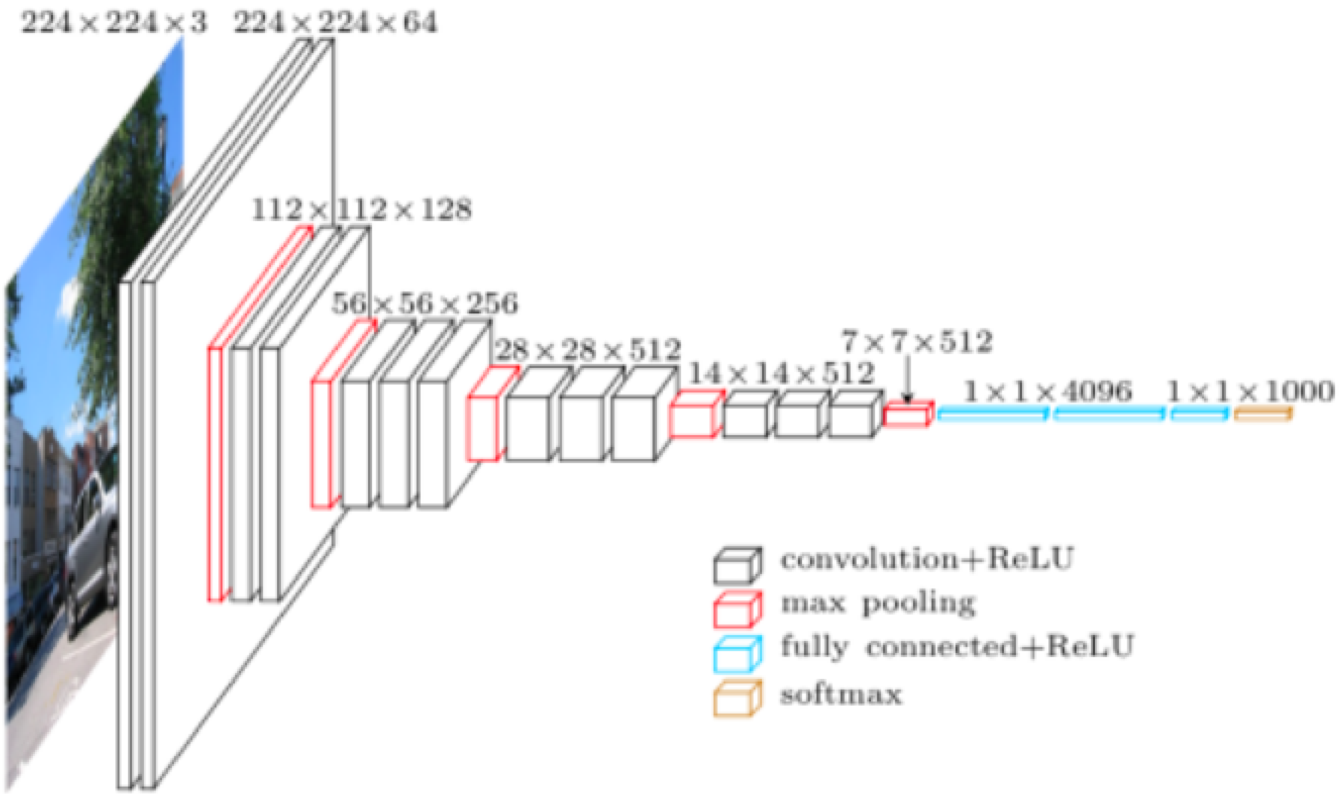
TTIC 31230, Fundamentals of Deep Learning

David McAllester, Winter 2019

Convolutional Neural Networks (CNNs)

What is a CNN?

VGG, Zisserman, 2014



Davi Frossard

Review: Einstein Notation for Linear Threshold Layer

$$y = \sigma\left(Wx - B\right)$$

is an abbreviation for

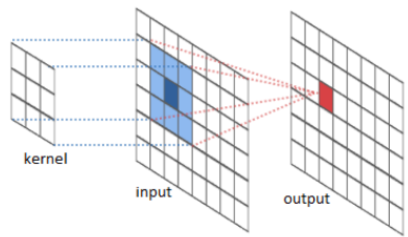
$$y[b,j] = \sigma\left(\left(\sum_i W[j,i]\, x[b,i]\right) - B[j]\right)$$

Think of this as a separate assignment statement for each (b, j).

