Convolutional Neural Networks

Overview

Last time:

- Multilayer perceptron
- Backpropagation

Today:

- Review of MLPs/Backpropagation
- CNNs: neural networks for images

Note

- There are no lecture notes for today’s material

Notation

$$z^{(\ell)} = z^{(\ell)}(x)$$
$$a^{(\ell)} = a^{(\ell)}(x)$$

- Input layer

$$\boldsymbol{z}^{(0)} = x \in \mathbb{R}^d$$

- Hidden layers: $1 \leq \ell < L$

$$\boldsymbol{a}^{(\ell)} = W^{(\ell)} z^{(\ell-1)} \quad \text{where} \quad W^{(\ell)} = \underbrace{\left[w_{ij}^{(\ell)} \right]}_{d_\ell \times d_{\ell-1}}$$

$$\boldsymbol{z}^{(\ell)} = \sigma(a^{(\ell)}) \text{ applied elementwise}$$

where d_ℓ = # of nodes in layer ℓ

- If bias desired, prepend a 1 to any $\boldsymbol{z}^{(\ell)}$, $0 \leq \ell < L$, and add column of weights to beginning of $\boldsymbol{W}^{(\ell+1)}$

- Output layer

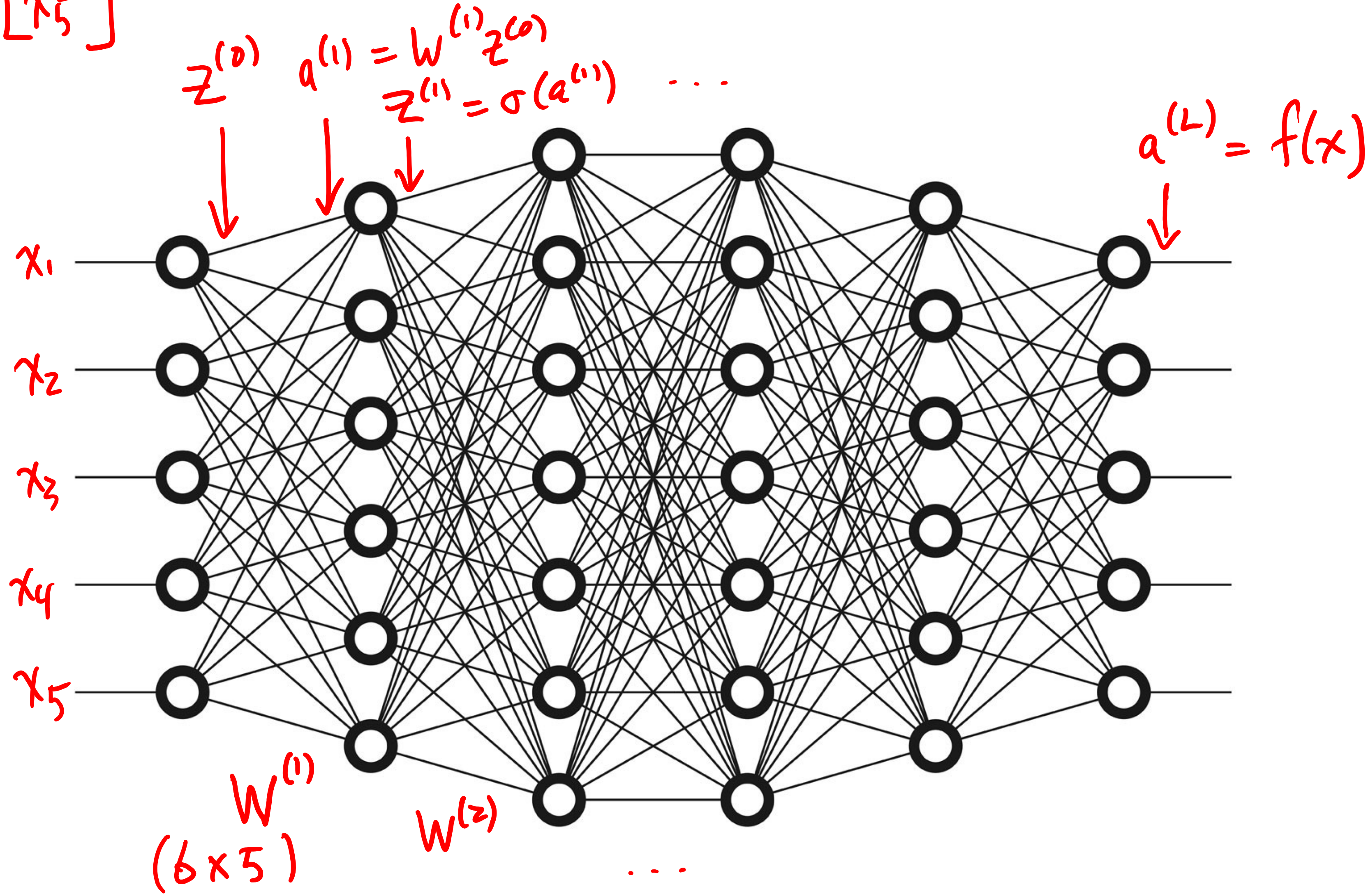
$$f(x) = a^{(L)}$$

followed by *identity activation*

- Evaluation of the output from the input is called forward propagation

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_5 \end{bmatrix} = z^{(0)}$$

Illustration of Notation



Backprop

Forward pass:

Using current weights $\boldsymbol{\theta}$ compute $f(\boldsymbol{x}_n)$

and store intermediate values $a_{ni}^{(\ell)}, z_{nj}^{(\ell)}$

Initialize backward pass:

For $i = 1$ to d_L

Compute $\delta_{ni}^{(L)}$

End

Backward pass:

For $\ell = L - 1$ downto 1

For $i = 1$ to d_ℓ

$\delta_{ni}^{(\ell)} = \sum_k \delta_{nk}^{(\ell+1)} w_{ki}^{(\ell+1)} \sigma'(a_{ni}^{(\ell)})$

For $j = 1$ to $d_{\ell-1}$

$\frac{\partial R_n(\boldsymbol{\theta})}{\partial w_{ij}^{(\ell)}} \longleftarrow \delta_{ni}^{(\ell)} z_{nj}^{(\ell-1)}$

$w_{ij}^{(\ell)} \longleftarrow w_{ij}^{(\ell)} - \eta \frac{\partial R_n(\boldsymbol{\theta})}{\partial w_{ij}^{(\ell)}}$

