

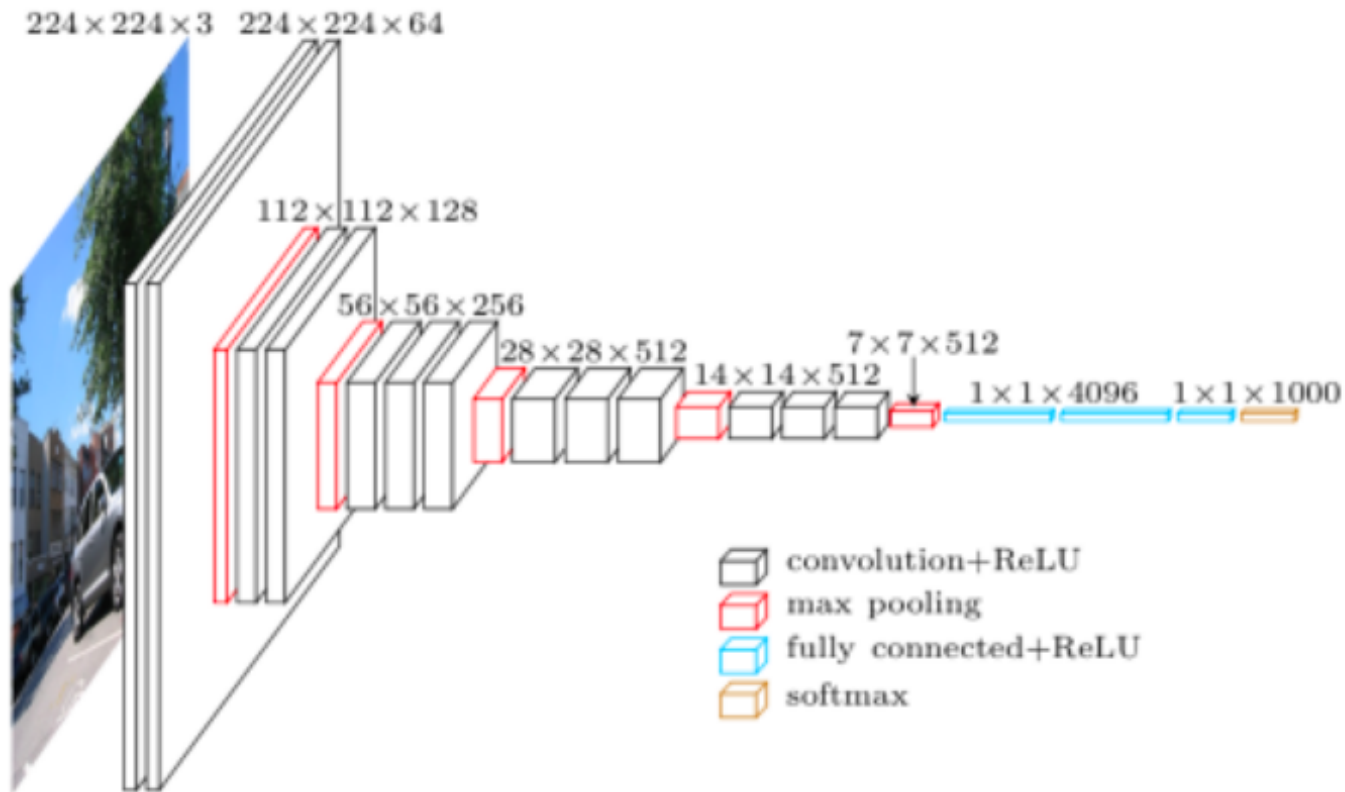
TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2019

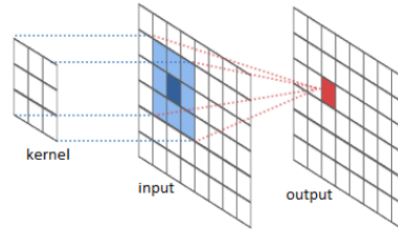
Convolutional Neural Networks (CNNs)

What is a CNN?

