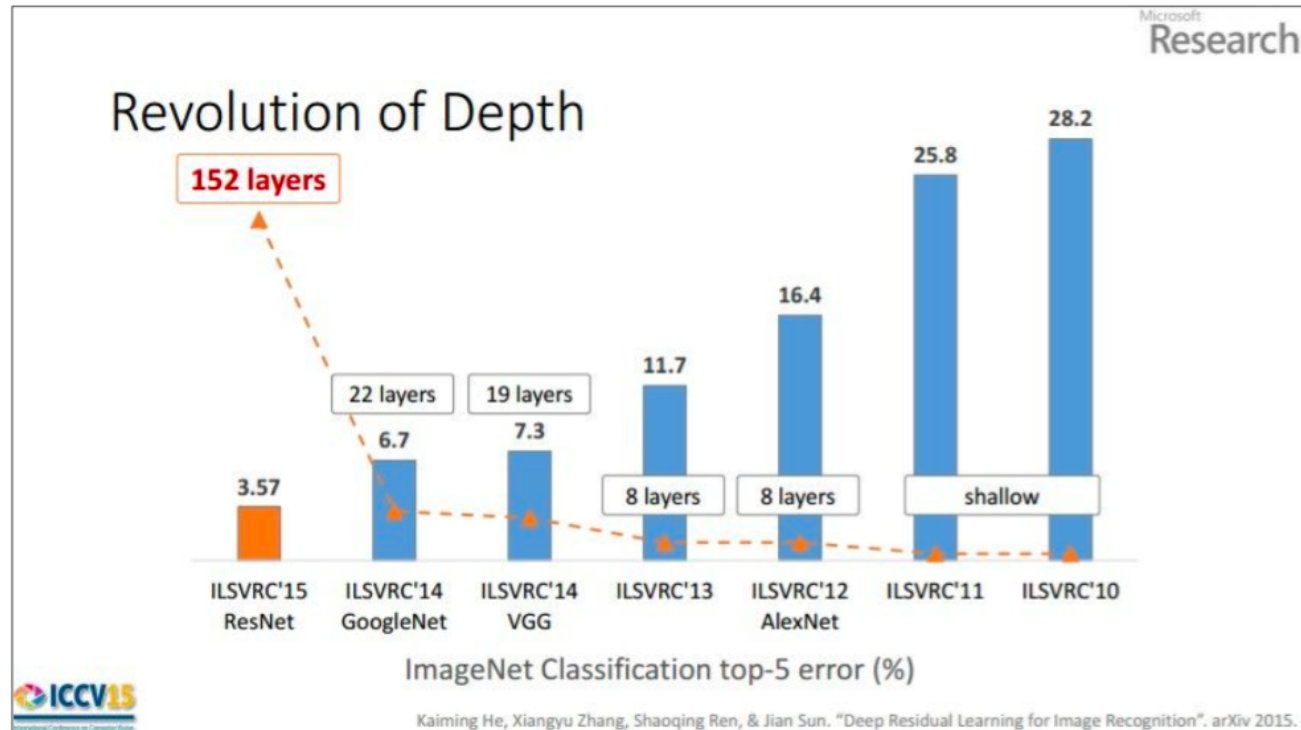# TTIC 31230, Fundamentals of Deep Learning

