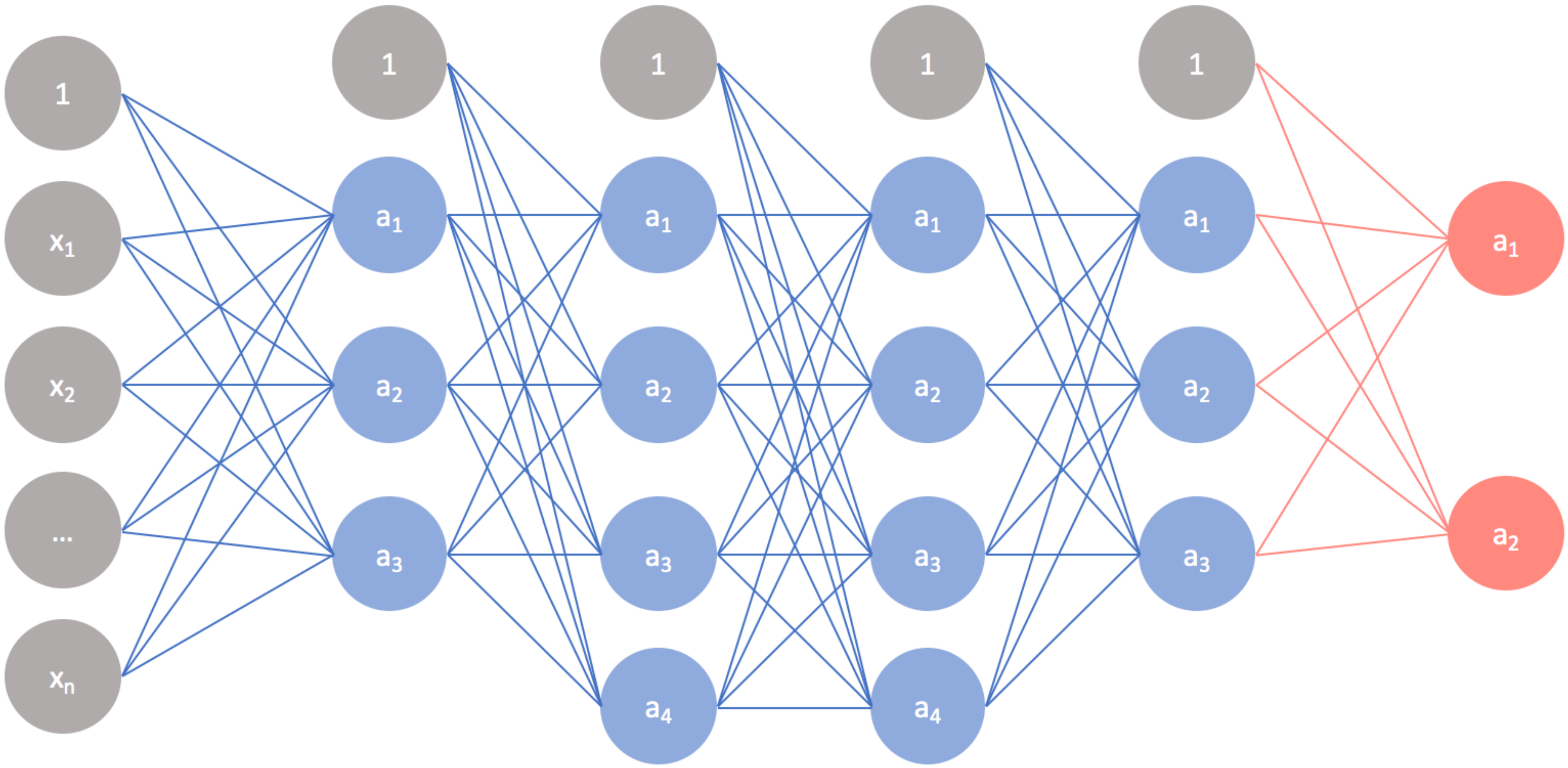
NN basics 2



References

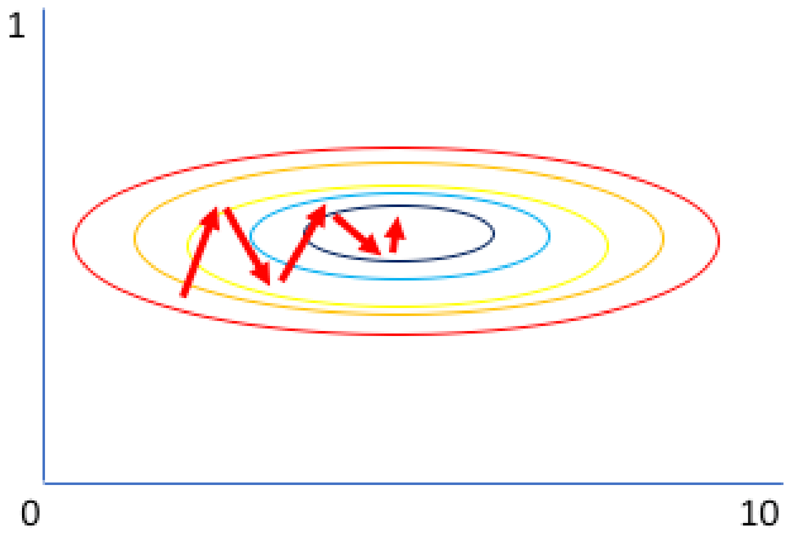
- http://cs231n.stanford.edu/index.html
- http://www.cs.cornell.edu/courses/cs5670/2019sp/lectures/lectures.html
- http://www.cs.cmu.edu/~16385/

A good fully connected example

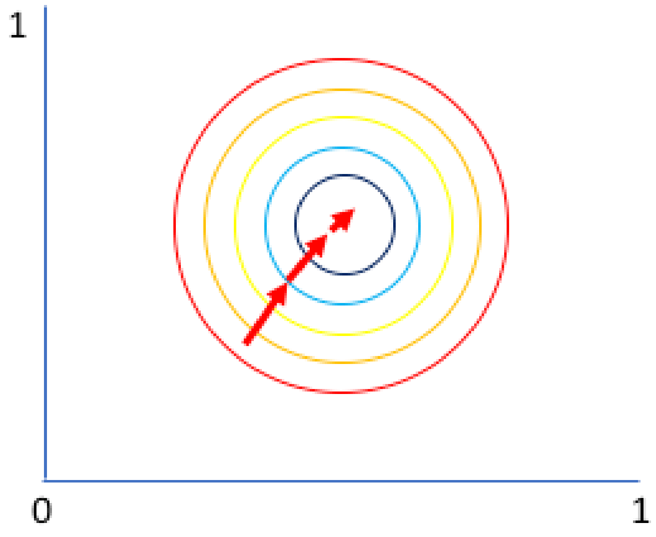
- https://playground.tensorflow.org/#activation=tanh&batchSize=10&dataset=spiral&regDataset=reg-plane&learningRate=0.03&regularizationRate=0&noise=0&networkShape=8,8,8&seed=0.68609&showTestData=false&discretize=false&percTrainData=50&x=true&y=true&xTimesY=true&xSquared=true&ySquared=true&cosX=false&sinX=true&cosY=false&sinY=true&collectStats=false&problem=classification&initZero=false&hideText=false

Data normalization

- Assuming 2D input data with different scales ($x_1 \in [0,1], x_2 \in [0,1000]$)
- The weights needed to make x_1 significant as x_2 are much larger and hence make the loss function ellipsoid like in one direction.
- This will cause the gradient descent method to converge in more steps than if the two axis where at the same scale.



Gradient of larger parameter dominates the update



Both parameters can be updated in equal proportions

Data normalization

- In order to overcome this, we shall normalize the data before the entrance to the NN:

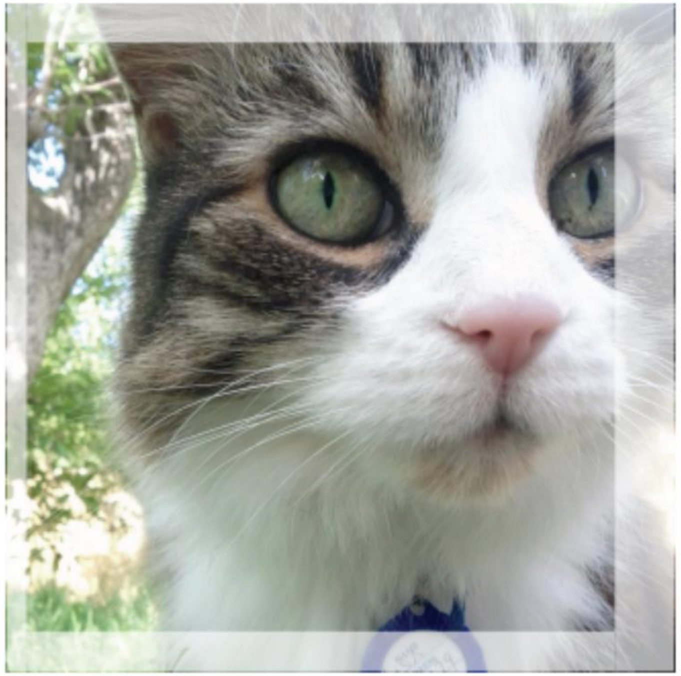
$$\mu = \frac{1}{m}\sum_{i=0}^{m} x_i \,, \sigma^2 = \frac{1}{m-1}\sum_{i=0}^{m}(x_i - \mu)^2$$

$$\widetilde{x_i} = \frac{x_i - \mu}{\sigma^2}$$

- **This should be done for each dimension of the input vector independently.**
- The test data should be normalized with the same variables found in the train data.
- This is a common practice to do even if the data are at the same scale for all dimensions since the default hyperparameters for all NN are based on such normalized data.

Data augmentation

Augment the data — extract random crops from the input, with slightly jittered offsets. Without this, typical ConvNets (e.g. [Krizhevsky 2012]) overfit the data.



E.g. 224x224 patches extracted from 256x256 images

Randomly reflect horizontally

Perform the augmentation live during training

Figure: Alex Krizhevsky

Loss derivation

- How do we actually do $\nabla_W L$?

Bad idea- by hand

$$s = f(x; W) = Wx$$

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

$$= \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1)$$

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i + \lambda \sum_k W_k^2$$

$$= \frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1) + \lambda \sum_k W_k^2$$

$$\nabla_W L = \nabla_W \left(\frac{1}{N} \sum_{i=1}^{N} \sum_{j \neq y_i} \max(0, W_{j,:} \cdot x + W_{y_i,:} \cdot x + 1) + \lambda \sum_k W_k^2 \right)$$

Problem: Very tedious: Lots of matrix calculus, need lots of paper

Problem: What if we want to change loss? E.g. use softmax instead of SVM? Need to re-derive from scratch =(

Problem: Not feasible for very complex models!