VGG, Zisserman, 2014



A Convolution Layer



$$W[\Delta x, \Delta y, i, j] \quad L_{\text{IN}}[b, x, y, i] \quad L_{\text{out}}[b, x, y, j]$$

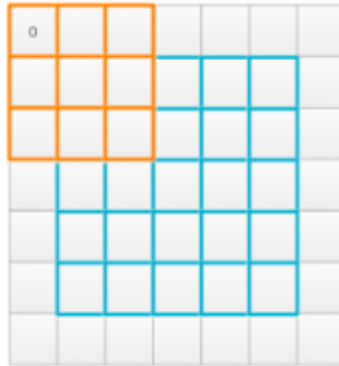
River Trail Documentation

Procedure CONV($W[\Delta x, \Delta y, i, y,]$, $B[j]$, $L_{\text{in}}[b, x, , y, i]$)

Return $L_{\text{out}}[b, x, y, j]$

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

Padding



Jonathan Hui

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

Procedure CONV(Φ , $L_{\text{in}}[b, x, , y, i]$, padding p)

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

$$L_{\text{out}}[b, x, y, j] = \left(\sum_{\substack{\Delta x, \Delta y, i \\ +B[j]}} W[\Delta x, \Delta y, i, j] L'_{\text{in}}[b, x + \Delta x, y + \Delta y, i] \right)$$

Return $L_{\text{out}}[b, x, y, j]$

Strides

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

$$L_{\text{out}}[b, x, y, j] = W[\Delta x, \Delta y, i, j] L_{\text{in}}[b, s * x + \Delta x, s * y + \Delta y, i] + B[j]$$

Max Pooling

$$L_{\text{out}}[b, x, y, c] = \max_{\Delta x, \Delta y} L_{\text{in}}[b, s * x + \Delta x, s * y + \Delta y, c]$$

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

Fully Connected (FC) Layers

We reshape $L_{\text{in}}[b, x, y, c]$ to $L_{\text{in}}[b, (x, y, c)]$ and then

$$L_{\text{out}}[b, j] = \left(\sum_i W[i, j] L_{\text{in}}[b, i] \right) + B[j]$$

Basics

- Convolution
- Padding
- Strides
- Max Pooling
- Fully Connected Layers

Alexnet

Given Input[227, 227, 3]

$$L_1[55 \times 55 \times 96] = \text{ReLU}(\text{CONV}(\text{Input}, \Phi_1, \text{width } 11, \text{pad } 0, \text{stride } 4))$$

$$L_2[27 \times 27 \times 96] = \text{MaxPool}(L_1, \text{width } 3, \text{stride } 2))$$

$$L_3[27 \times 27 \times 256] = \text{ReLU}(\text{CONV}(L_2, \Phi_3, \text{width } 5, \text{pad } 2, \text{stride } 1))$$

$$L_4[13 \times 13 \times 256] = \text{MaxPool}(L_3, \text{width } 3, \text{stride } 2))$$

$$L_5[13 \times 13 \times 384] = \text{ReLU}(\text{CONV}(L_4, \Phi_5, \text{width } 3, \text{pad } 1, \text{stride } 1))$$

$$L_6[13 \times 13 \times 384] = \text{ReLU}(\text{CONV}(L_5, \Phi_6, \text{width } 3, \text{pad } 1, \text{stride } 1))$$

$$L_7[13 \times 13 \times 256] = \text{ReLU}(\text{CONV}(L_6, \Phi_7, \text{width } 3, \text{pad } 1, \text{stride } 1))$$