David McAllester, Autumn 2020

Convolutional Neural Networks (CNNs)
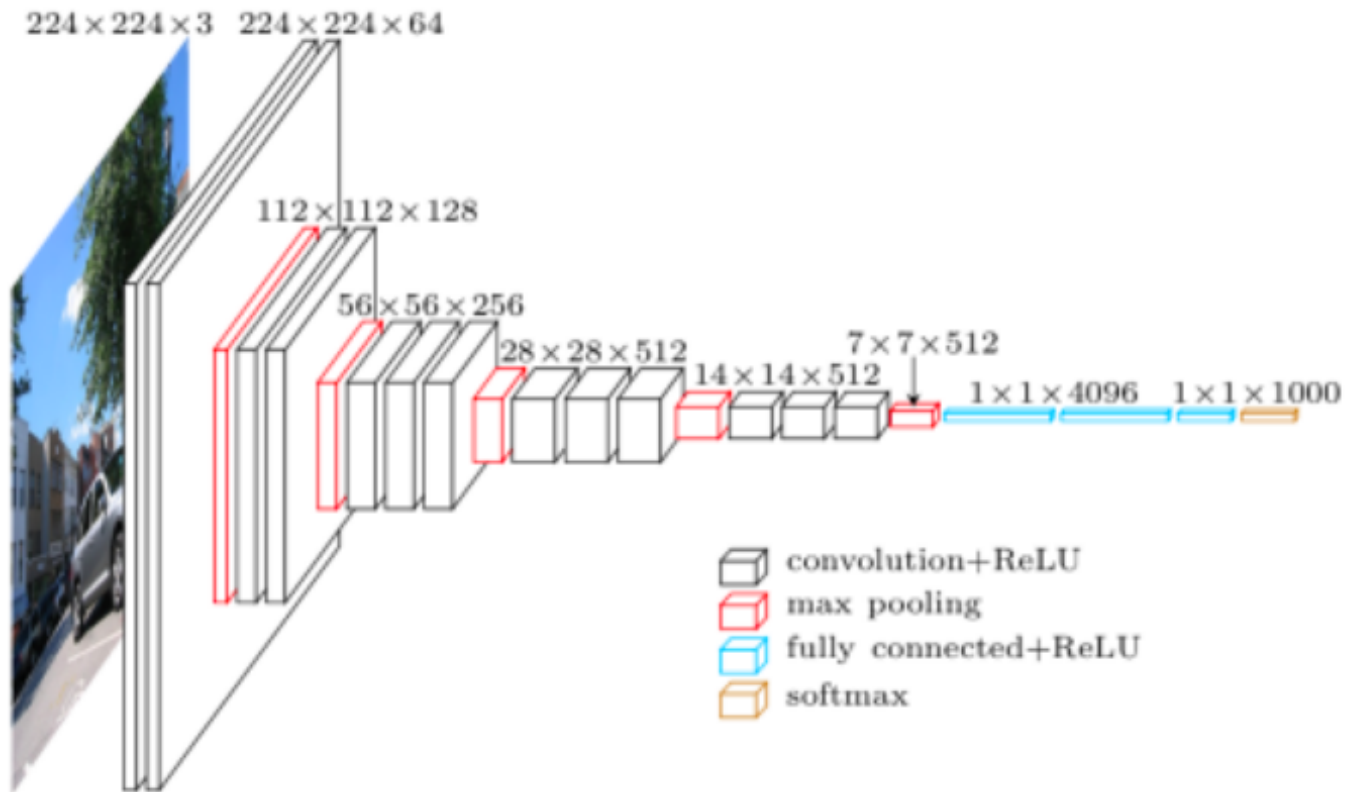
Imagenet Classification

1000 kinds of objects.

(slide from Kaiming He's recent presentation)

2016 is 3.0%, is 2017 2.25%

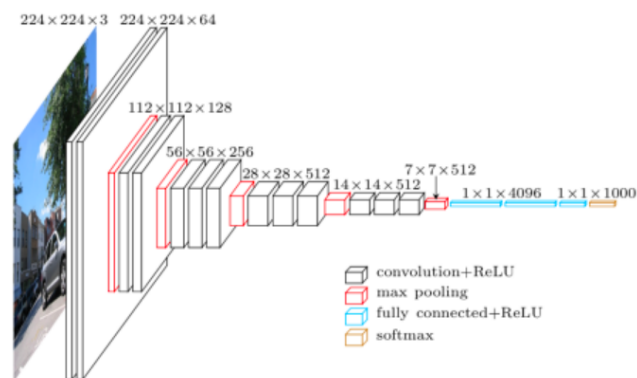SOTA as of January 2020 is 1.3%

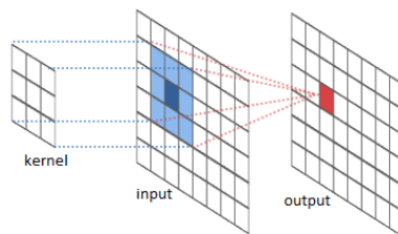# What is a CNN?
VGG, Zisserman, 2014

Davi Frossard

A Convolution Layer

Each box is a tensor $L_\ell[b, x, y, i]$

Each value $L_\ell[b, x, y, i]$ (for $\ell > 0$) is the output of a single linear threshold unit.