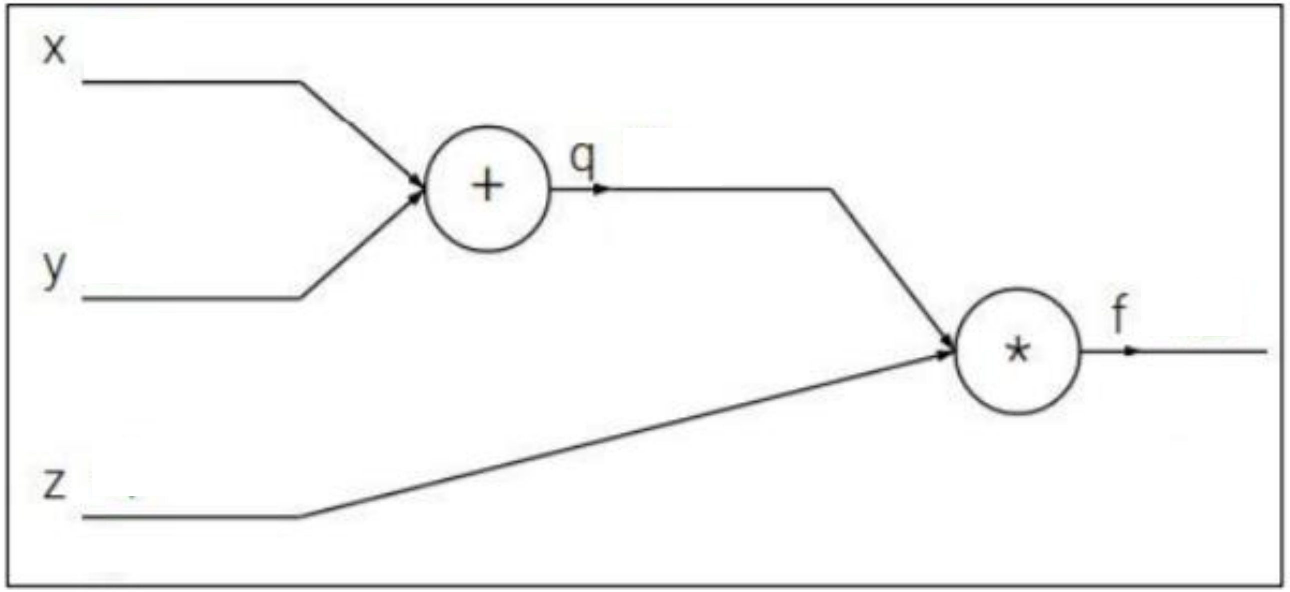
Good idea- back propagation

- We are using the chain rule for gradient calculation:

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

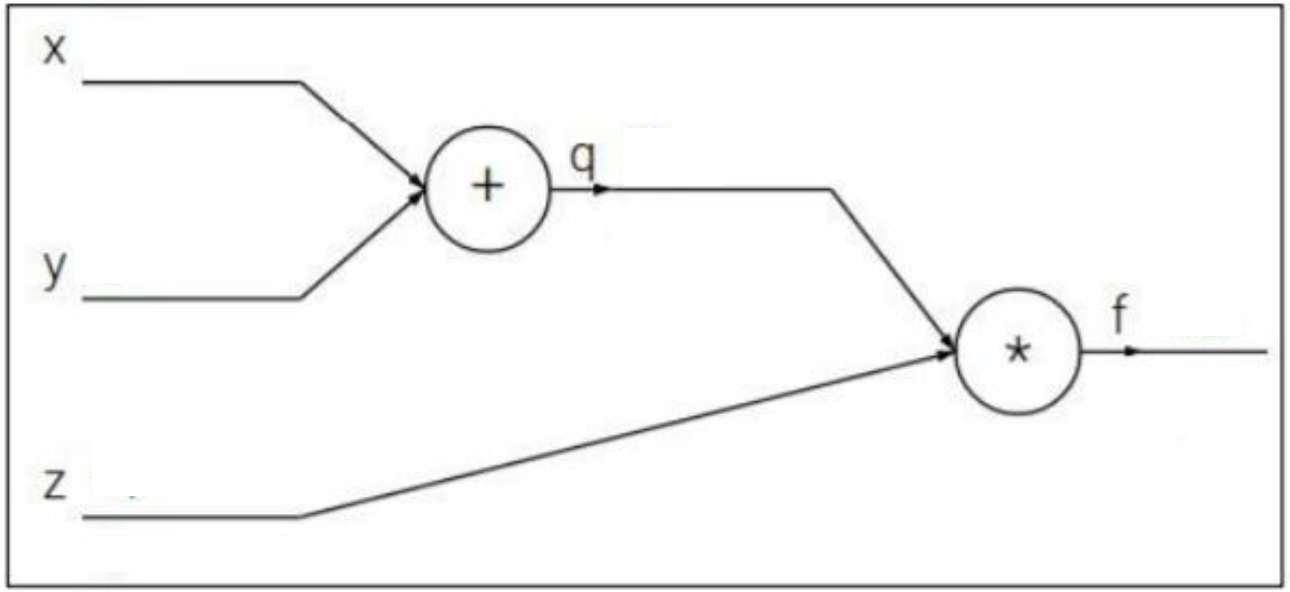


Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

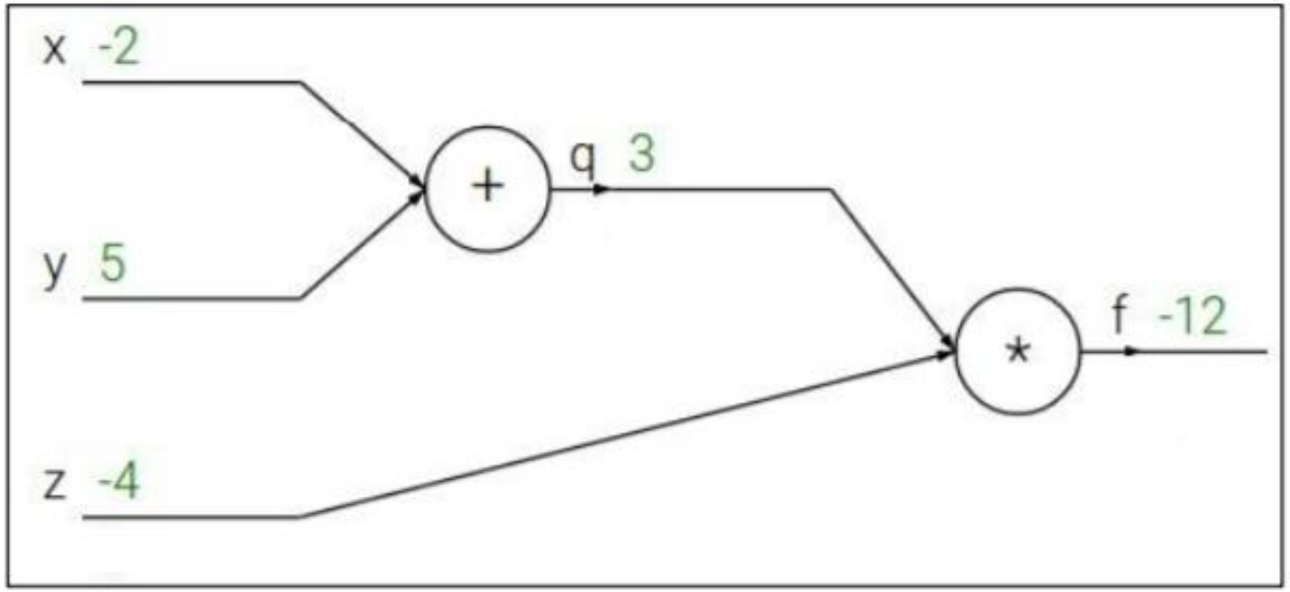


Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



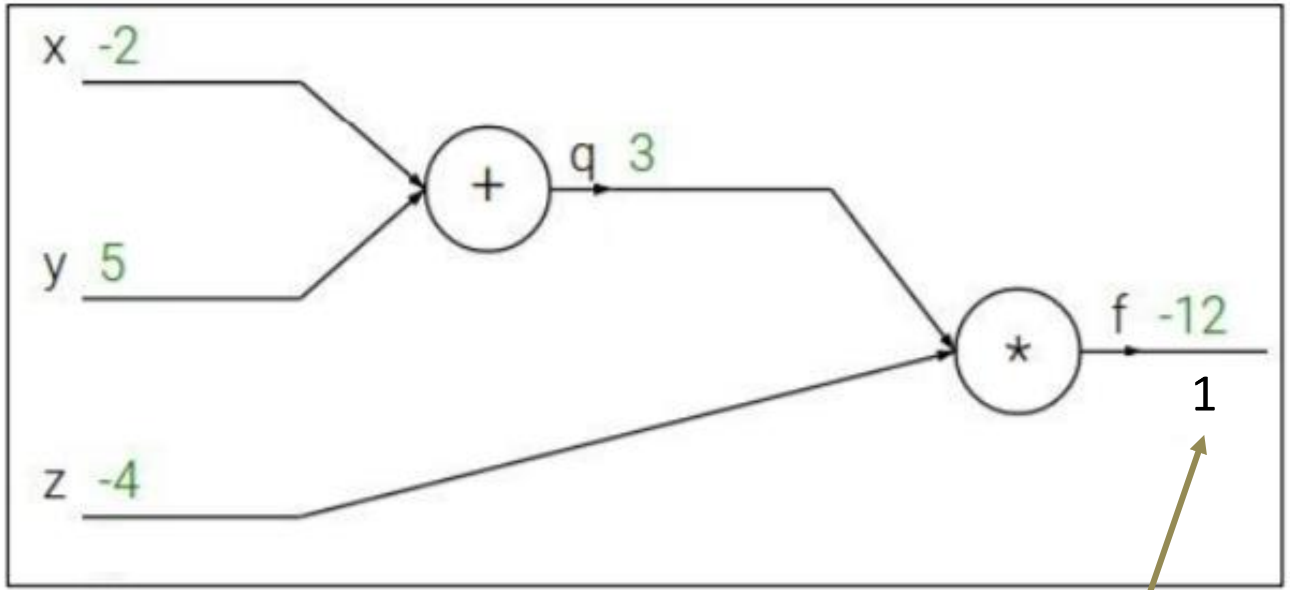
This is called- **the forward pass**

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



Step 0: initialization

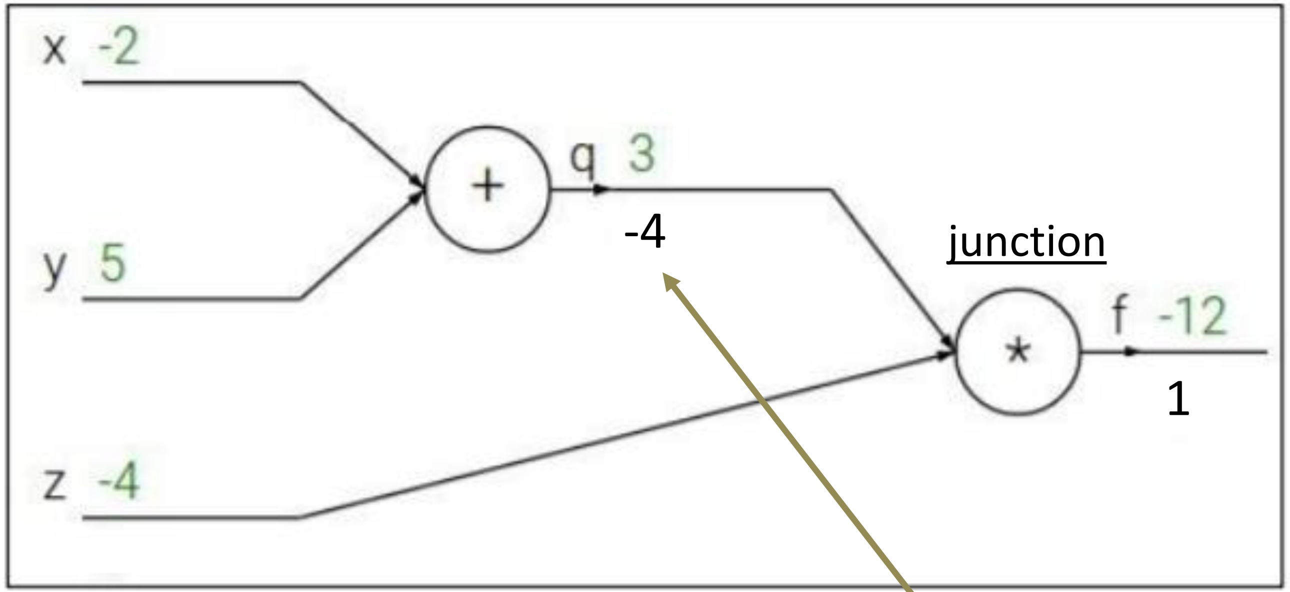
$$\frac{\partial f}{\partial f} = 1$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



Iterative step: chain rule on locale junction.

$$\frac{\partial f}{\partial q} = \frac{\partial f}{\partial f} \cdot \frac{\partial f}{\partial q} = 1 \cdot -4 = -4$$

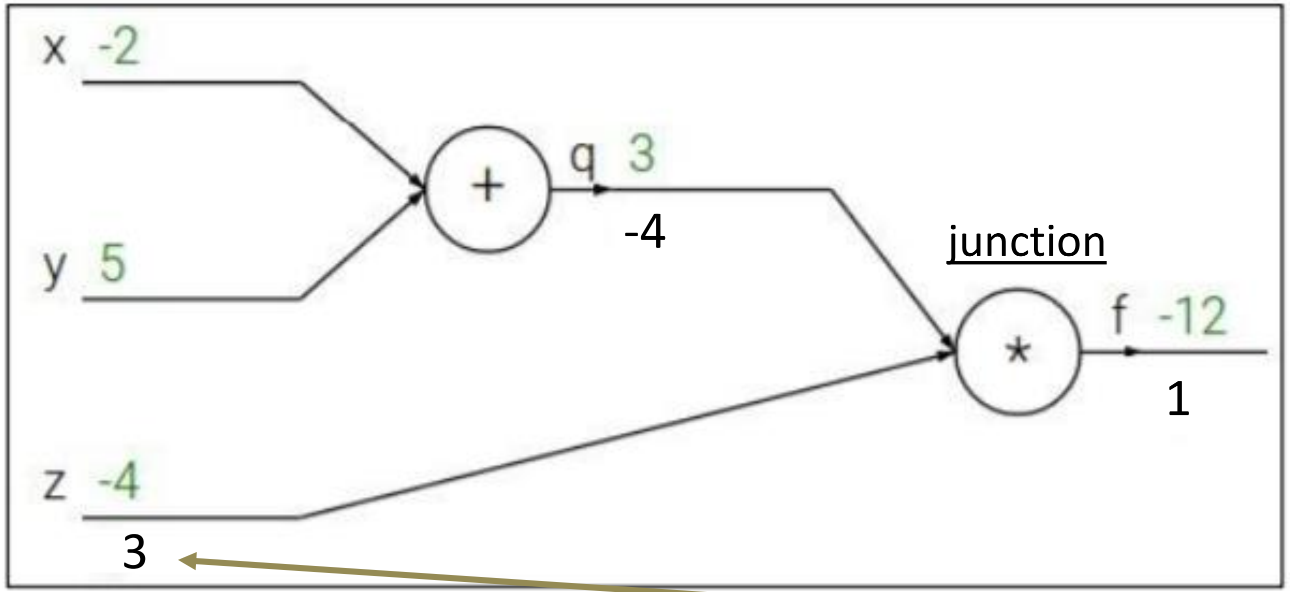
$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4



Iterative step: chain rule on locale junction.

$$\frac{\partial f}{\partial z} = \frac{\partial f}{\partial f} \cdot \frac{\partial f}{\partial z} = 1 \cdot 3 = 3$$

$$f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

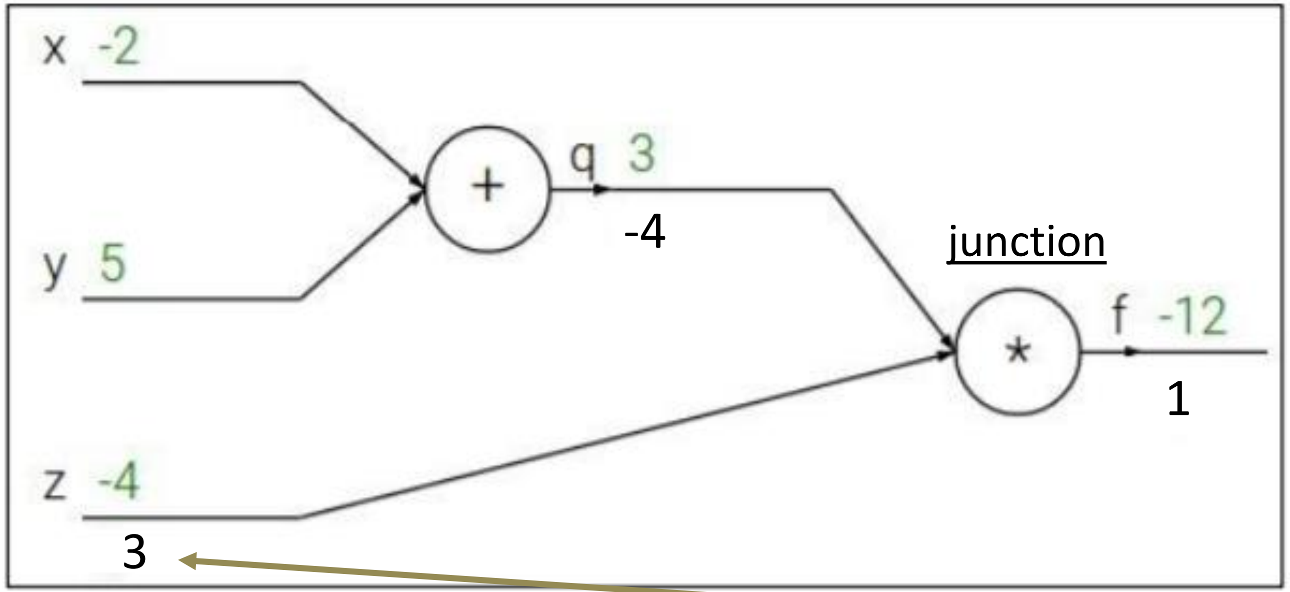
Backpropagation: a simple example

$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$\boxed{f = qz \qquad \frac{\partial f}{\partial q} = z, \frac{\partial f}{\partial z} = q}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$ ✔



Iterative step: chain rule on locale junction.

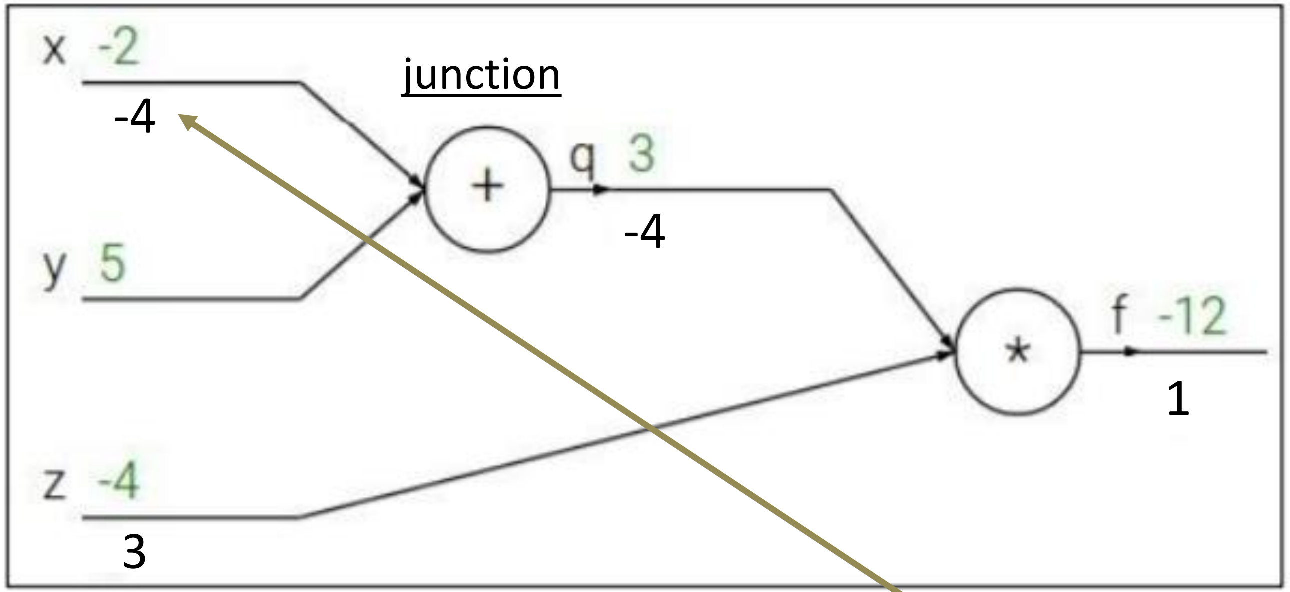
$$\boxed{\frac{\partial f}{\partial z} = \frac{\partial f}{\partial f} \cdot \frac{\partial f}{\partial z} = 1 \cdot 3 = 3}$$

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$



Iterative step: chain rule on locale junction.