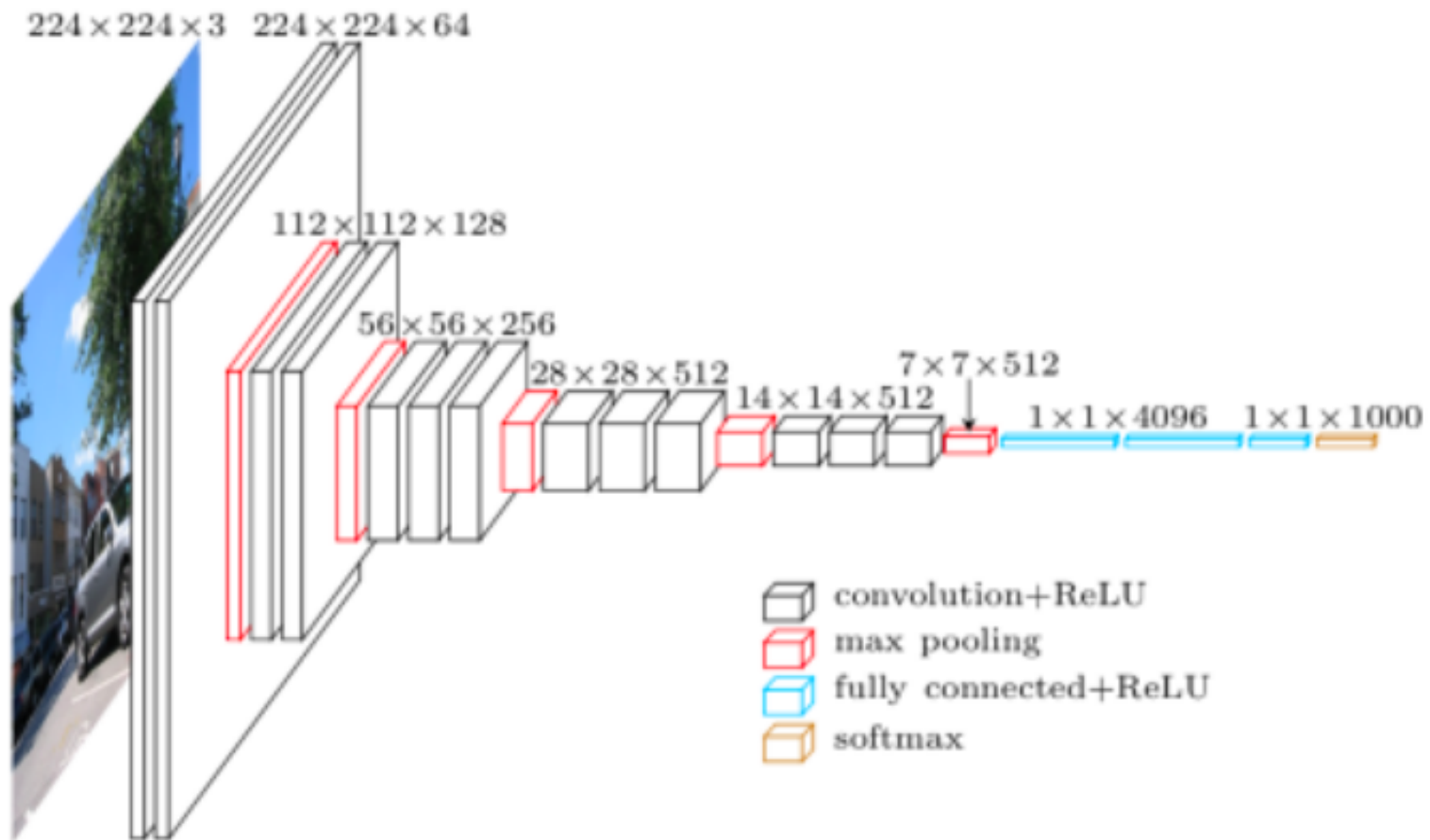
$$L_8[6 \times 6 \times 256] = \text{MaxPool}(L_7, \text{width } 3, \text{stride } 2))$$