A Convolution Layer

$$W[\Delta x, \Delta y, i, j] \qquad L_\ell[b, x, y, i] \qquad L_{\ell+1}[b, x, y, j]$$

River Trail Documentation

$$L_{\ell+1}[b, x, y, j]$$

$$= \sigma\left(W[\Delta X, \Delta Y, I, j]\, L_\ell[b, x + \Delta X, y + \Delta Y, I] - B[j]\right)$$

5

2D CNN in PyTorch

conv2d(**input, weight, bias, stride, padding, dilation, groups**)

input – tensor (minibatch,in-channels,iH,iW)

weight – filters (out-channels, in-channels/groups,kH,kW)

bias – tensor (out-channels) . Default: None

stride – Single number or (sH, sW). Default: 1

padding – Single number or (padH, padW). Default: 0

dilation – Single number or (dH, dW). Default: 1

groups – split input into groups. Default: 1

Padding
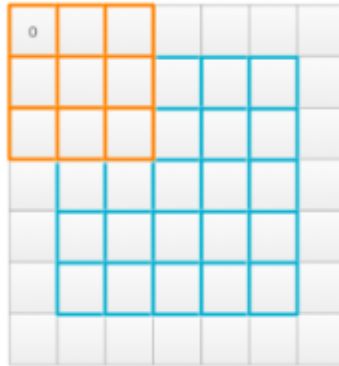
If we pad the input with zeros then the input and output can
have the same spatial dimensions.

Zero Padding in NumPy

In NumPy we can add a zero padding of width p to an image as follows:

```
padded = np.zeros(W + 2*p,  H + 2*p)

padded[p:W+p, p:H+p] = x
```

Padding

$$L'_\ell = \text{Padd}(L_\ell, \ p)$$

$$L_{\ell+1}[b, x, y, j] =$$

$$\sigma \left(W[\Delta X, \Delta Y, I, j] \ L'_\ell[b, x + \Delta X, y + \Delta Y, I] - B[j] \right)$$

If the input is padded but the output is not padded then Δx and Δy are non-negative.