Each $y[b,j]$ is the output of a "linear threshold unit".

Einstein notation makes all indeces and summations explicit.

A Convolution Layer



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

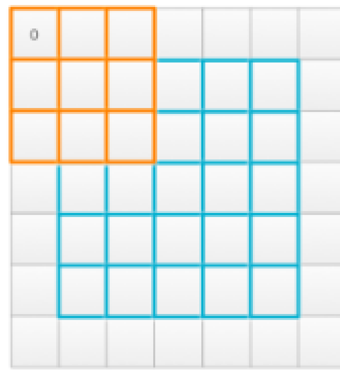
River Trail Documentation

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

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

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

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

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

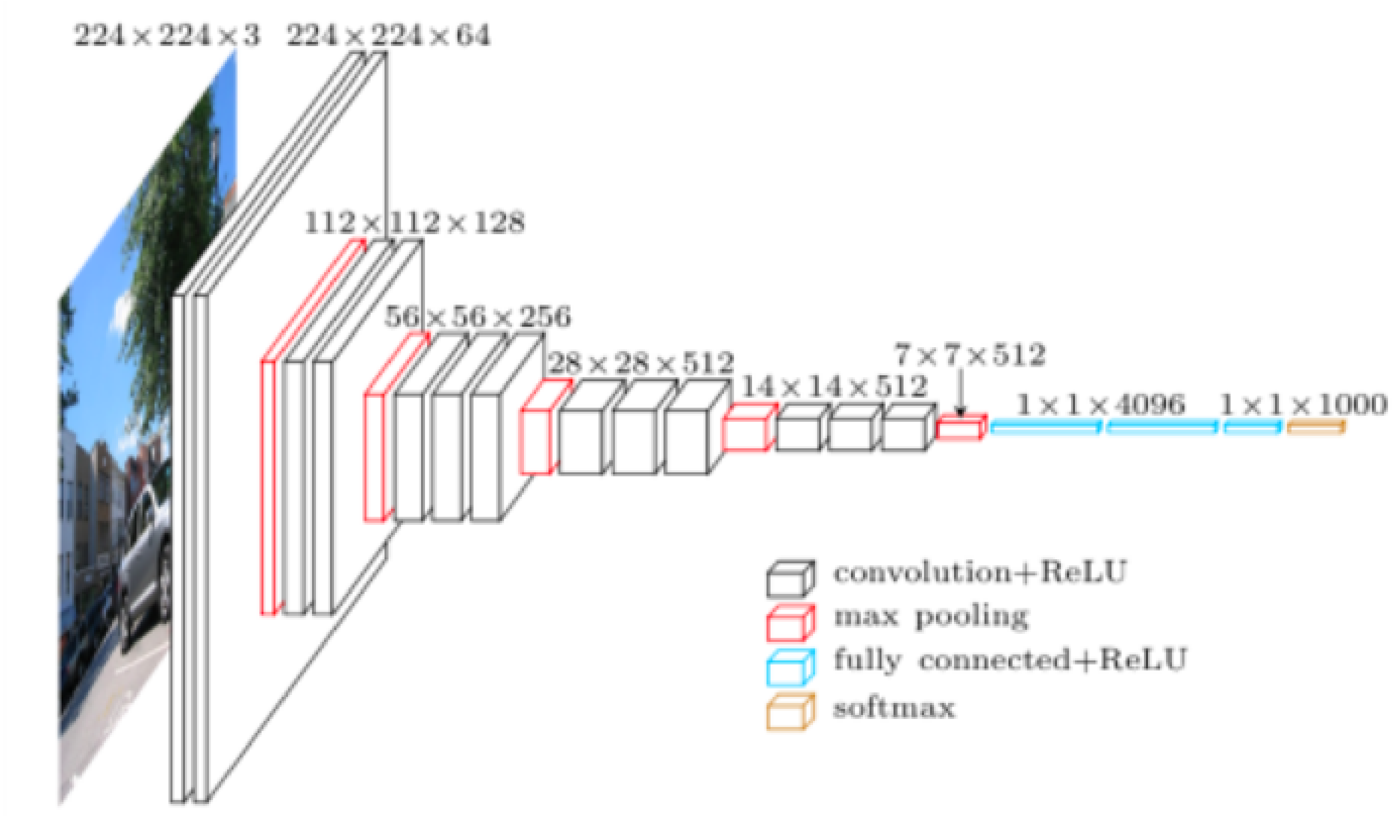
$L'_{\text{in}} = \text{Padd}(L_{\text{in}},\ p)$