$$L_9[4096] = \text{ReLU}(\text{FC}(L_8, \Phi_9))$$

$$L_{10}[4096] = \text{ReLU}(\text{FC}(L_9, \Phi_{10}))$$

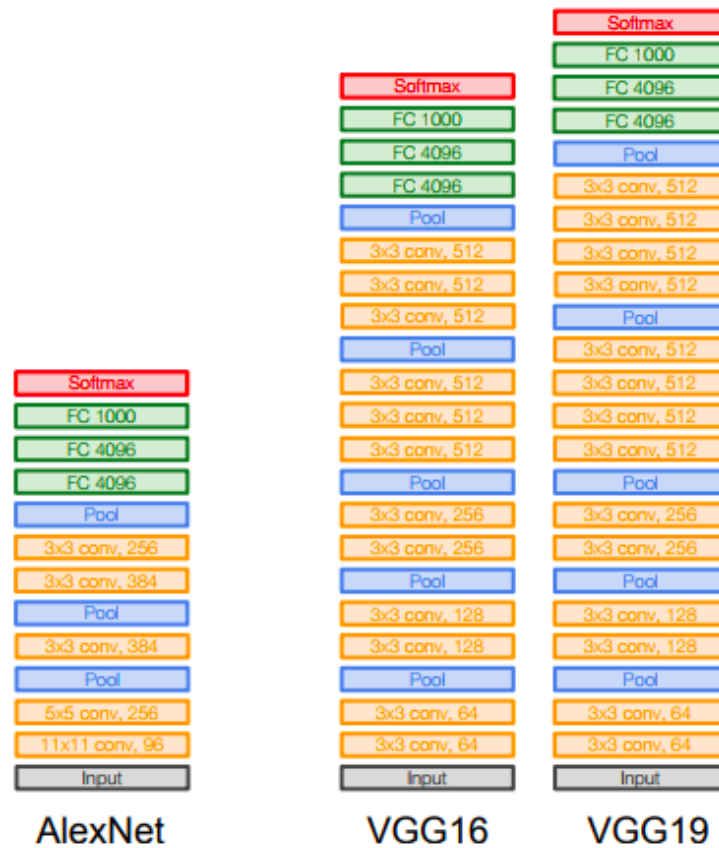
$$s[1000] = \text{ReLU}(\text{FC}(L_{10}, \Phi_s)) \quad \text{class scores}$$