Reducing Spatial Dimention
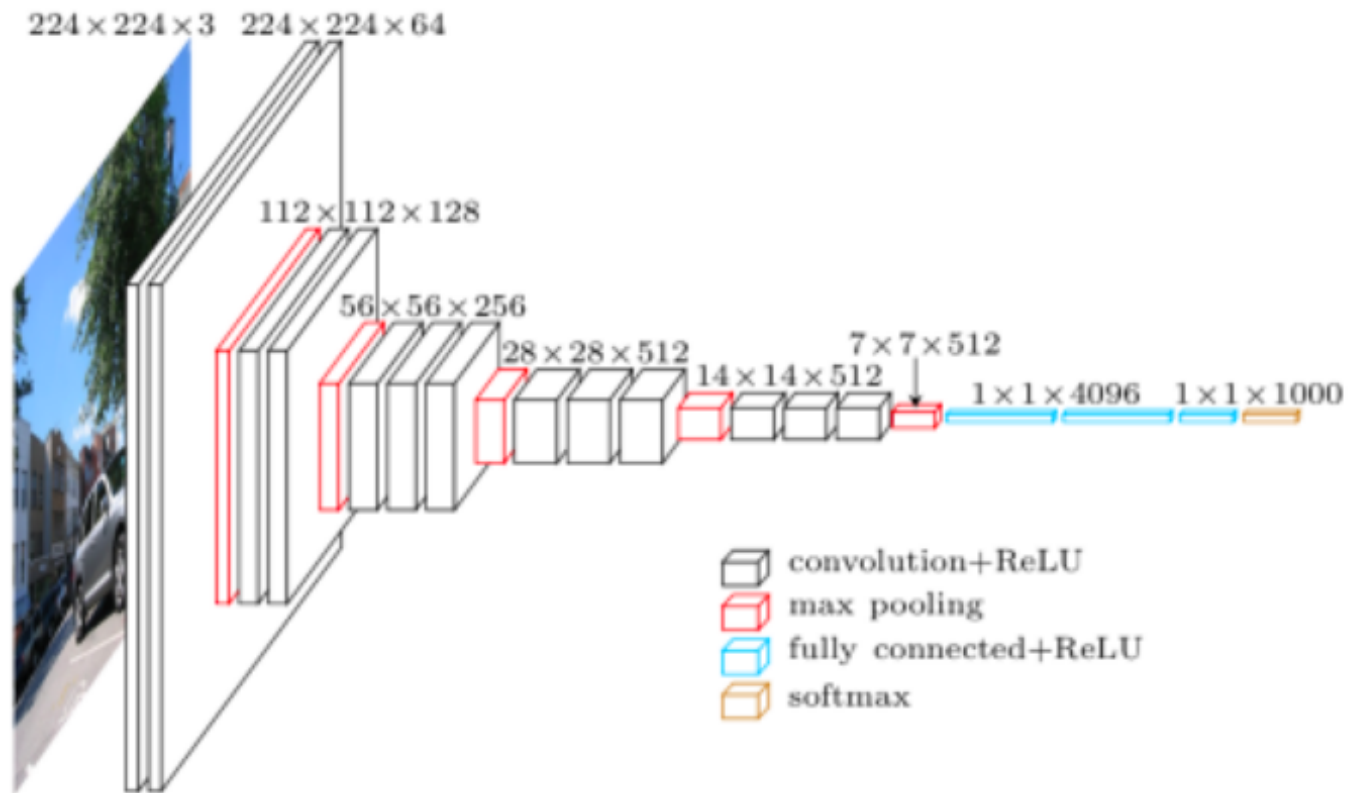
Reducing Spatial Dimensions: Max Pooling
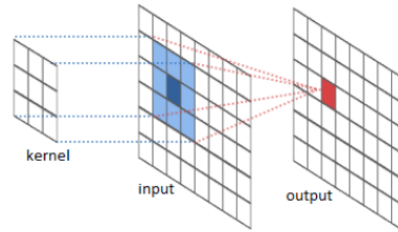
$$L_{\ell+1}[b, x, y, i] = \max_{\Delta x, \Delta y}\ L_\ell[b, s*x + \Delta x,\ s*y + \Delta y,\ i]$$

This is typically done with a stride greater than one so that the image dimension is reduced.

Reducing Spatial Dimensions: Strided Convolution

We can move the filter by a "stride" s for each spatial step.

$$L_{\ell+1}[b, x, y, j] =$$

$$\sigma\left(W[\Delta X, \Delta Y, I, j]L_\ell[b, s*x + \Delta X, s*y + \Delta Y, I] - B[j]\right)$$

For strides greater than 1 the spatial dimention is reduced.