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

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

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

Reducing Spatial Dimention



Strides

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

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

$$\sigma\left(\left(\sum_{\Delta x, \Delta y, j} W[\Delta x, \Delta y, i, j] L_{\mathrm{in}}[b, s * x + \Delta x, s * y + \Delta y, i]\right) - B[j]\right)$$

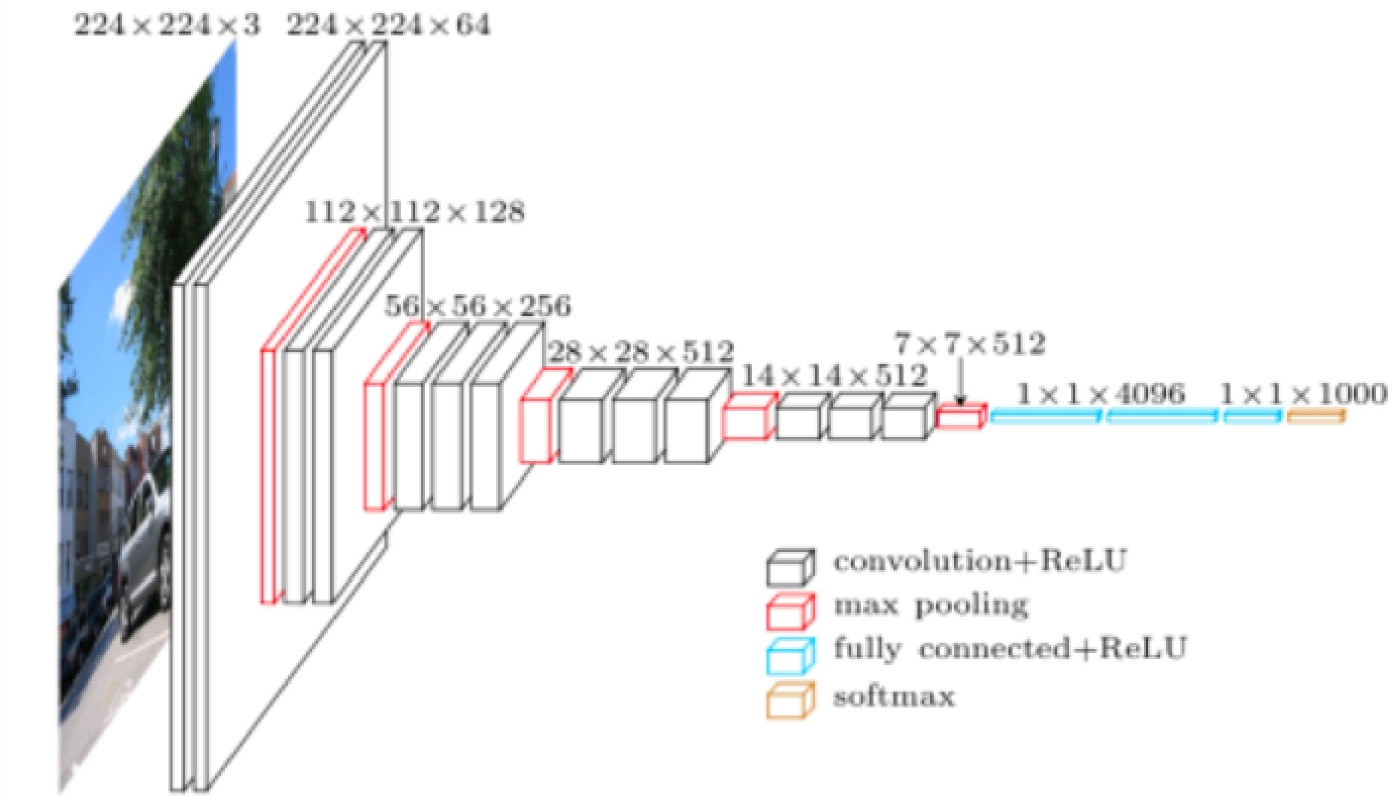
For strides greater than 1 the spatial dimention is reduced.