VGG, Zisserman, 2014



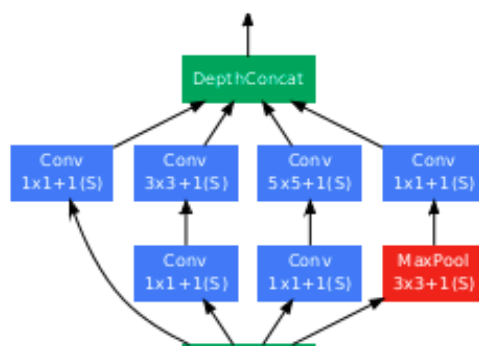
Davi Frossard

VGG



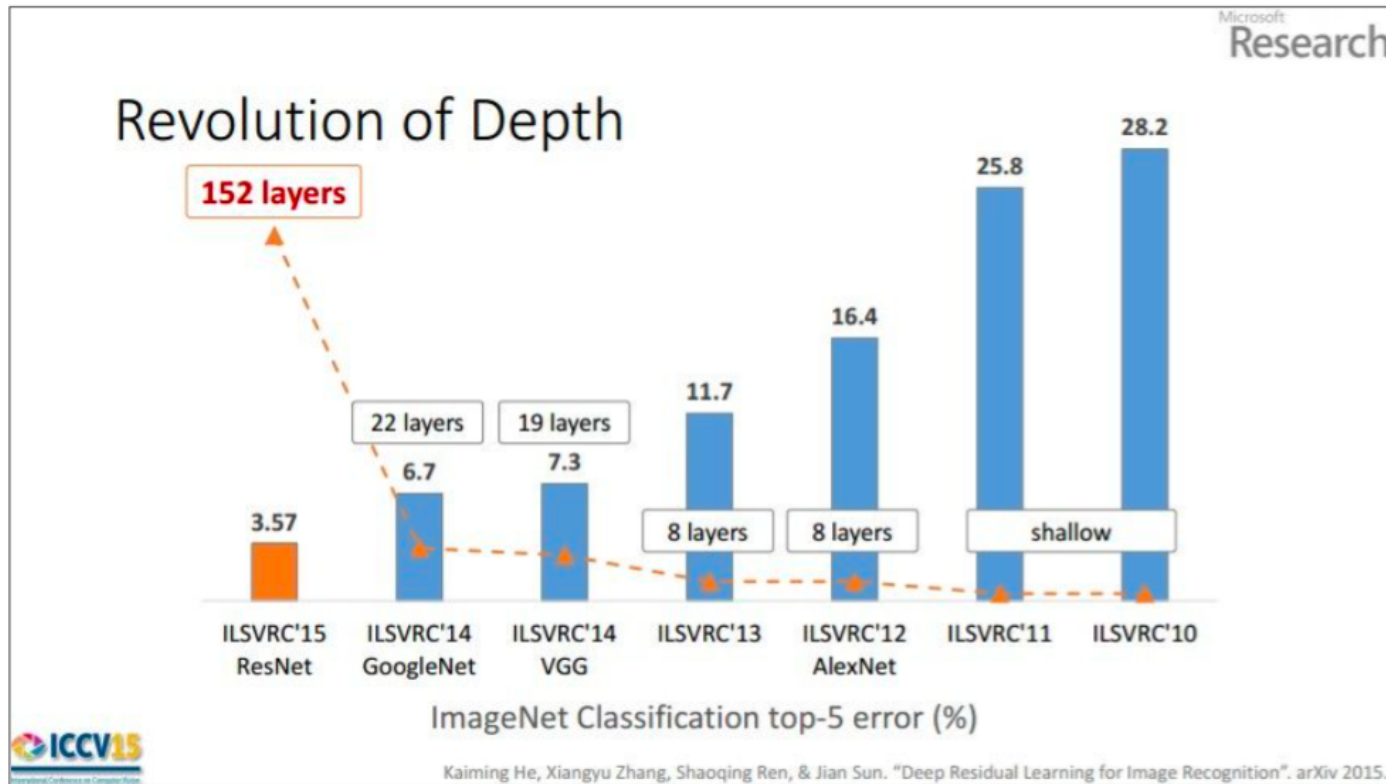
Stanford CS231

Inception, Google, 2014



Imagenet Classification

1000 kinds of objects.



(slide from Kaiming He's recent presentation)

2016 error rate is 3.0%

2017 error rate is 2.25%

Use Swap Rule to get Backward Method

$$L_{\text{out}}[b, x, y, j] \ += \ W[\Delta x, \Delta y, i, j] \ L_{\text{in}}[b, x + \Delta x, y + \Delta y, i]$$

$$W.\text{grad}[\Delta x, \Delta y, i, j] \ += \ \frac{1}{B} \ L_{\text{out}}.\text{grad}[b, x, y, j] \ L_{\text{in}}[b, x + \Delta x, y + \Delta y, i]$$

$$L_{\text{in}}.\text{grad}[b, x + \Delta x, y + \Delta y, i, j] \ += \ L_{\text{out}}.\text{grad}[b, x, y, j] \ W[\Delta x, \Delta y, i, j]$$

Image to Column (Im2C)