Fully Connected (FC) Layers

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
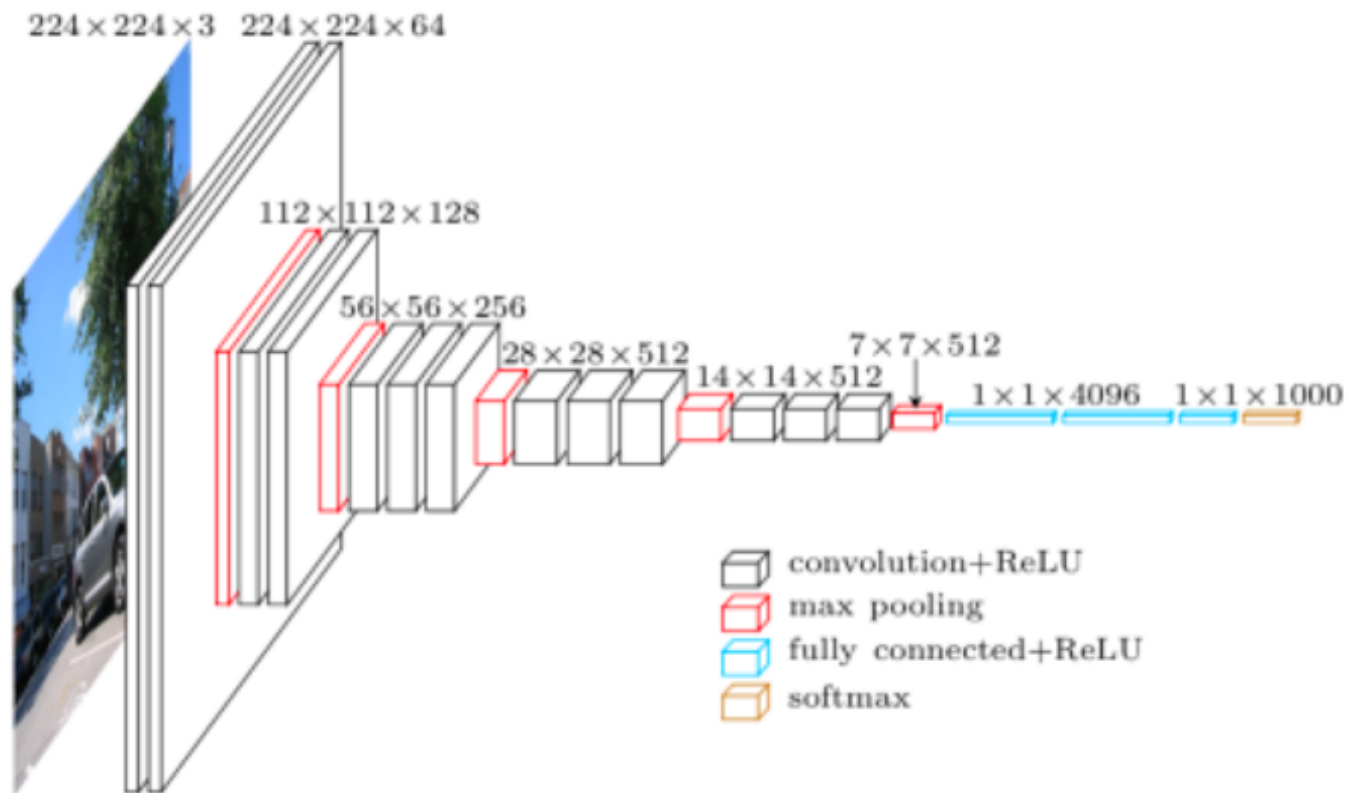
Fully Connected (FC) Layers

We reshape $L_\ell[b, x, y, i]$ to $L_\ell[b, i']$ and then

$$L_{\ell+1}[b, j] = \sigma\left(W[j, I]\, L_\ell[b, I] - B[j]\right)$$

2D CNN in PyTorch

conv2d(**input, weight, bias, stride, padding, dilation, groups**)

input – tensor (minibatch,in-channels,iH,iW)

weight – filters (out-channels, in-channels/groups,kH,kW)

bias – tensor (out-channels) . Default: None

stride – Single number or (sH, sW). Default: 1

padding – Single number or (padH, padW). Default: 0

dilation – Single number or (dH, dW). Default: 1

groups – split input into groups. Default: 1

Dilation and grouping are discussed in a separate unit.

Modern Trends

Modern Convolutions use 3X3 filters. This is faster and has fewer parameters. Expressive power is preserved by increasing depth with many stride 1 layers.

Max pooling and dilation seem to have disappeared.

ResNet and resnet-like architectures are now dominant.