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \cdot \frac{\partial q}{\partial x} = -4 \cdot 1 = -4$$

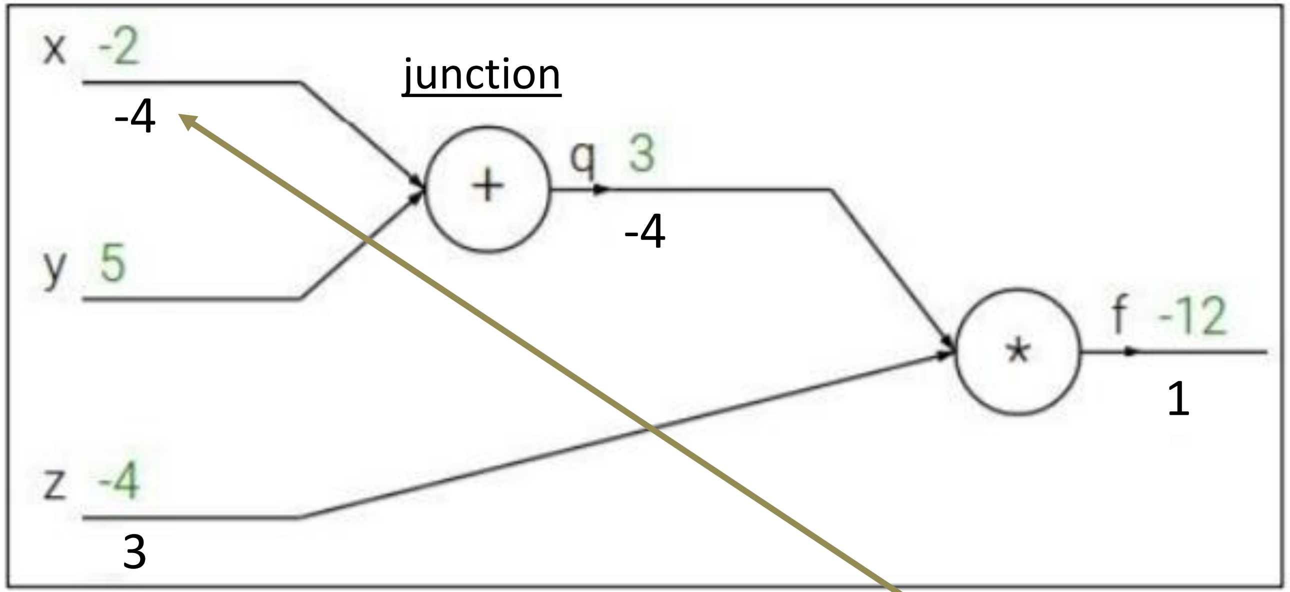
Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1$$



Iterative step: chain rule on locale junction.

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q} \cdot \frac{\partial q}{\partial x} = -4 \cdot 1 = -4$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

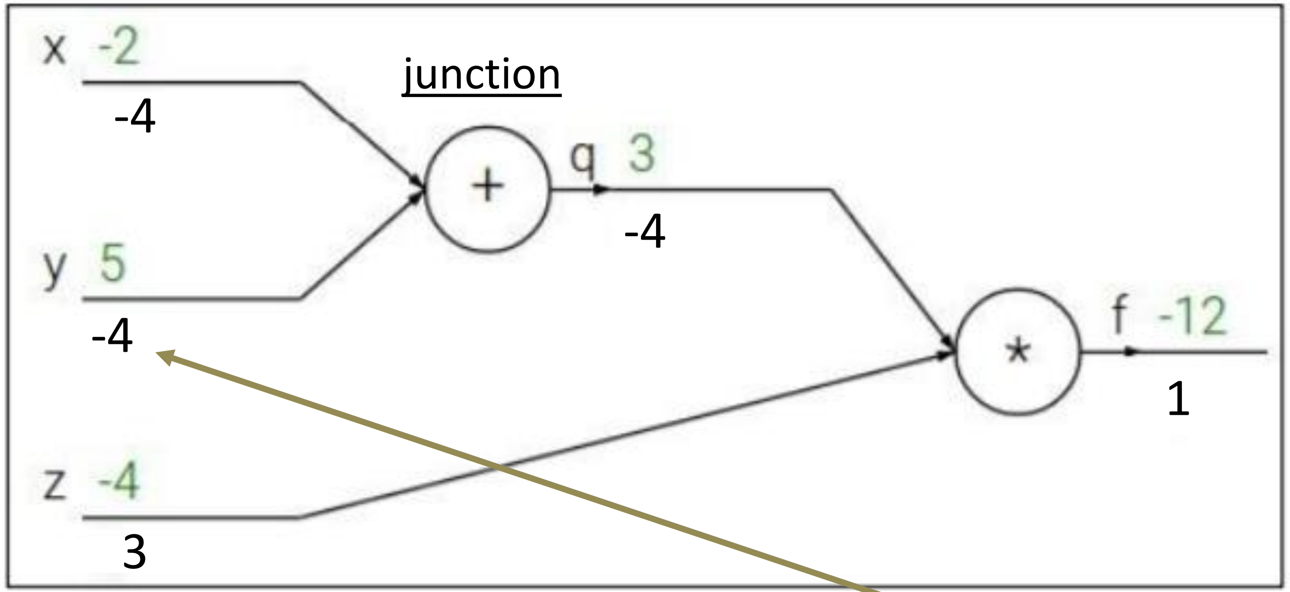
Backpropagation: a simple example

$$f(x,y,z) = (x+y)z$$

e.g. x = -2, y = 5, z = -4

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$



Iterative step: chain rule on locale junction.

$$\boxed{\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \cdot \frac{\partial q}{\partial y} = -4 \cdot 1 = -4}$$

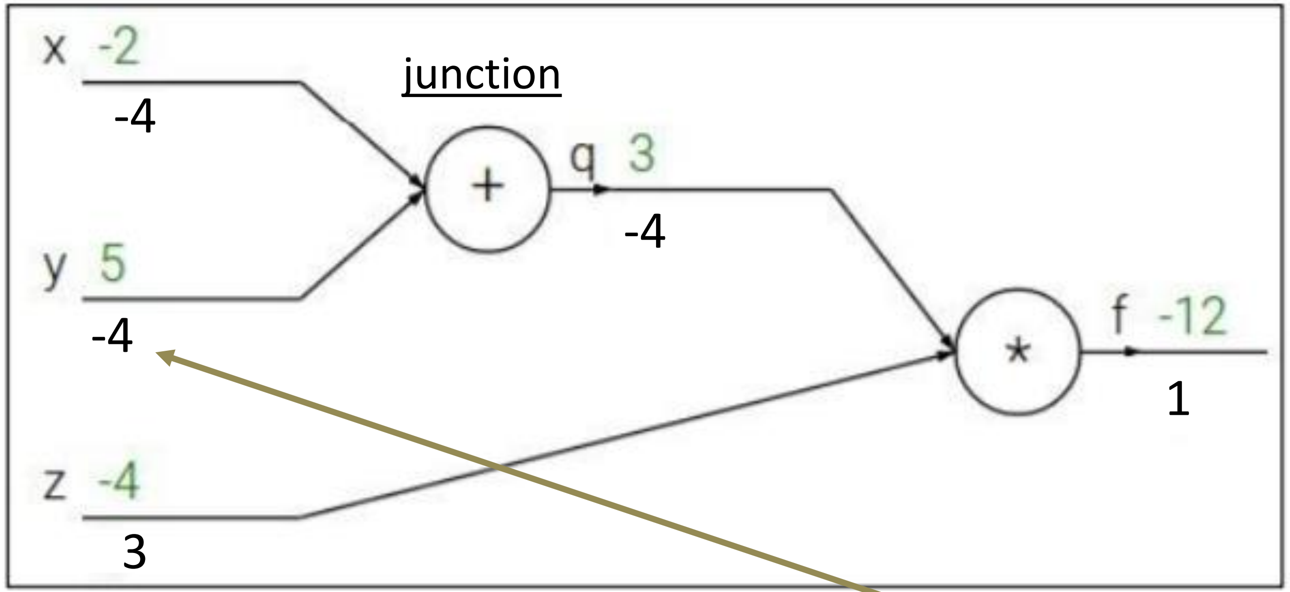
Backpropagation: a simple example

$$f(x, y, z) = (x + y)z$$

e.g. x = -2, y = 5, z = -4

$$\boxed{q = x + y \qquad \frac{\partial q}{\partial x} = 1, \frac{\partial q}{\partial y} = 1}$$

Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

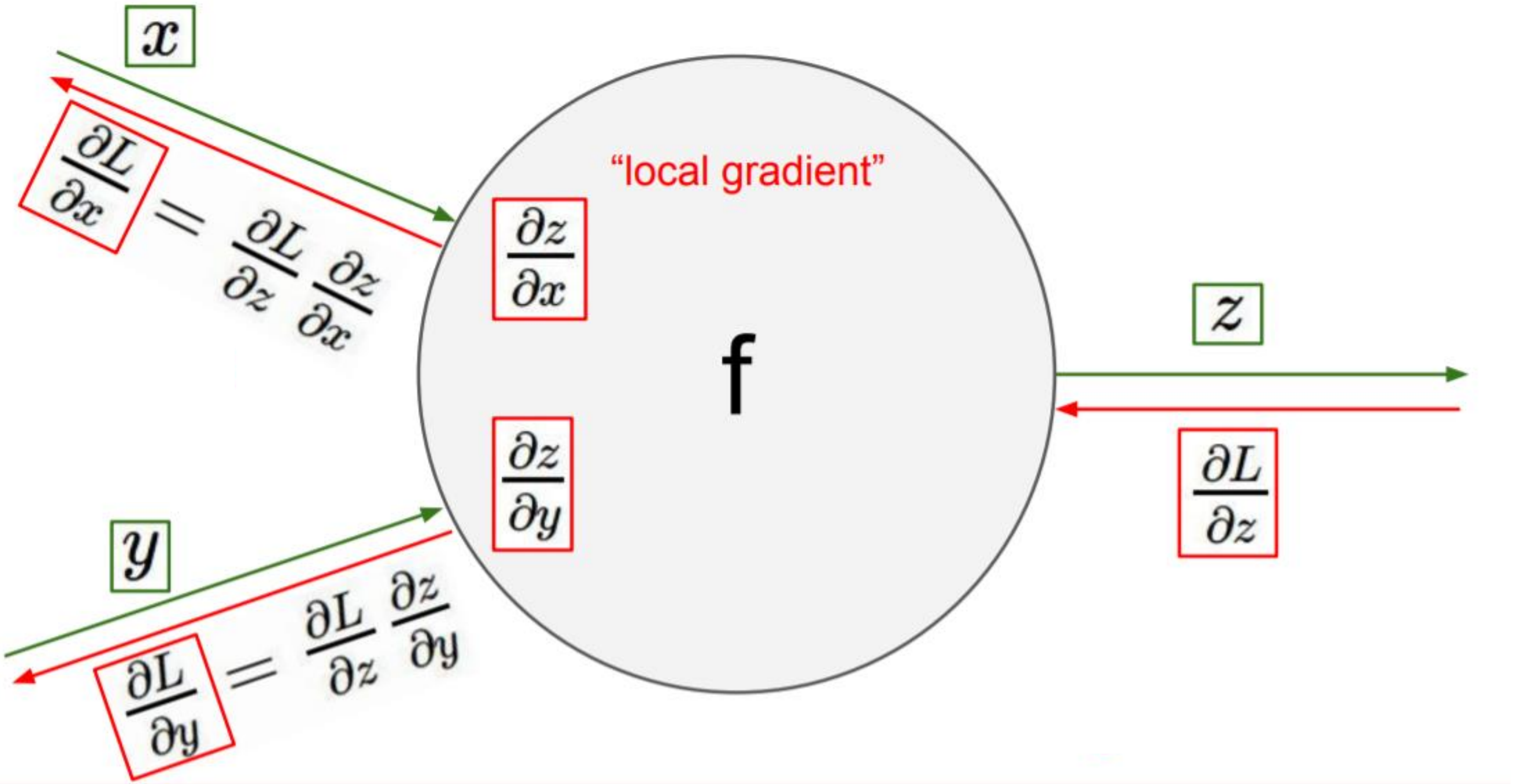


Iterative step: chain rule on locale junction.

