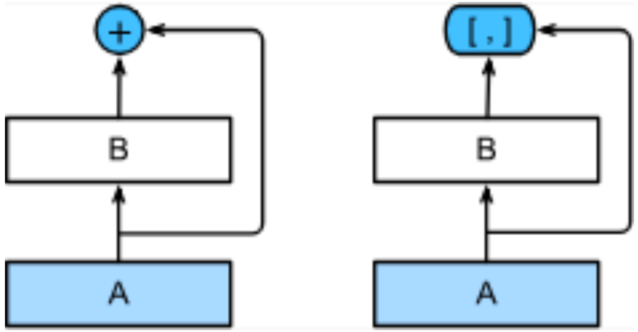
- ResNet combines x and f(x)
- DenseNet uses higher order 'Taylor series' expansion

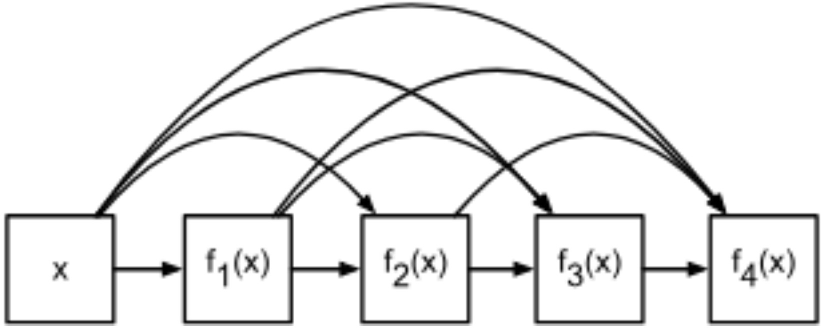
$$x_{i+1} = [x_i, f_i(x_i)]$$

$$x_1 = x$$

$$x_2 = [x, f_1(x)]$$

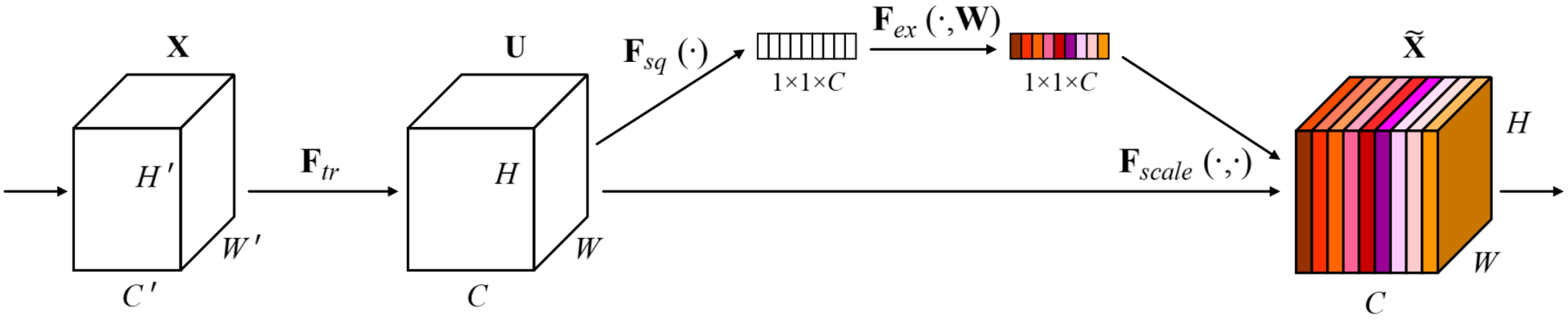
$$x_2 = [x, f_1(x), f_2([x, f_1(x)])]$$





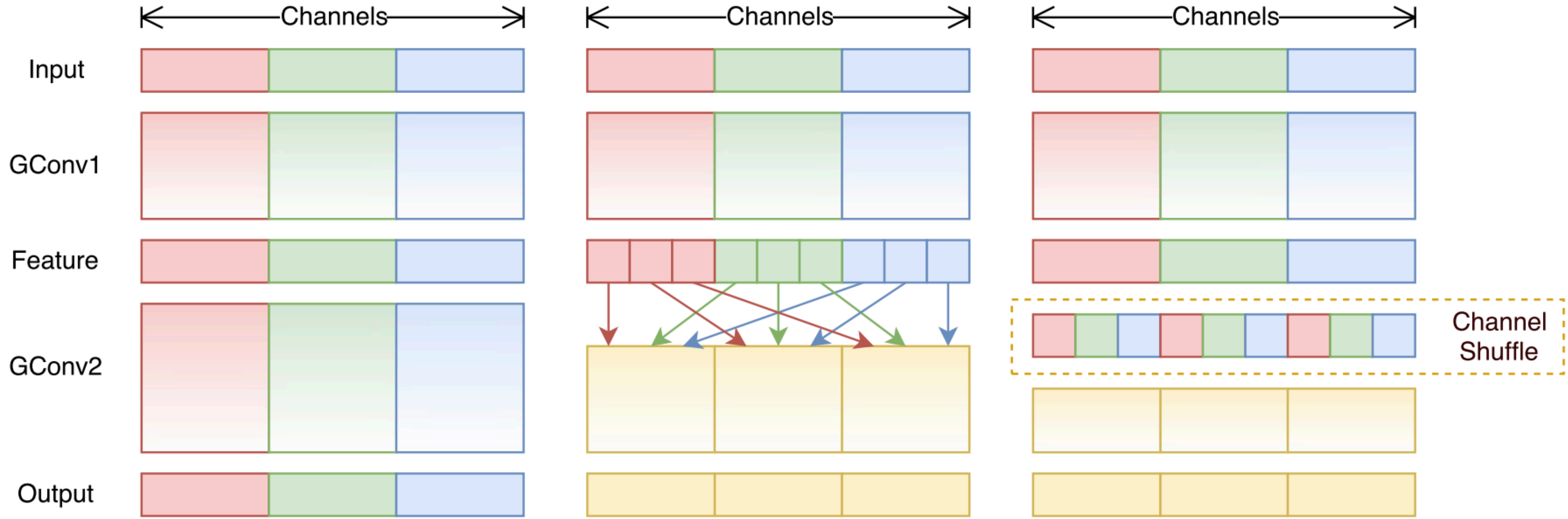
- Occasionally need to reduce resolution (transition layer)