Max Pooling

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

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

Fully Connected (FC) Layers



Fully Connected (FC) Layers

We reshape $L_{\mathrm{in}}[b,x,y,i]$ to $L_{\mathrm{in}}[b,i']$ and then

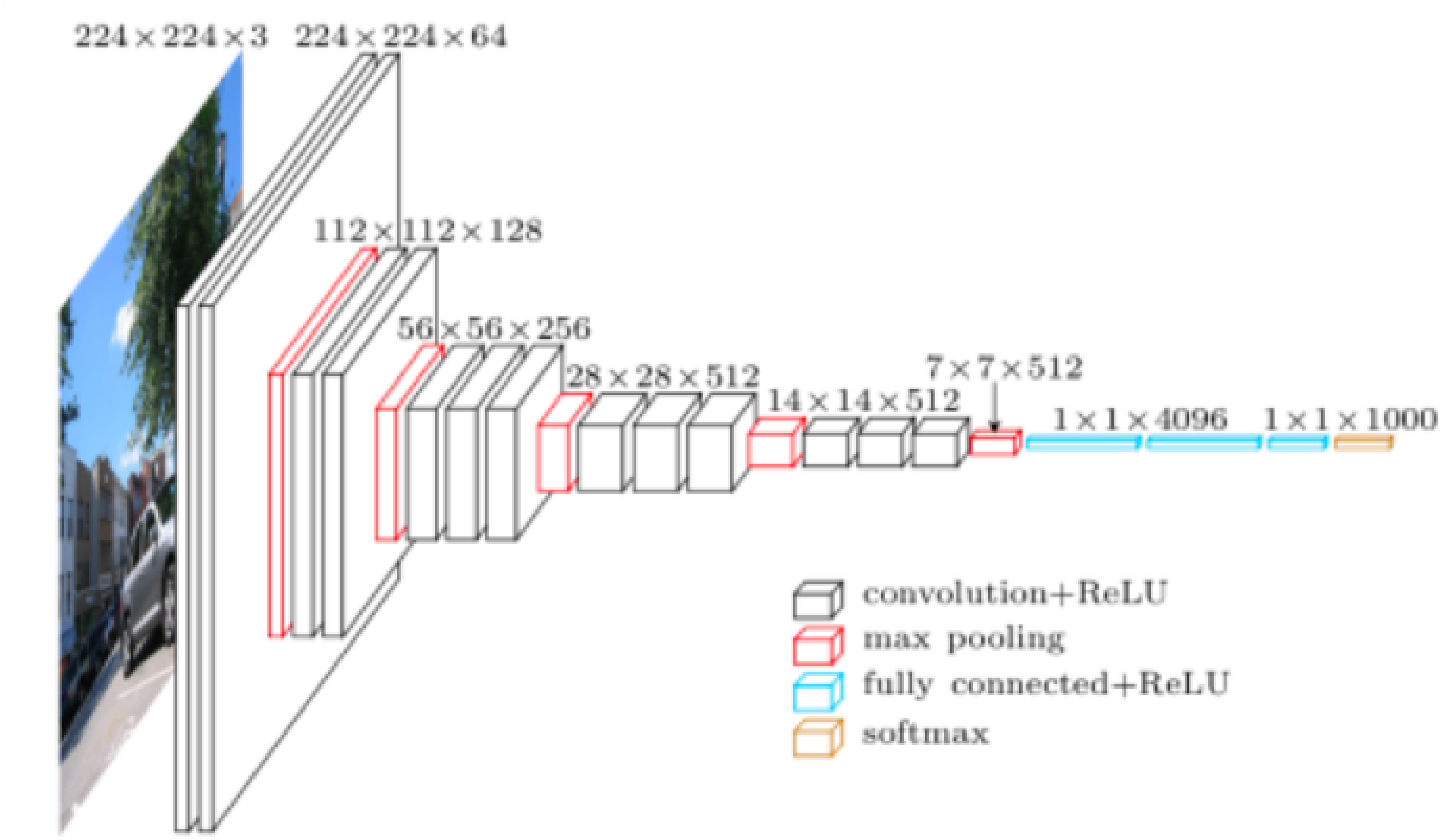
$$L_{\mathrm{out}}[b,j] = \sigma\left(\left(\sum_{i'} W[j,i]\ L_{\mathrm{in}}[b,i']\right) - B[j]\right)$$