$$\boxed{\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \cdot \frac{\partial q}{\partial y} = -4 \cdot 1 = -4}$$

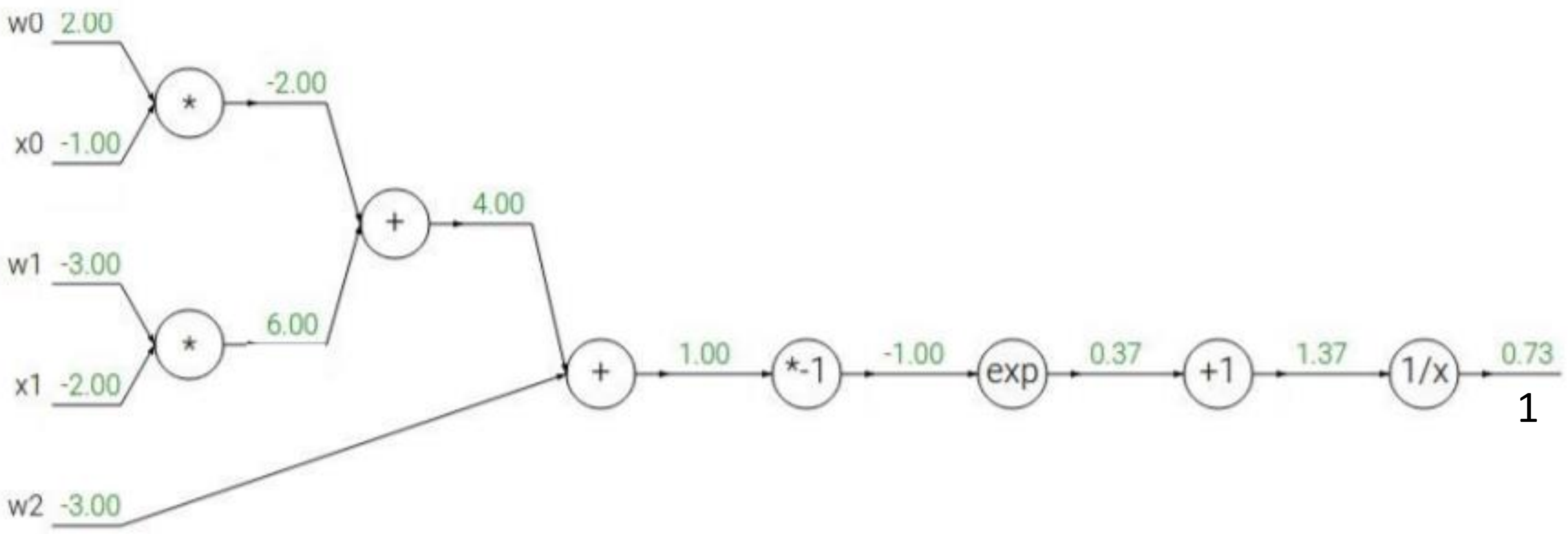
Backpropagation- conclusions

- All that we need to know in each junction is:
  - The local gradient from the junction's equation.
  - the chain rule result that was "back propagated" to the junction.



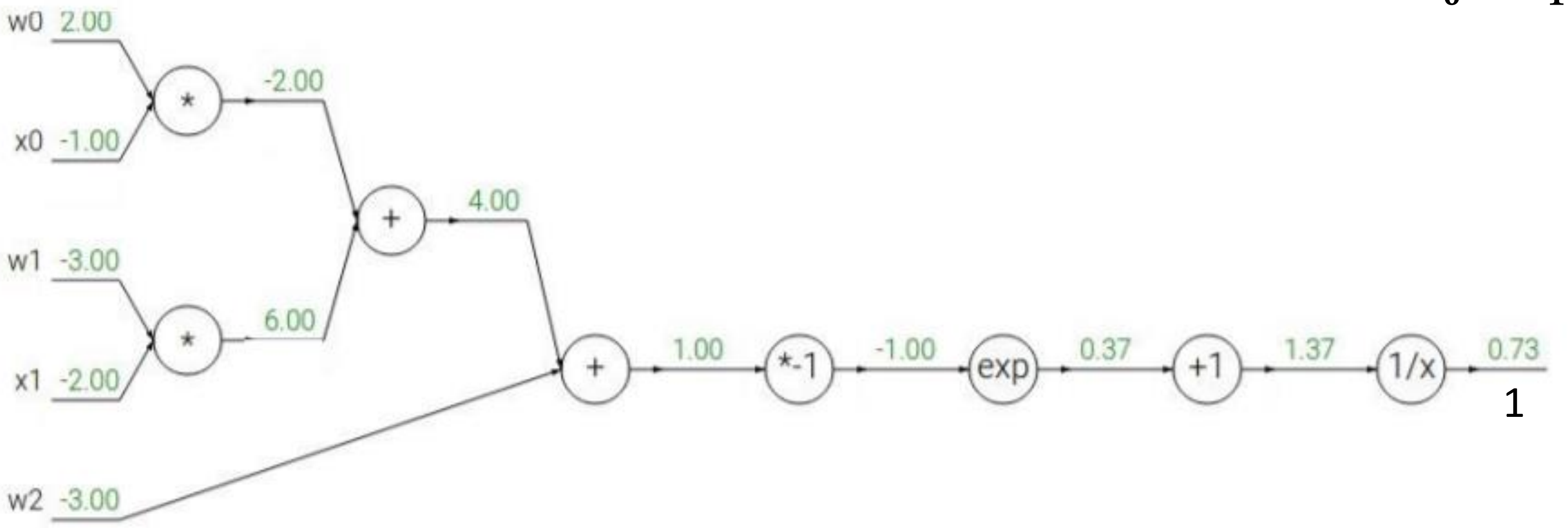
Another example:

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

We want to find:
$\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$

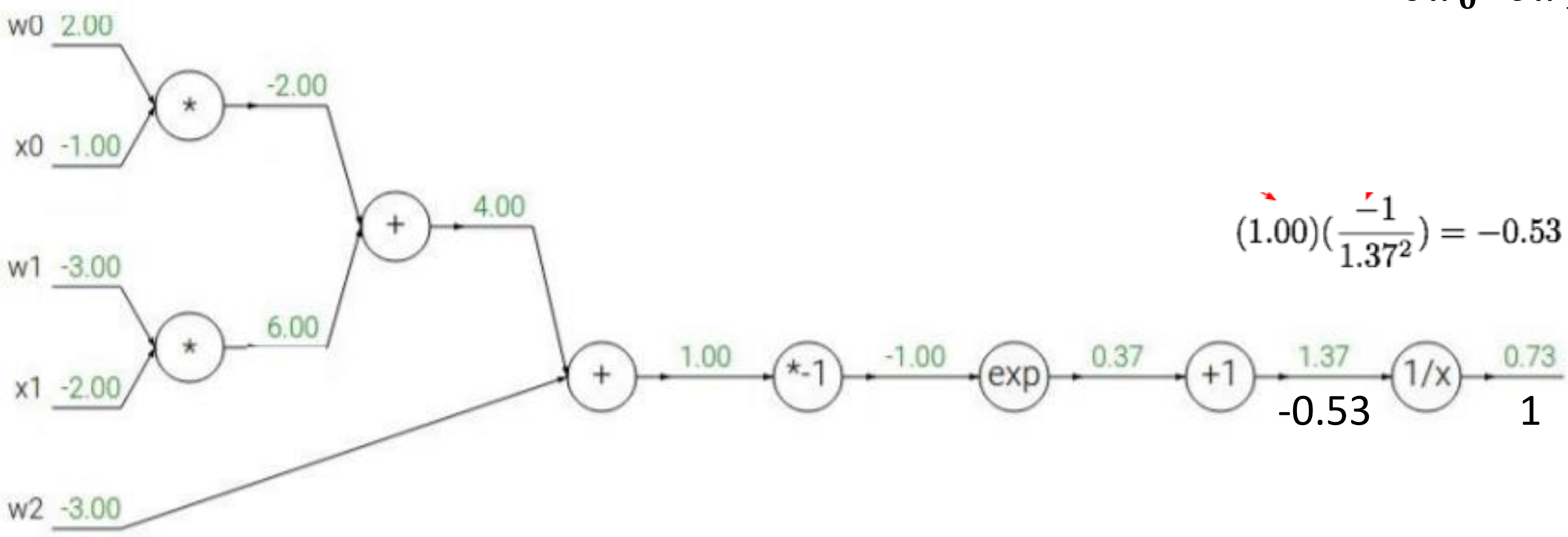


Another example:

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

We want to find:

$$\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$$



$$(1.00)\left(\frac{-1}{1.37^2}\right) = -0.53$$

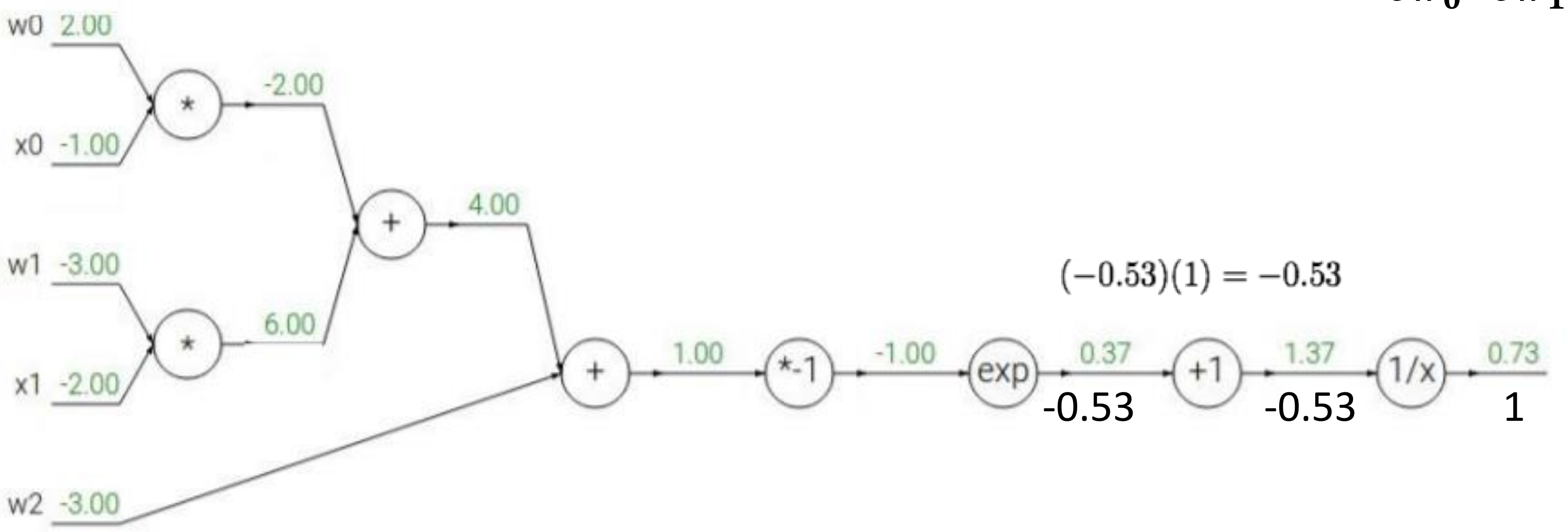
$$f(x) = \frac{1}{x} \quad \rightarrow \quad \frac{df}{dx} = -1/x^2$$

Another example:

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

We want to find:

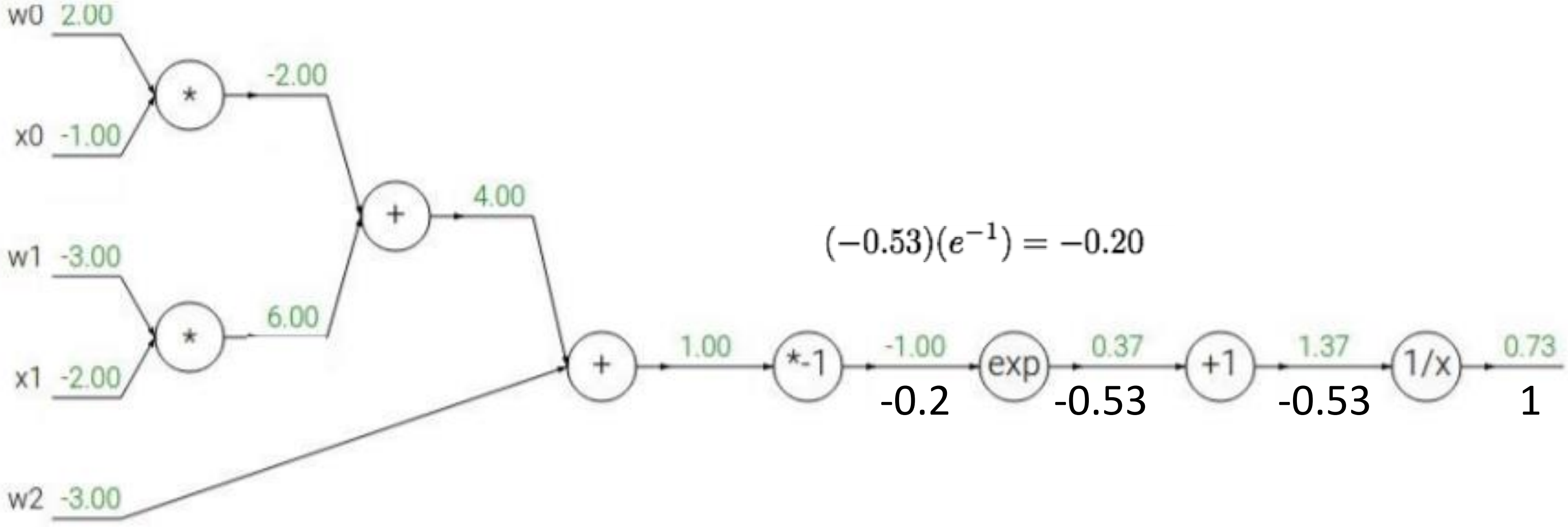
$$\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$$



$$f_c(x) = c + x \qquad \rightarrow \qquad \frac{df}{dx} = 1$$

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

We want to find:
$\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$



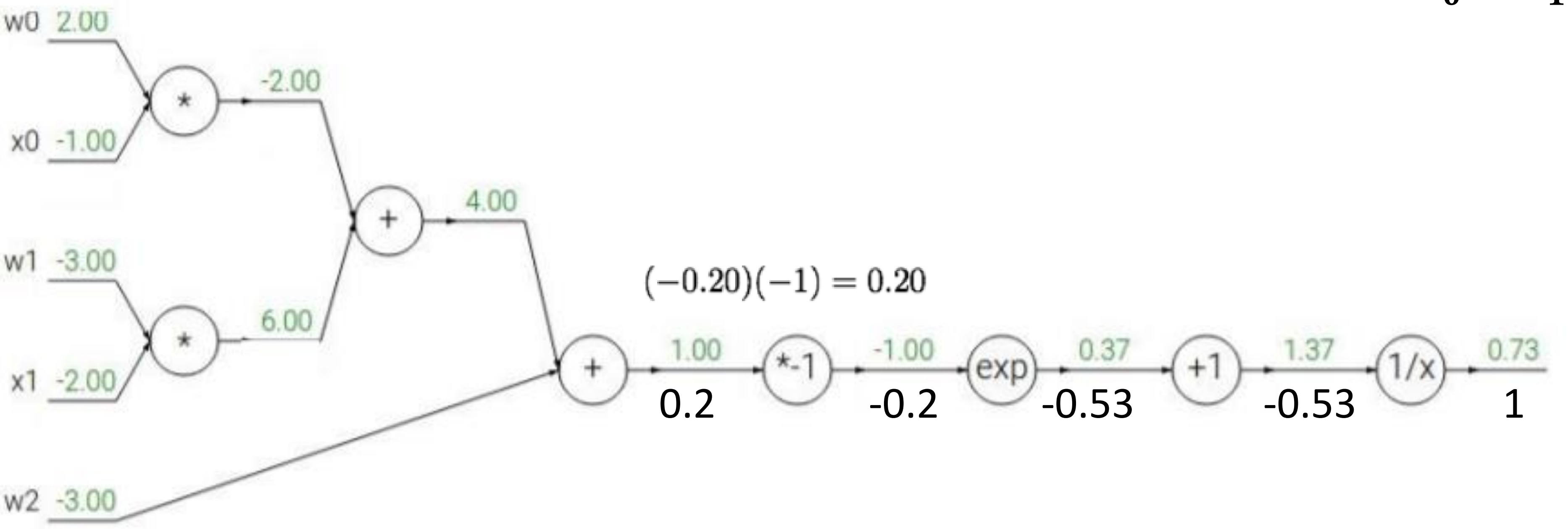
$f(x) = e^x \quad \rightarrow \quad \frac{df}{dx} = e^x$

Another example:

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

We want to find:

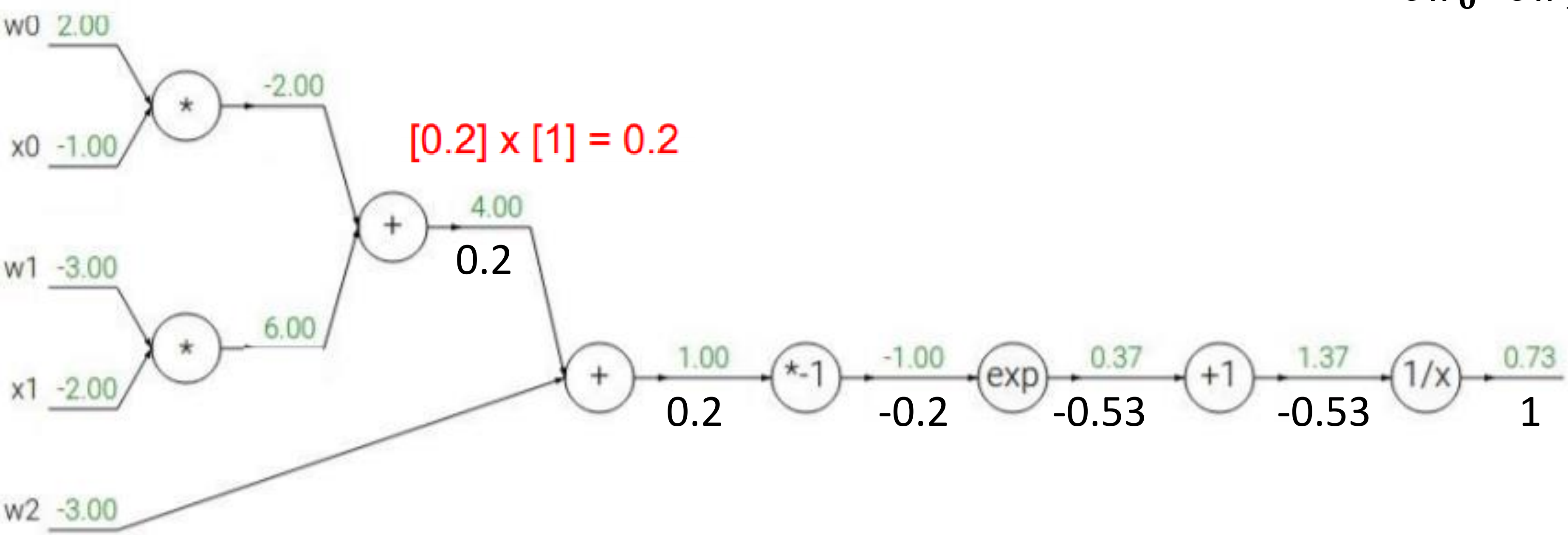
$$\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$$