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)

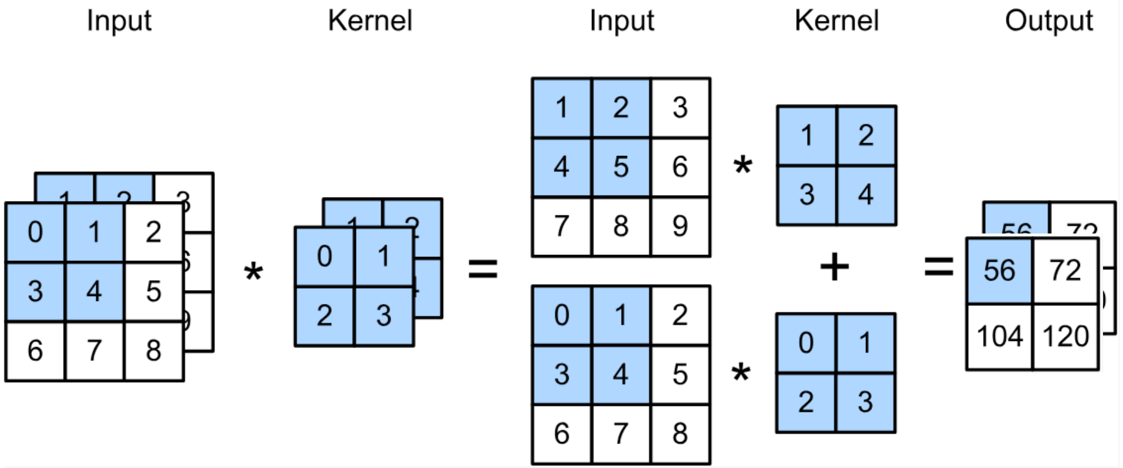


- ResNext breaks convolution into channels
- ShuffleNet mixes by grouping (very efficient for mobile)

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

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



Summary

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