End

End

End

Poll

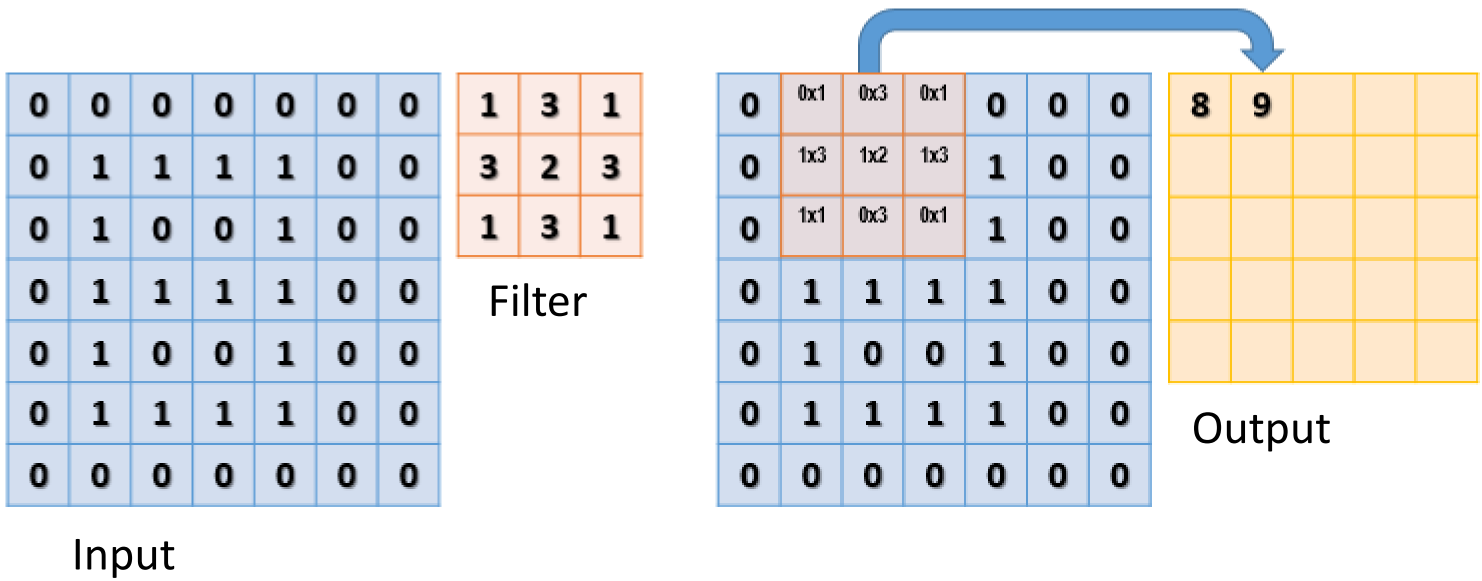
For an MLP with ReLU activation, a subgradient is guaranteed to exist at every iteration of backprop.

(A) True

(B) False

Convolutions

- Basically a sliding window
- Figure assumes a *stride* (shift increment) of 1, but larger strides are possible
- The filter is also called a



Poll

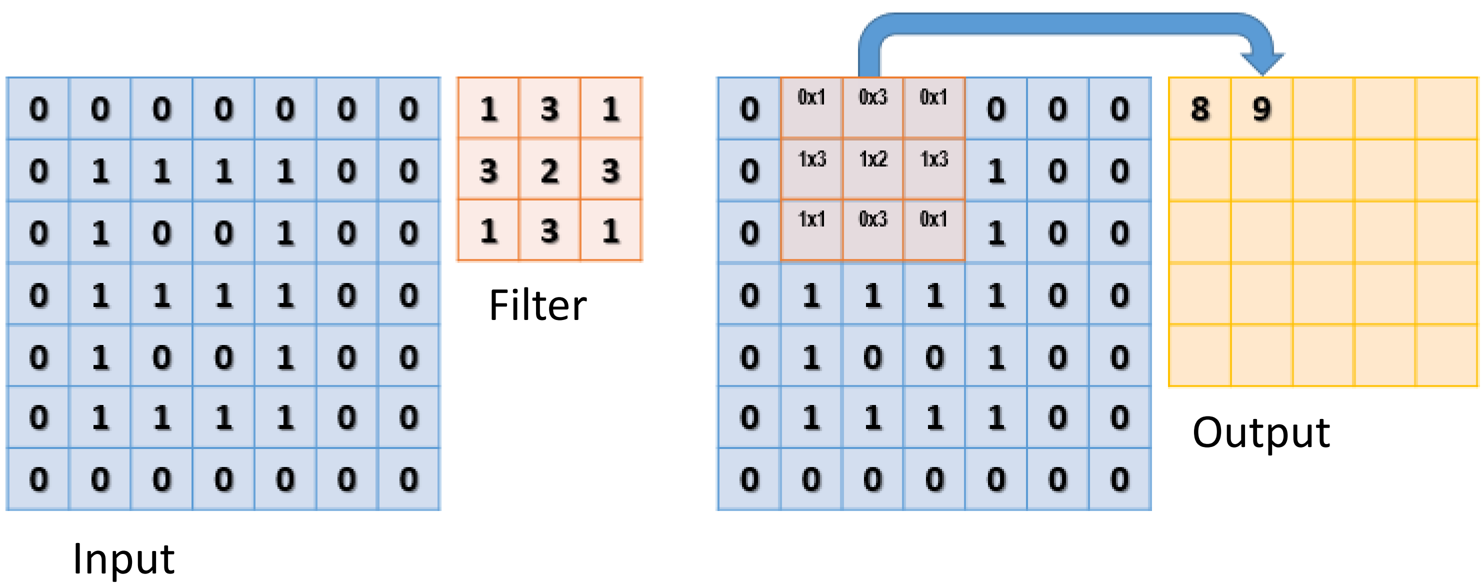
What would be the output size with a stride of 2?