Alexnet, 2012

Given Input$[227, 227, 3]$

$$L_1[55 \times 55 \times 96] = \text{ReLU}(\text{CONV}(\text{Input}, \Phi_1, \text{width } 11, \text{pad } 0, \text{stride } 4))$$
$$L_2[27 \times 27 \times 96] = \text{MaxPool}(L_1, \text{width } 3, \text{stride } 2))$$
$$L_3[27 \times 27 \times 256] = \text{ReLU}(\text{CONV}(L_2, \Phi_3, \text{width } 5, \text{pad } 2, \text{stride } 1))$$
$$L_4[13 \times 13 \times 256] = \text{MaxPool}(L_3, \text{width } 3, \text{stride } 2))$$
$$L_5[13 \times 13 \times 384] = \text{ReLU}(\text{CONV}(L_4, \Phi_5, \text{width } 3, \text{pad } 1, \text{stride } 1))$$
$$L_6[13 \times 13 \times 384] = \text{ReLU}(\text{CONV}(L_5, \Phi_6, \text{width } 3, \text{pad } 1, \text{stride } 1))$$
$$L_7[13 \times 13 \times 256] = \text{ReLU}(\text{CONV}(L_6, \Phi_7, \text{width } 3, \text{pad } 1, \text{stride } 1))$$
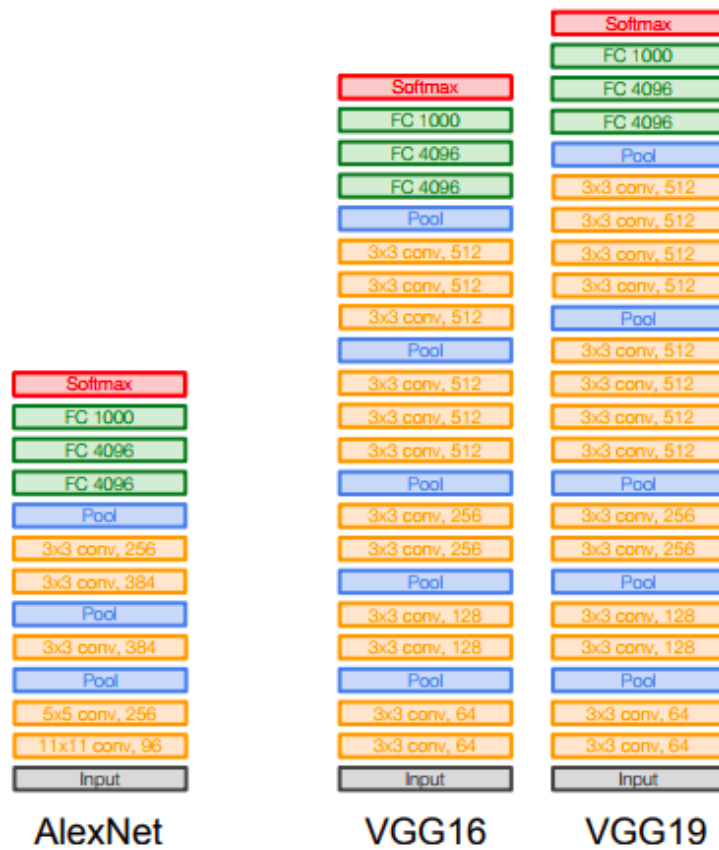$$L_8[6 \times 6 \times 256] = \text{MaxPool}(L_7, \text{width } 3, \text{stride } 2))$$
$$L_9[4096] = \text{ReLU}(\text{FC}(L_8, \Phi_9))$$
$$L_{10}[4096] = \text{ReLU}(\text{FC}(L_9, \Phi_{10}))$$
$$s[1000] = \text{ReLU}(\text{FC}(L_{10}, \Phi_s)) \quad \text{class scores}$$

VGG, 2014

AlexNet:
- Softmax
- FC 1000
- FC 4096
- FC 4096
- Pool
- 3x3 conv, 256
- 3x3 conv, 384
- Pool
- 3x3 conv, 384
- Pool
- 5x5 conv, 256
- 11x11 conv, 96
- Input

VGG16:
- Softmax
- FC 1000
- FC 4096
- FC 4096
- Pool
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- Pool
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- Pool
- 3x3 conv, 256
- 3x3 conv, 256
- Pool
- 3x3 conv, 128
- 3x3 conv, 128
- Pool
- 3x3 conv, 64
- 3x3 conv, 64
- Input