Alexnet

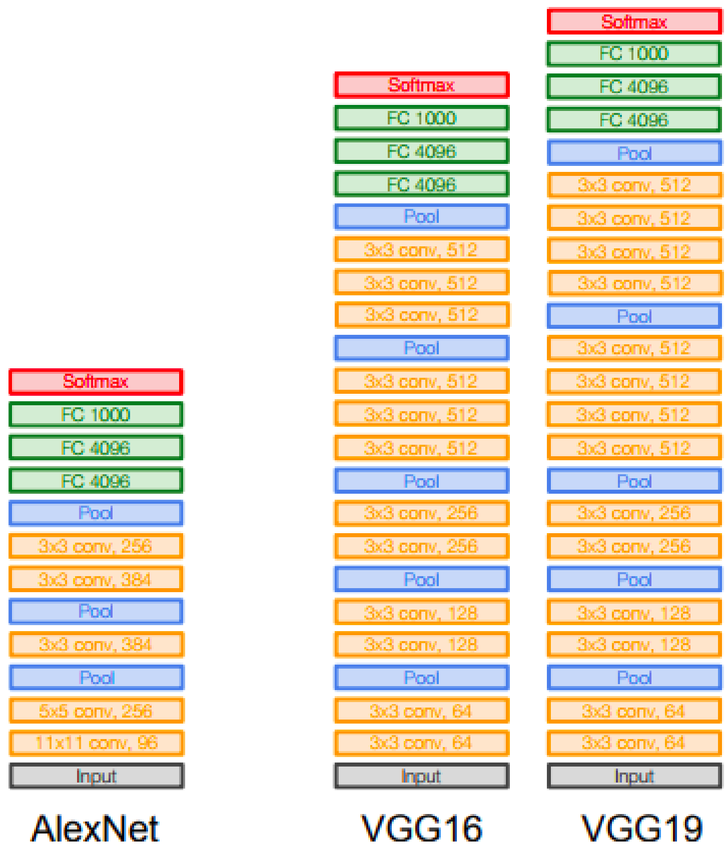
Given $\text{Input}[227, 227, 3]$

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

VGG, Zisserman, 2014

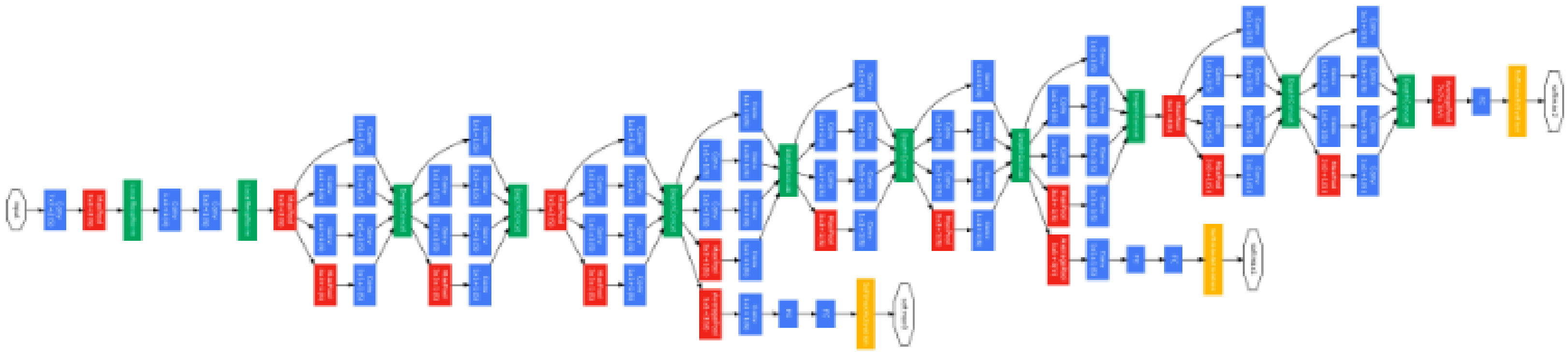


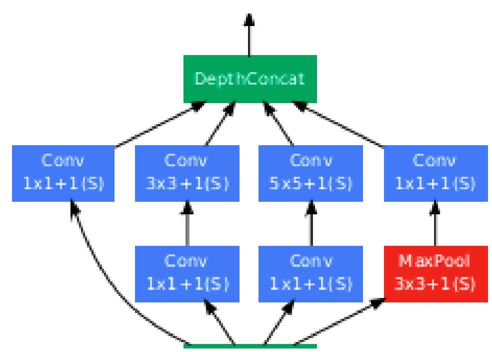
VGG



Stanford CS231

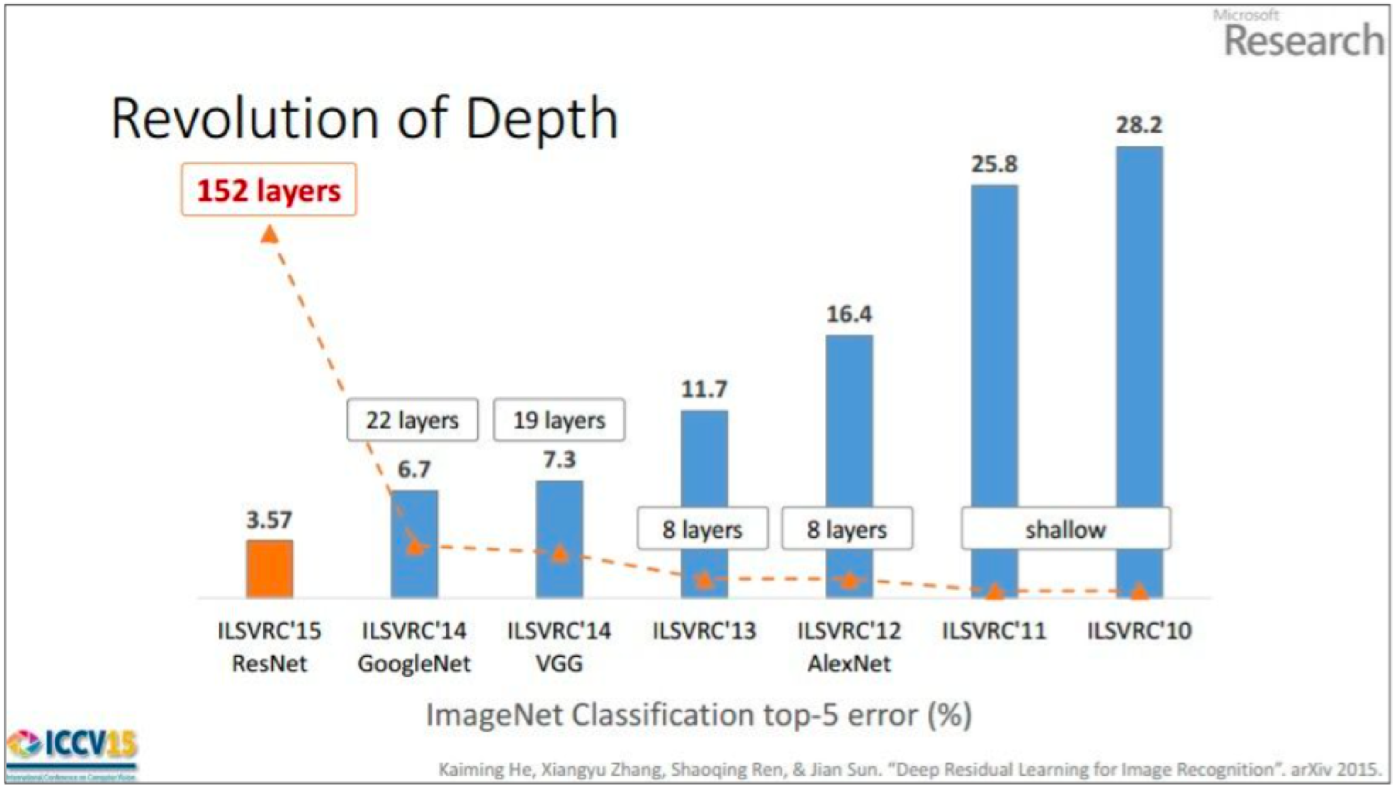
Inception, Google, 2014





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%

Review of The Swap Rule

$$\tilde{y}[b,j] = \sum_i W[j,i]\, x[b,i]$$

$$y[b,j] = \sigma(\tilde{y}[b,j] - B[j])$$

$$x.\mathrm{grad}[b,i] \mathrel{+}= \sum_j \tilde{y}.\mathrm{grad}[b,j] W[j,i]$$

$$W.\mathrm{grad}[j,i] \mathrel{+}= \sum_b \tilde{y}.\mathrm{grad}[b,j] x[b,i]$$

Alternative Swap Rule

Intialize all computed tensors to zero and write the program using only +=.