(A) 2×2

(B) 3×3

(C) 4×4

(D) 5×5



Filters are Feature Extractors

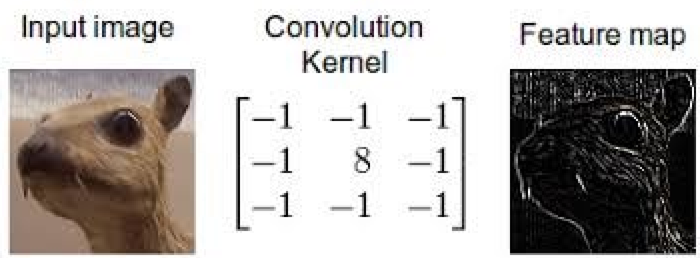


Figure: https://developer.nvidia.com/discover/convolution

Padding

0	0	0	0	0	0	0
0	2	4	9	1	4	0
0	2	1	4	4	6	0
0	1	1	2	9	2	0
0	7	3	5	1	3	0
0	2	3	4	8	5	0
0	0	0	0	0	0	0

Image

$\otimes$

1	2	3
-4	7	4
2	-5	1

Filter / Kernel

=

21	59	37	-19	2
30	51	66	20	43
-14	31	49	101	-19
59	15	53	-2	21
49	57	64	76	10

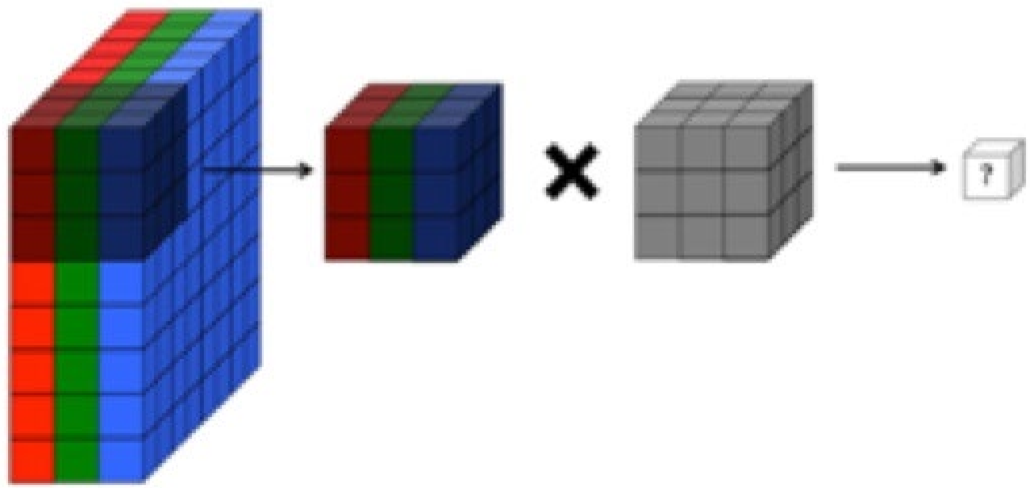
Feature

Demo

https://cs231n.github.io/convolutional-networks/

Channels

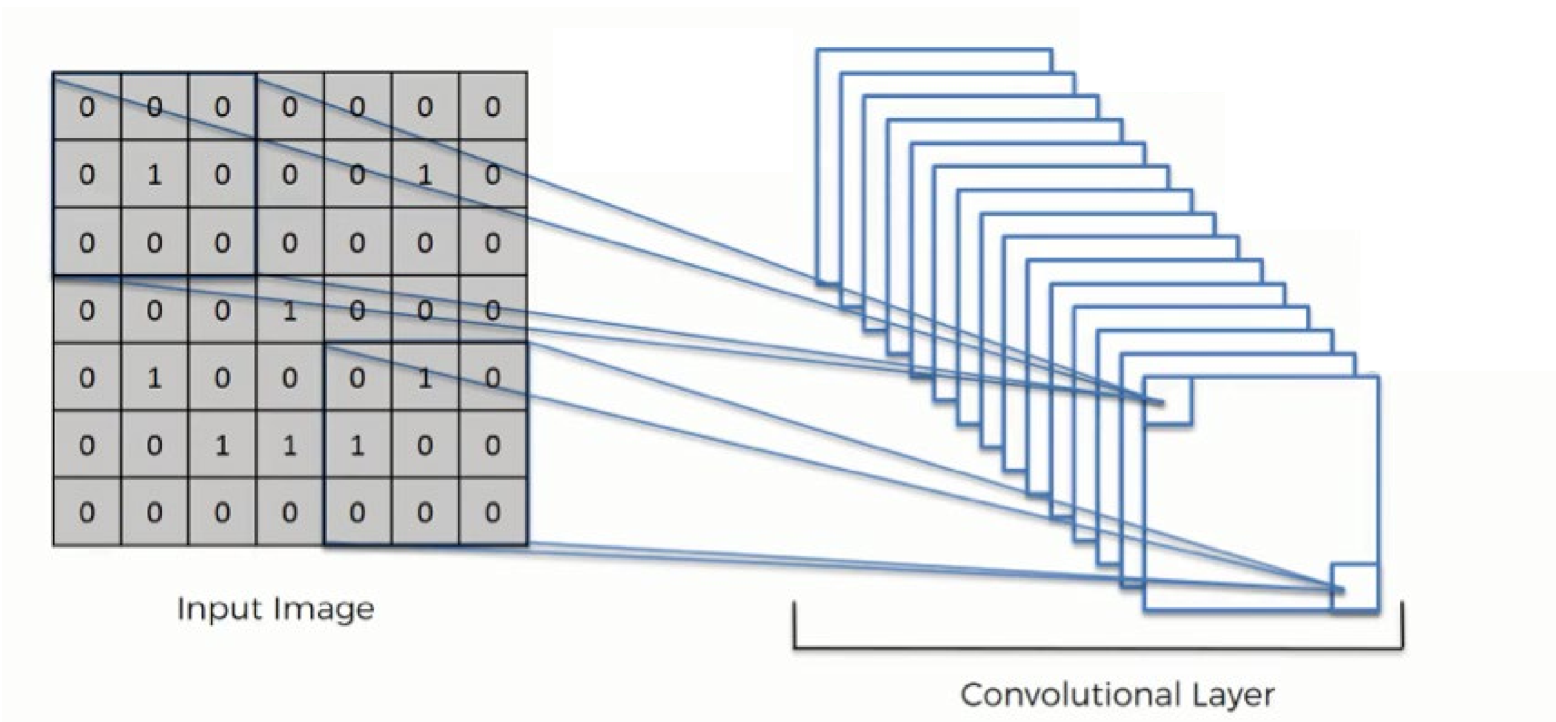
- Color images have three channels.
- When there are multiple channels, the filter is actually three dimensional, where the depth of the fiter is the number of channels.
- The convolution is still *two-dimensional*. Hence the output of the convolution is still two-dimensional.