VGG19:
- Softmax
- FC 1000
- FC 4096
- FC 4096
- Pool
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- Pool
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- 3x3 conv, 512
- Pool
- 3x3 conv, 256
- 3x3 conv, 256
- Pool
- 3x3 conv, 128
- 3x3 conv, 128
- Pool
- 3x3 conv, 64
- 3x3 conv, 64
- Input

AlexNet   VGG16   VGG19

Stanford CS231

18

Inception, Google, 2014

ResNet, 2015

plain net

ResNet

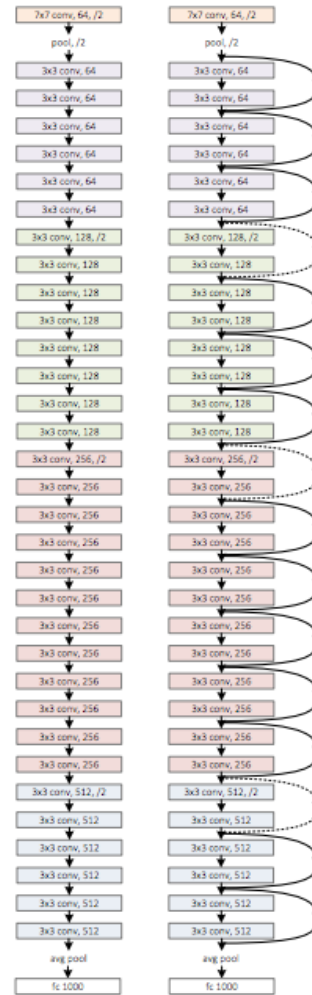| 7x7 conv, 64, /2 |
| pool, /2 |
| 3x3 conv, 64 |
| 3x3 conv, 64 |
| 3x3 conv, 64 |
| 3x3 conv, 64 |
| 3x3 conv, 64 |
| 3x3 conv, 64 |
| 3x3 conv, 128, /2 |
| 3x3 conv, 128 |
| 3x3 conv, 128 |
| 3x3 conv, 128 |
| 3x3 conv, 128 |
| 3x3 conv, 128 |
| 3x3 conv, 128 |
| 3x3 conv, 128 |
| 3x3 conv, 256, /2 |
| 3x3 conv, 256 |
| 3x3 conv, 256 |
| 3x3 conv, 256 |
| 3x3 conv, 256 |
| 3x3 conv, 256 |
| 3x3 conv, 256 |
| 3x3 conv, 256 |
| 3x3 conv, 256 |
| 3x3 conv, 256 |
| 3x3 conv, 256 |
| 3x3 conv, 256 |
| 3x3 conv, 512, /2 |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| 3x3 conv, 512 |
| avg pool |
| fc 1000 |

er)

[Kaiming He]

END

Image to Column (Im2C)

Reduce convolution to matrix multiplication

more space but faster.

$$\tilde{L}_{\ell+1}[b, x, y, j]$$

$$= \left(\sum_{\Delta x, \Delta y, i} W[\Delta x, \Delta y, i, j] * L_\ell[b, x + \Delta x, \ y + \Delta y, \ i] \right) + B[j]$$

We make a bigger tensor $\tilde{L}$ with two additional indeces.

$$\tilde{L}_\ell[b, x, y, \Delta x, \Delta y, i] = L_\ell[b, x + \Delta x, y + \Delta y, i]$$

Image to Column (Im2C)

$$\tilde{L}_{\ell+1}[b, x, y, j]$$

$$= \left(\sum_{\Delta x, \Delta y, i} W[\Delta x, \Delta y, i, j] * L_\ell[b, x + \Delta x, \ y + \Delta y, \ i] \right) + B[j]$$

$$= \left(\sum_{\Delta x, \Delta y, i} \tilde{L}_\ell[b, x, y, \Delta x, \Delta y, i] * W[\Delta x, \Delta y, i, j] \right) + B[j]$$

$$= \left(\sum_{(\Delta x, \Delta y, i)} \tilde{L}_\ell[(b, x, y), (\Delta x, \Delta y, i)] * W[(\Delta x, \Delta y, i), j] \right) + B[j]$$