$$\text{for } b, i, j \qquad \tilde{y}[b,j] \mathrel{+}= W[j,i]\; x[b,i]$$

$$\text{for } b, i, j \quad x.\mathrm{grad}[b,i] \mathrel{+}= \tilde{y}.\mathrm{grad}[b,j] W[j,i]$$

$$\text{for } b, i, j \quad W.\mathrm{grad}[j,i] \mathrel{+}= \tilde{y}.\mathrm{grad}[b,j] x[b,i]$$

one swaps the output with one of the inputs inside the body of the loop.

Alternative Swap Rule for Convolution

for $b, x, y, i, j, \Delta x, \Delta y$

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

for $b, x, y, i, j, \Delta x, \Delta y$

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

for $b, x, y, i, j, \Delta x, \Delta y$

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

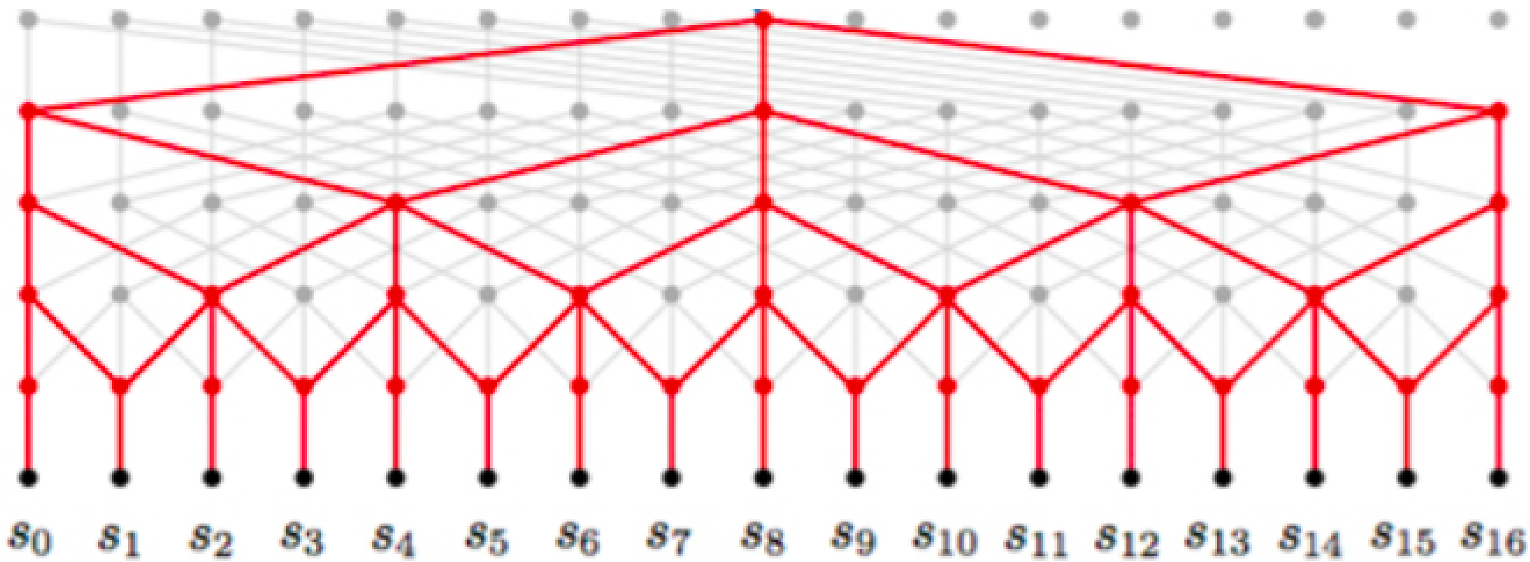
Image to Column (Im2C)

Reduce convolution to matrix multiplication — more space but faster.

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

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

Fully Convolutional Networks



Dilation

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

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

This is used for "fully convolutional" CNNs.

Summary

- Convolution
- Padding
- Stides
- Max Pooling
- Fully Connected Layers
- Dilation

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 (next lecture).

END