https://ai.stackexchange.com/questions/5769/in-a-cnn-does-each-new-filter-have-different-weights-for-each-input-channel-or

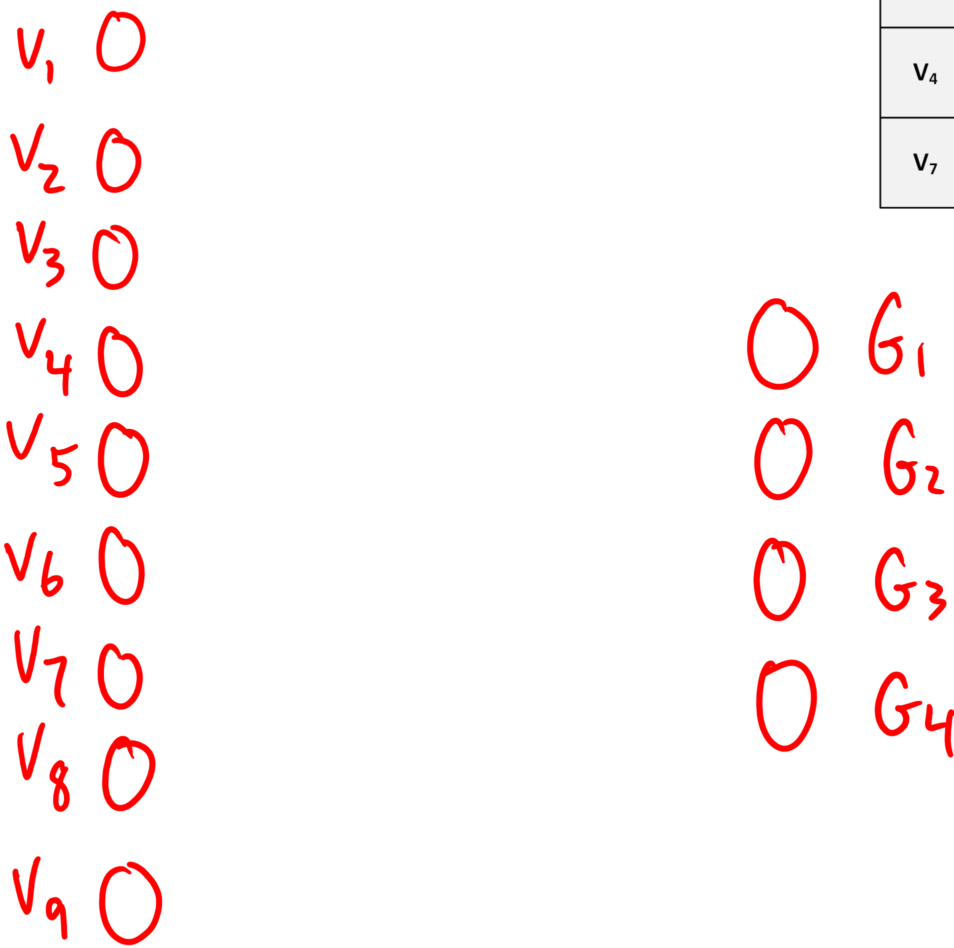
Convolutional Layers

- The initial layer in a CNN is the input image.
- A convolutional layer is a hidden layer formed by applying several convolutions (each with its own filter) to the previous layer.
- Filter coefficients are the weights to be learned
- # of weights into a layer = (filter $H \times W \times D$) $\times$ (# of filters)



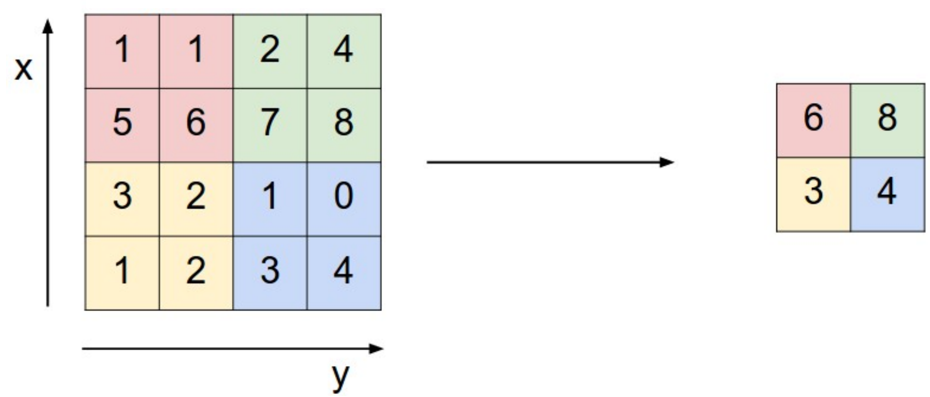
Conv. Layers are not Fully Connected and Weights are Shared

$$\begin{bmatrix} V_1 & V_2 & V_3 \\ V_4 & V_5 & V_6 \\ V_7 & V_8 & V_9 \end{bmatrix} \times \begin{bmatrix} W_1 & W_2 \\ W_3 & W_4 \end{bmatrix} = \begin{bmatrix} G_1 & G_2 \\ G_3 & G_4 \end{bmatrix}$$



Pooling/Downsampling Layer

- Commonly combined with convolutional layers
- Applies to each channel separately
- Common implementation for images: *max pooling*, 2×2 window, stride of 2 (no parameters to learn)
- No weights to learn in the pooling layer itself
- Shrinks layers leading to fewer weights to learn in subsequent layers
- Subsequent learned features have lower spatial resolution, but higher spatial scope



Convolutional Neural Networks

- Combination of convolutional, pooling, and fully connected layers
- Backprop extends naturally to CNNs
- Major breakthrough: LeNet5 (Yann LeCun et. al, 1998) for hand-written digit recognition
- CNNs now used in Facebook's face recognition system, self driving cars, and many other object recognition systems