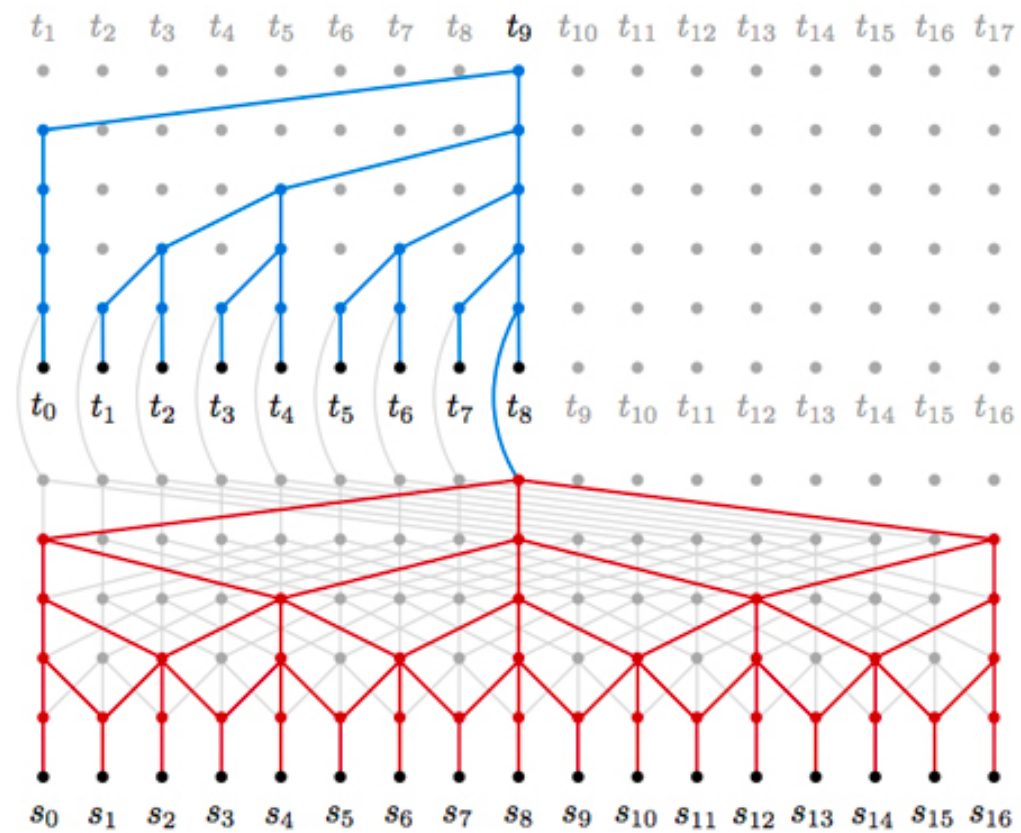
Reduce convolution to matrix multiplication — more space but faster.

$$L_{\text{in}}[b, x, y, \Delta x, \Delta y, i] = L_{\text{in}}[b, x + \Delta x, y + \Delta y, i]$$

$$L_{\text{out}}[b, x, y, j]$$

$$\begin{aligned} &= \left(\sum_{\Delta x, \Delta y, i} W[\Delta x, \Delta y, i, j] * L_{\text{in}}[b, x + \Delta x, y + \Delta y, i] \right) + B[j] \\ &= \left(\sum_{\Delta x, \Delta y, i} L_{\text{in}}[b, x, y, \Delta x, \Delta y, i] * W[\Delta x, \Delta y, i, j] \right) + B[j] \\ &= \left(\sum_{(\Delta x, \Delta y, i)} L_{\text{in}}[(b, x, j), (\Delta x, \Delta y, i)] * W[(\Delta x, \Delta y, i), j] \right) + B[j] \end{aligned}$$

Fully Convolutional Networks



Dilation

We can “dilate” the filter by introducing an image step size d for each step in the filter coordinates.

$$L_{\text{out}}[b, x, y, j] = W[\Delta x, \Delta y, i, j] L_{\text{in}}[b, x + d * \Delta x, y + d * \Delta y, i] + B[j]$$

This is used for “fully convolutional” CNNs.

END