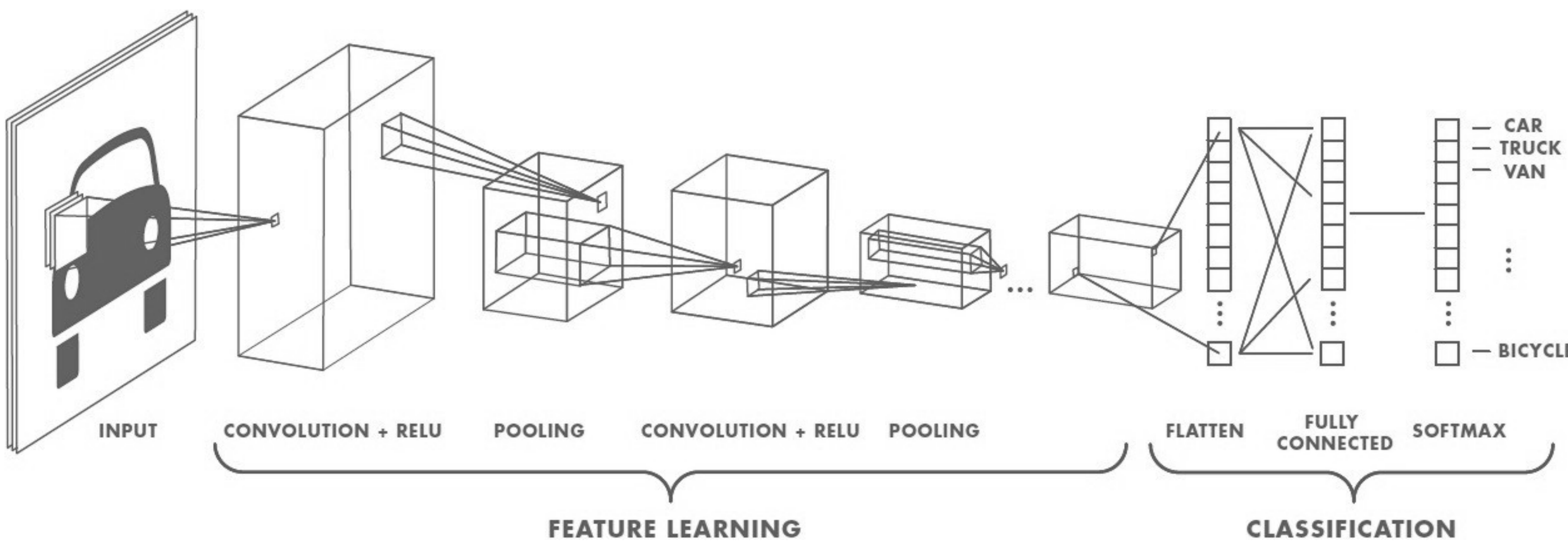
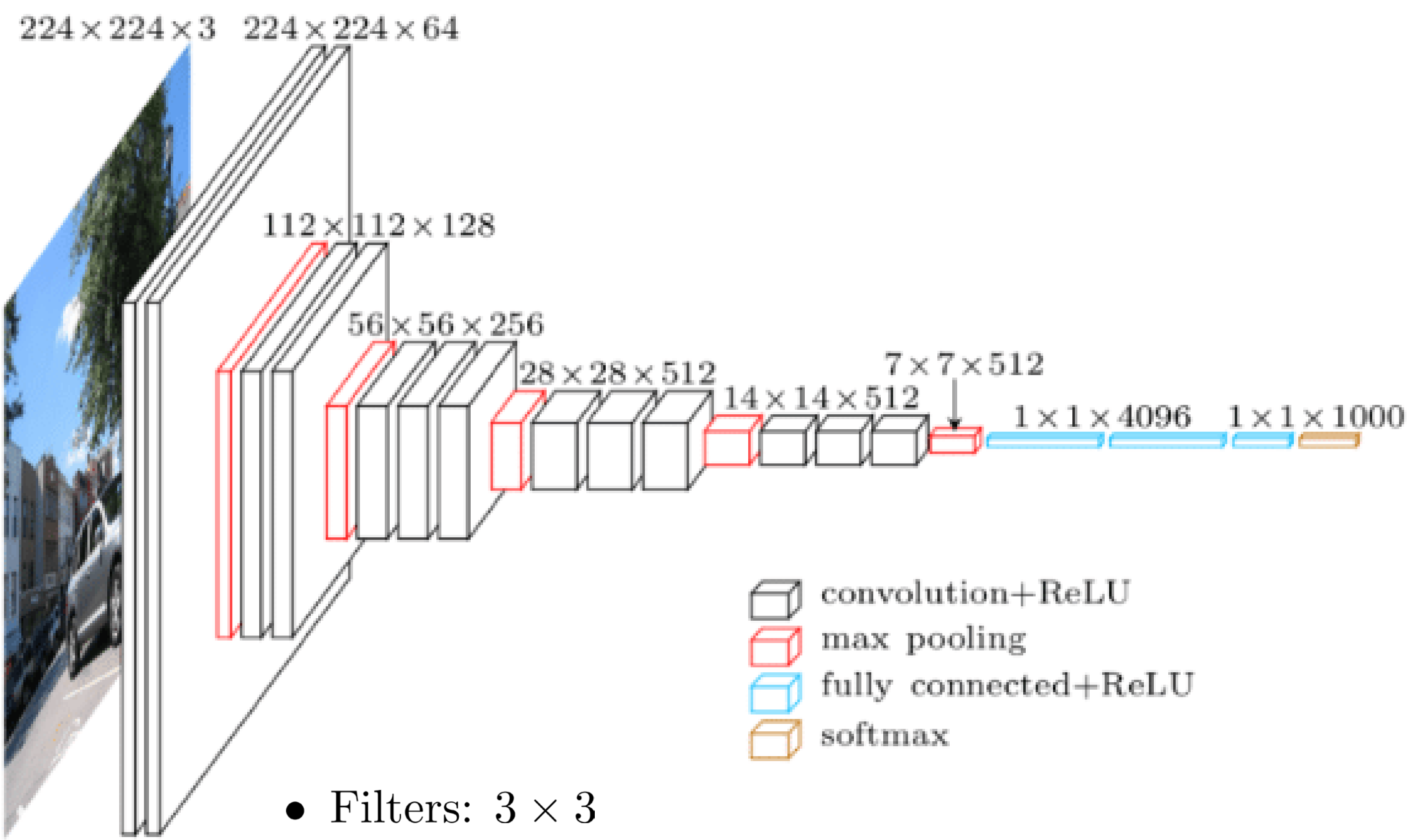


Figure: https://becominghuman.ai/what-exactly-does-cnn-see-4d436d8e6e52

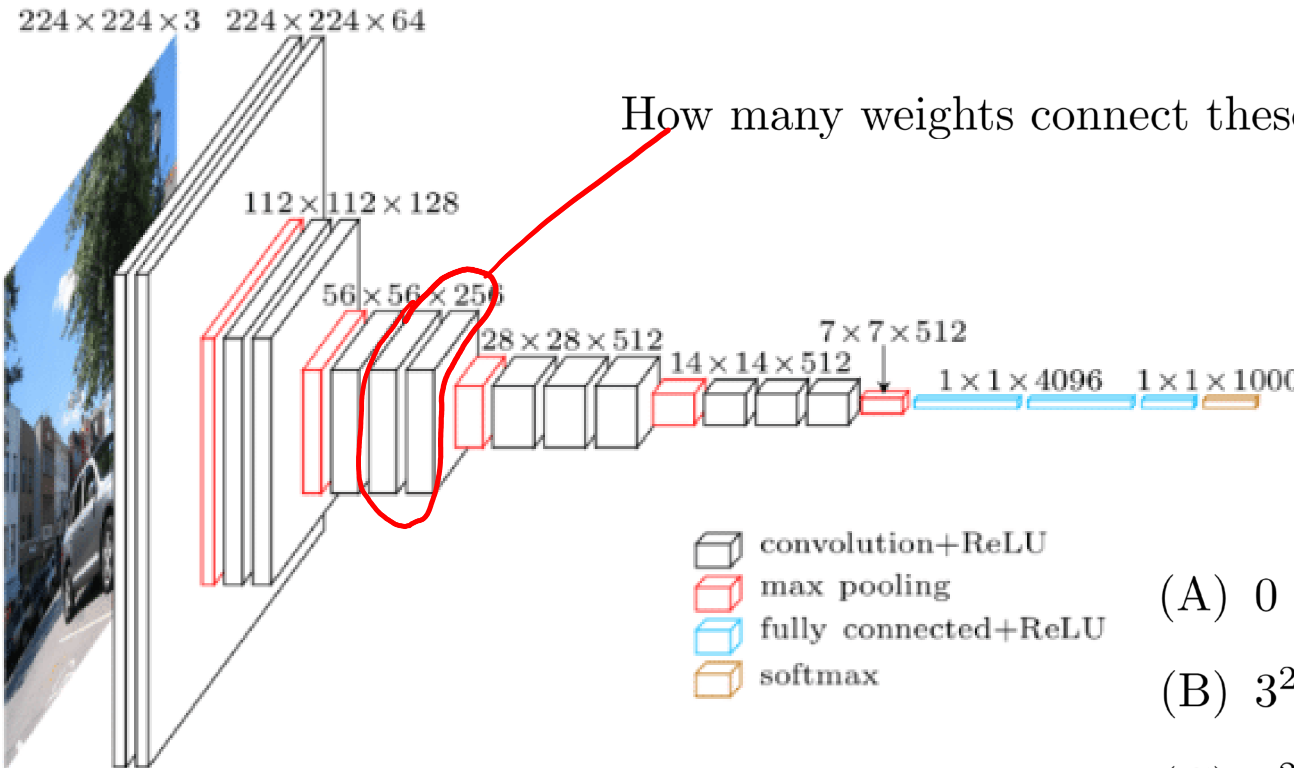
Example: VGG16

VGG16 proposed by K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition”.
Figure: https://neurohive.io/en/popular-networks/vgg16/



- Filters: 3×3

Poll



How many weights connect these two layers?

(A) 0

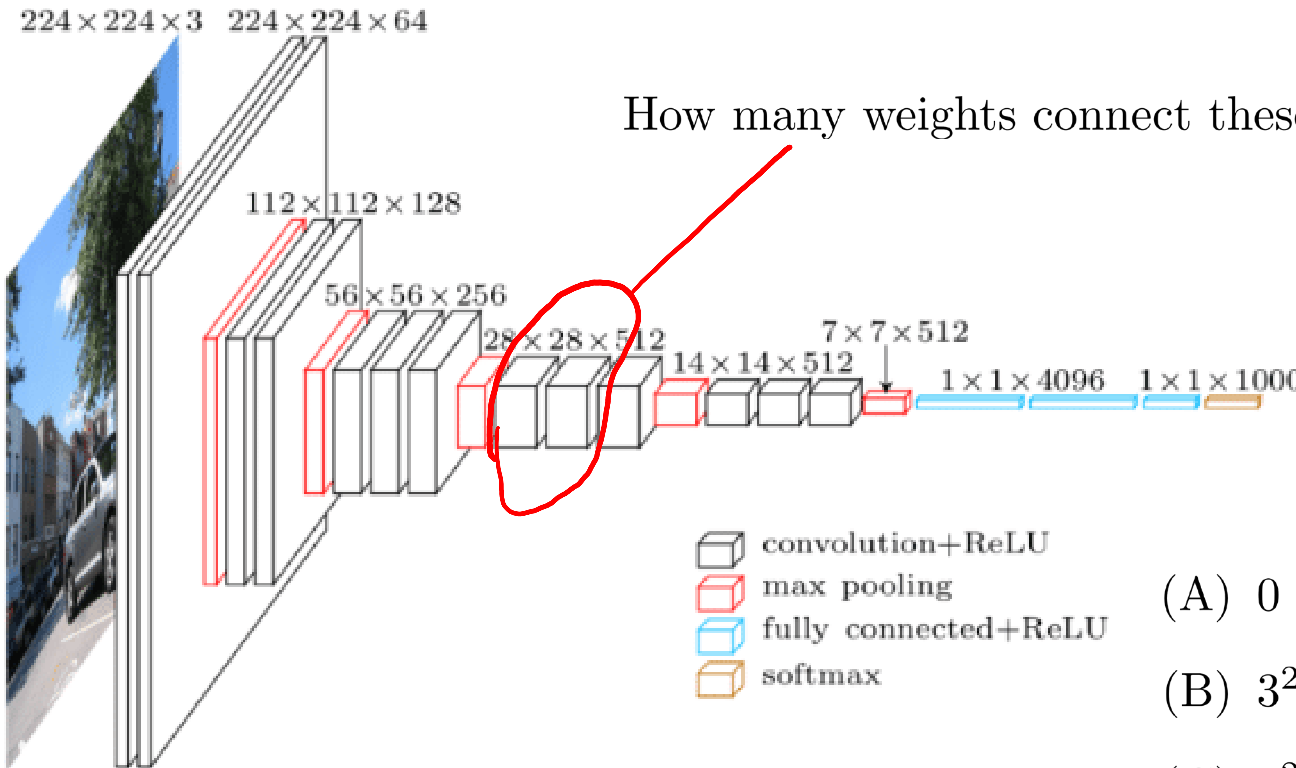
(B) $3^2 \times 256$

(C) $3^2 \times 56^2$

(D) $3^2 \times 256^2$

(E) $3^2 \times 56^2 \times 256^2$

Poll



How many weights connect these two layers?

(A) 0

(B) $3^2 \times 256$

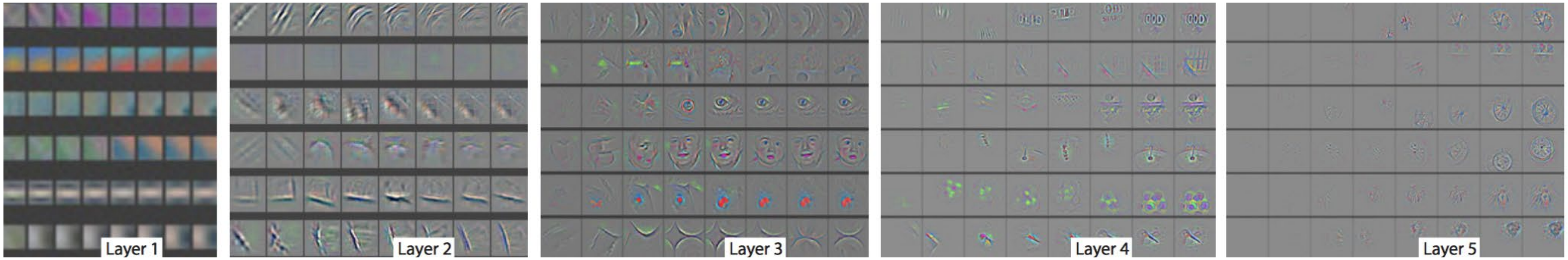
(C) $3^2 \times 56^2$

(D) $3^2 \times 256^2$

(E) $3^2 \times 56^2 \times 256^2$

Learned Filters

- Why convolutions (sliding windows)?
  - Fewer weights (as mentioned previously)
  - Salient features are often spatially localized
- Pooling layers lead to “multi-resolution” features
- Below are some filters from VGG16 (trained on a very large image dataset called ImageNet)
- Layers roughly correspond to level of detail.
- Weights are *learned*



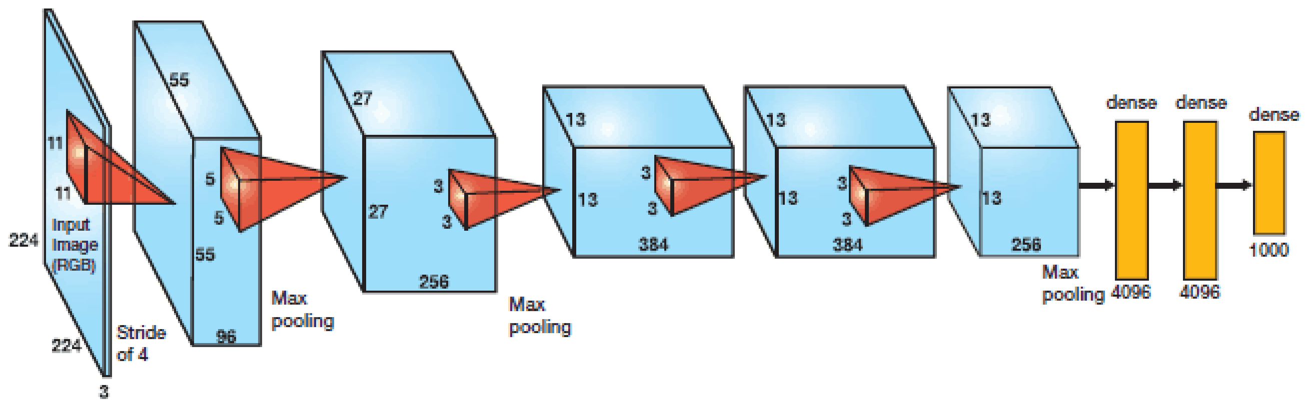
Deep Learning

Enabled by

- Graphics processing units (GPUs): parallel floating-point calculators with 100s-1000s of cores
- Large, public databases such as ImageNet (Fei Fei Li, 2009) which has over 14 million *labeled* examples and 20 thousand classes of objects
- New training strategies
  - Dropout
  - Modifications of SGD (e.g., Adam)
- New architectural elements
  - Residual connections
  - Layer normalization
  - Batch normalization
- Rectified linear units and other activation functions (helps with vanishing gradient problem)

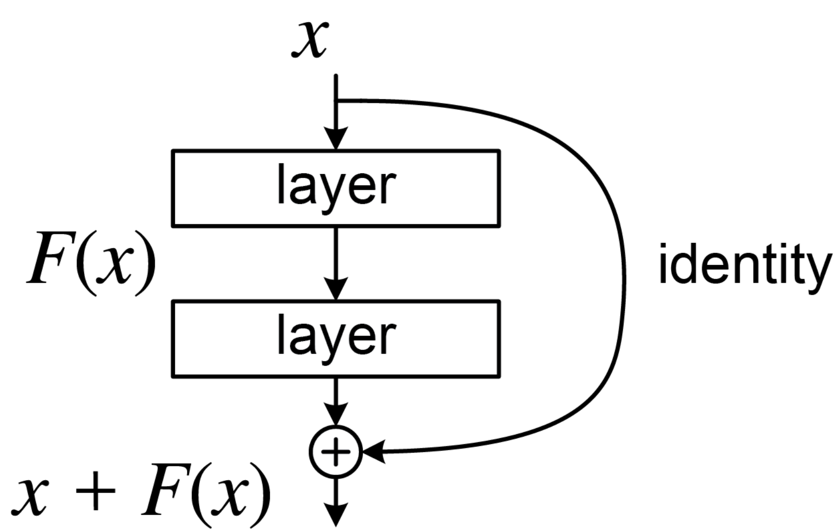
AlexNet

- The big breakthrough
- Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton (2012)
- Reduced error rate on ImageNet from 26% to 16%
- Used GPUs, dropout, ReLU, which have since become standard



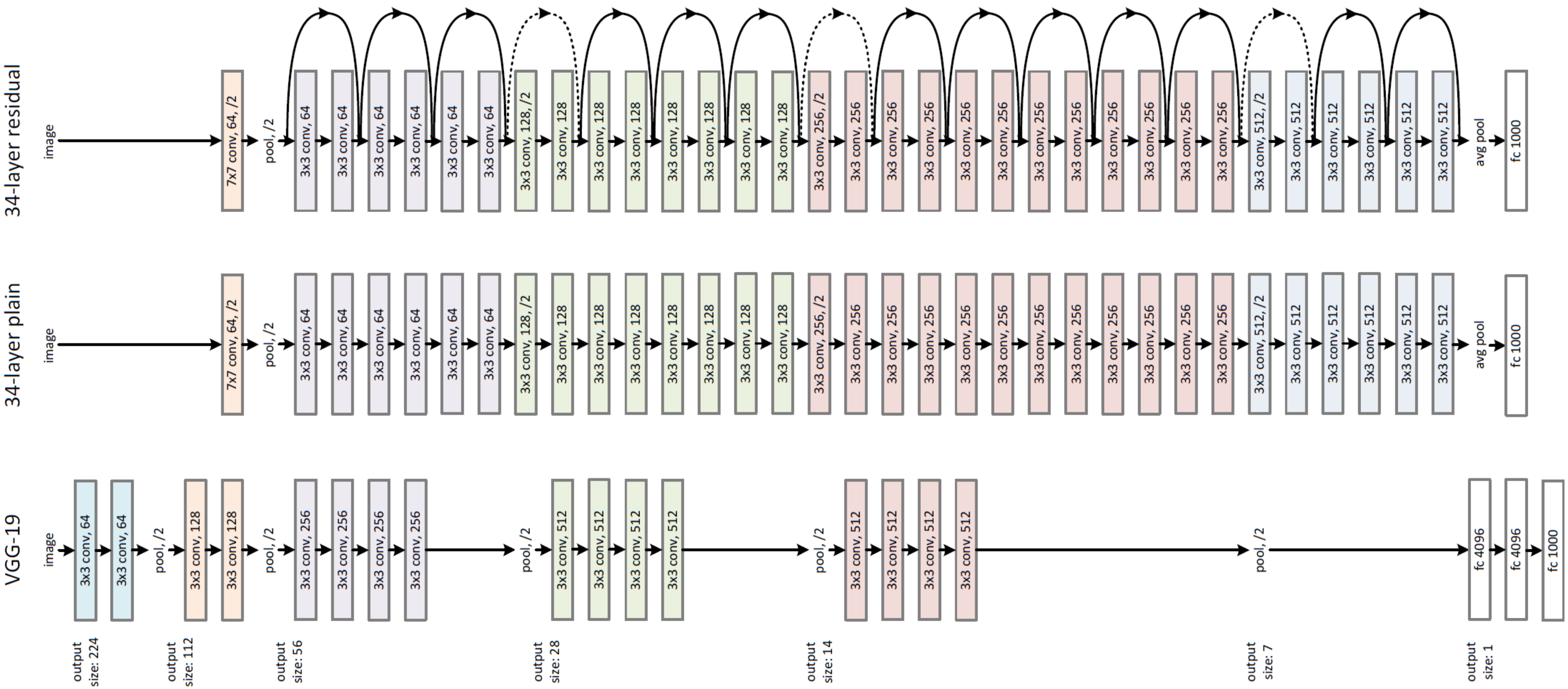
Residual Networks

- "Residual connections"
- Helps with vanishing gradients; gradient propagates directly back to earlier layers



- Addition is performed before applying activation function

ResNet

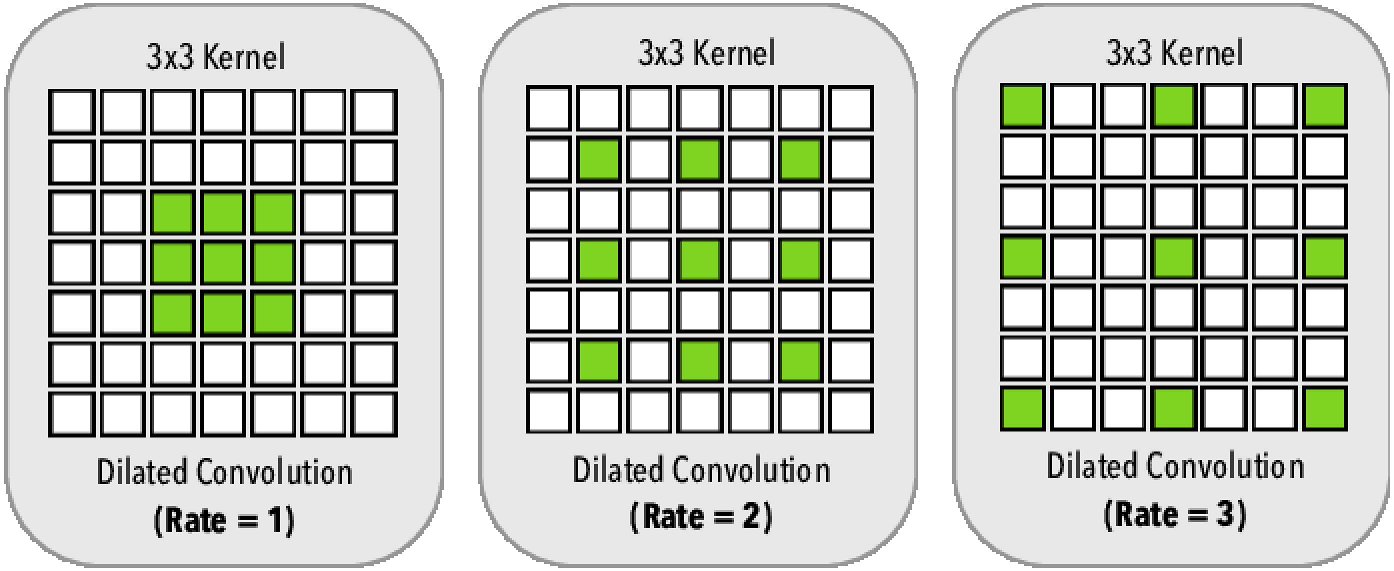


Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, “Deep Residual Learning for Image Recognition,” CVPR 2015

Final Thoughts

- CNNs: neural networks for images
- Feedforward but not fully connected; train with backpropagation

Dilated Convolutions



https://www.researchgate.net/figure/Dilated-convolution-On-the-left-we-have-the-dilated-convolution-with-dilation-rate-r_fig2_320195101A