$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

We want to find: $\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$



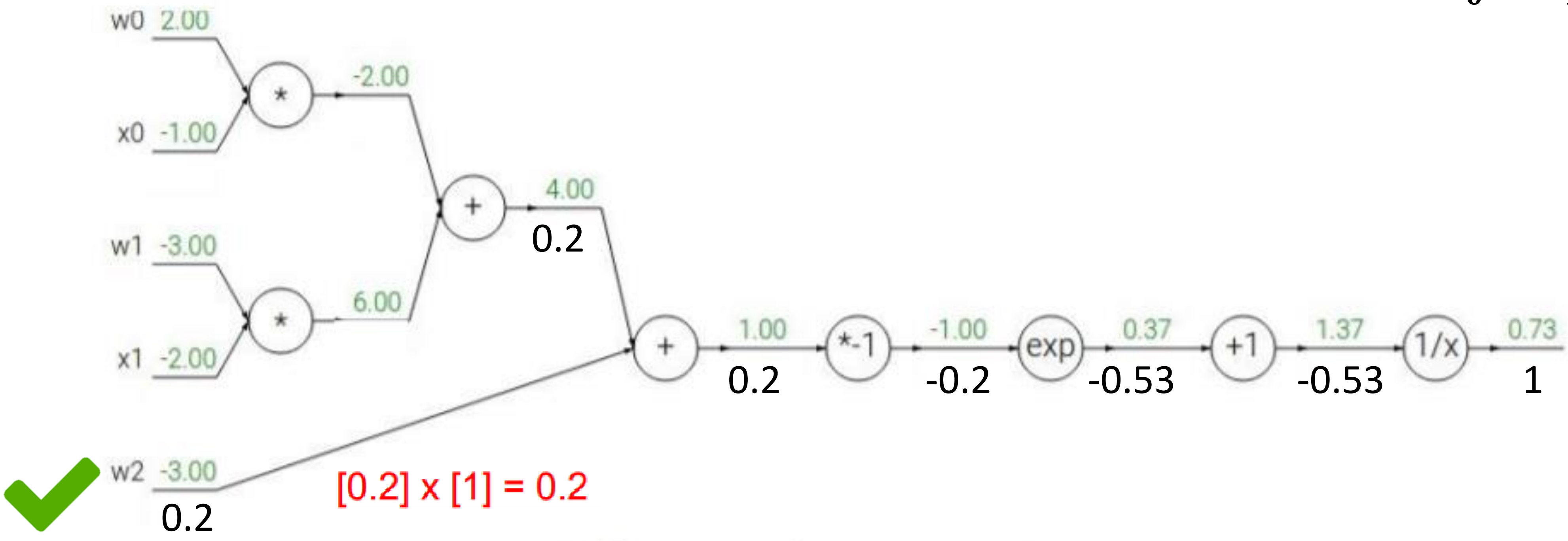
$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$

Another example:

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

We want to find:

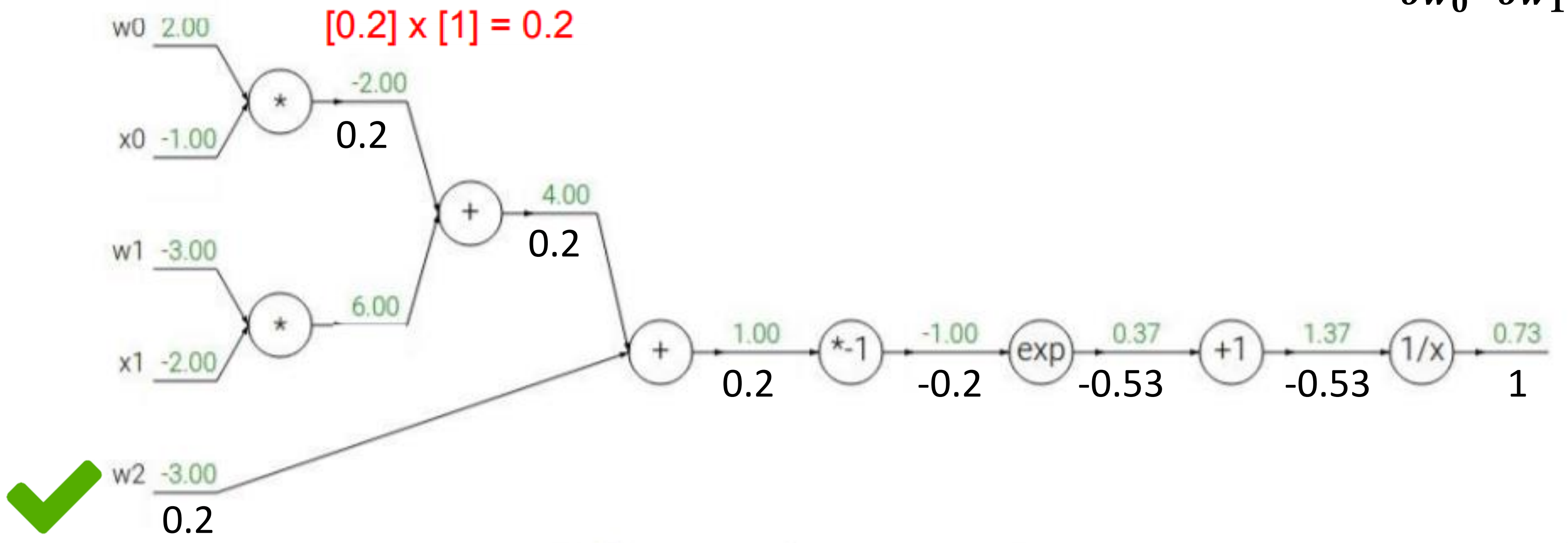
$$\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$$



$$f_c(x) = c + x \qquad \rightarrow \qquad \frac{df}{dx} = 1$$

Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

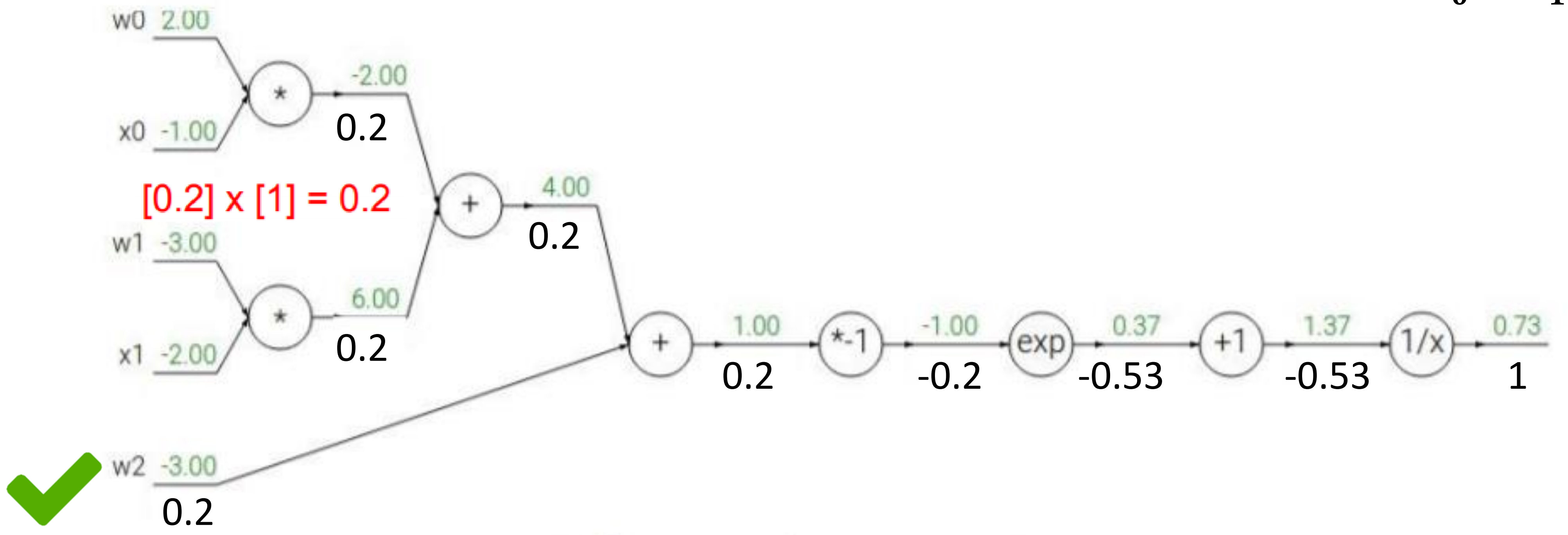
We want to find: $\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$



$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Another example: $f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$

We want to find: $\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$



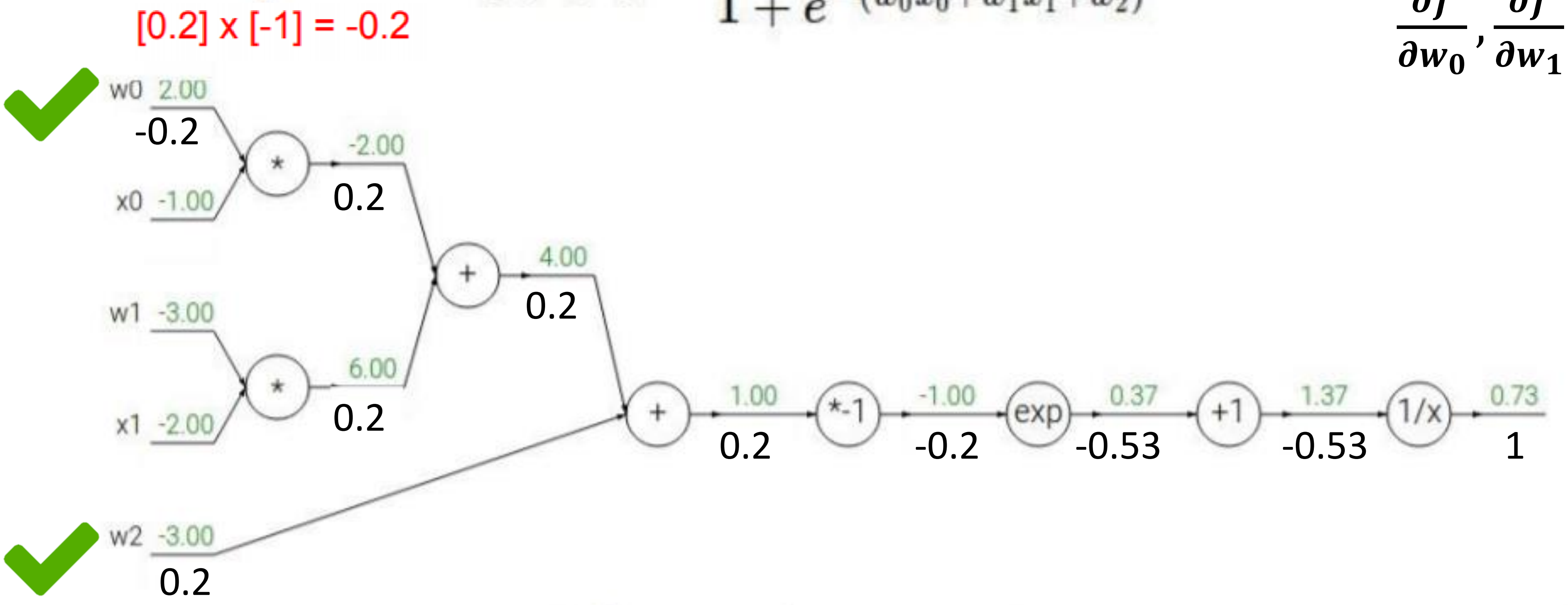
$$f_c(x) = c + x \quad \rightarrow \quad \frac{df}{dx} = 1$$

Another example:

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

We want to find:

$$\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$$



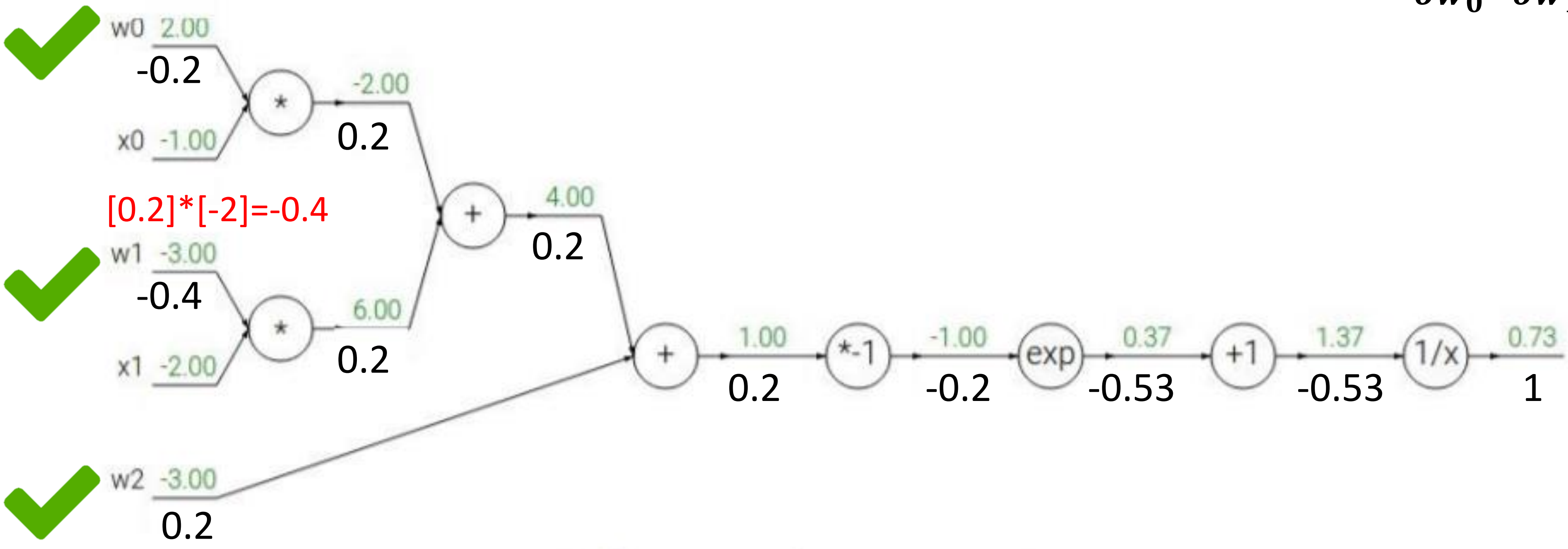
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

Another example:

$$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

We want to find:

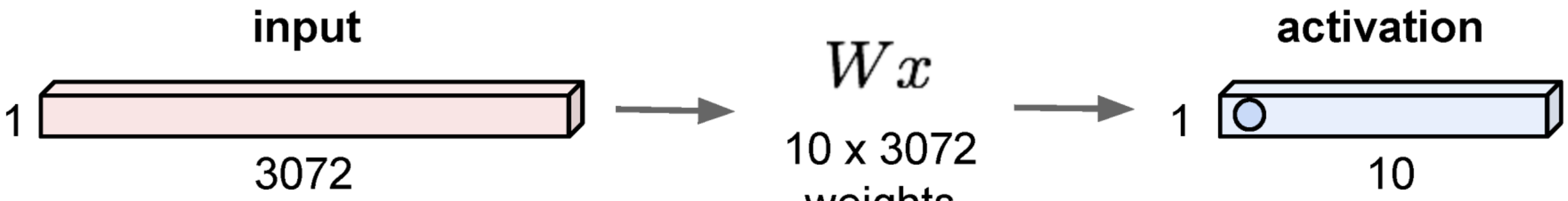
$$\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$$



$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1

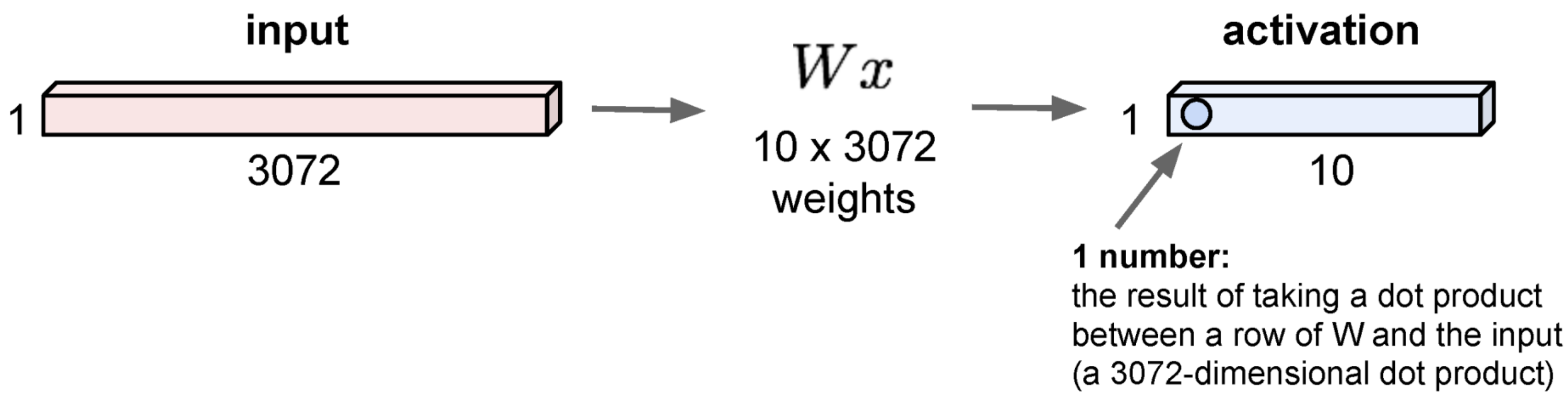


Wx

10 x 3072
weights

Fully Connected Layer

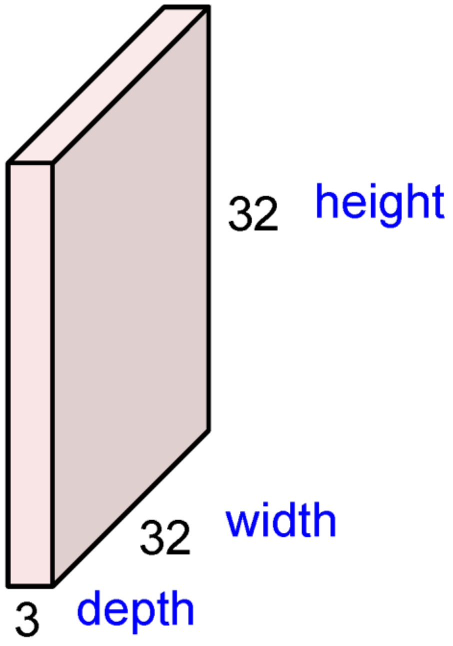
32x32x3 image -> stretch to 3072 x 1



Same as a linear classifier!

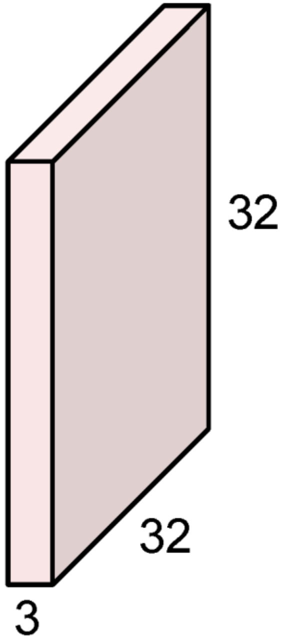
Convolution Layer

32x32x3 image -> preserve spatial structure

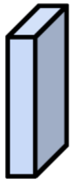


Convolution Layer

32x32x3 image



5x5x3 filter

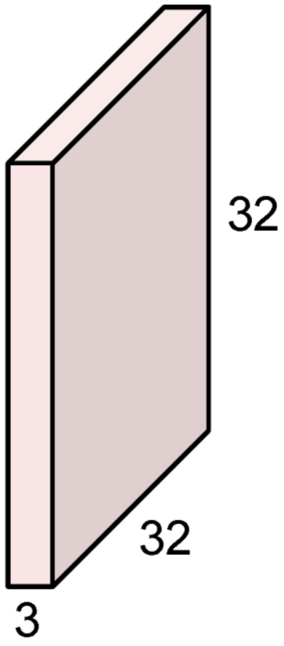


Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

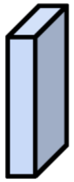
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



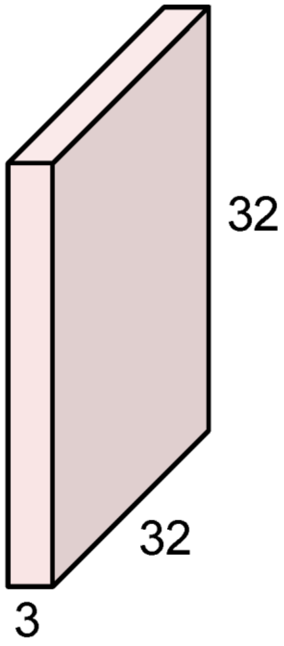
5x5x3 filter



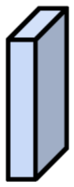
Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolution Layer

32x32x3 image



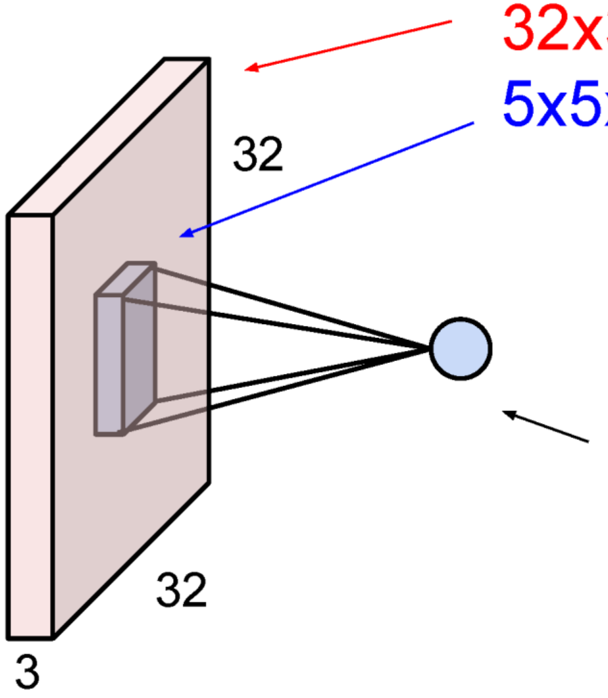
5x5x3 filter



Convolve the filter with the image
i.e. “slide over the image spatially, computing dot products”

Number of weights: 5 x 5 x 3 + 1 = **76**
(vs. 3072 for a fully-connected layer)
(+1 for bias)

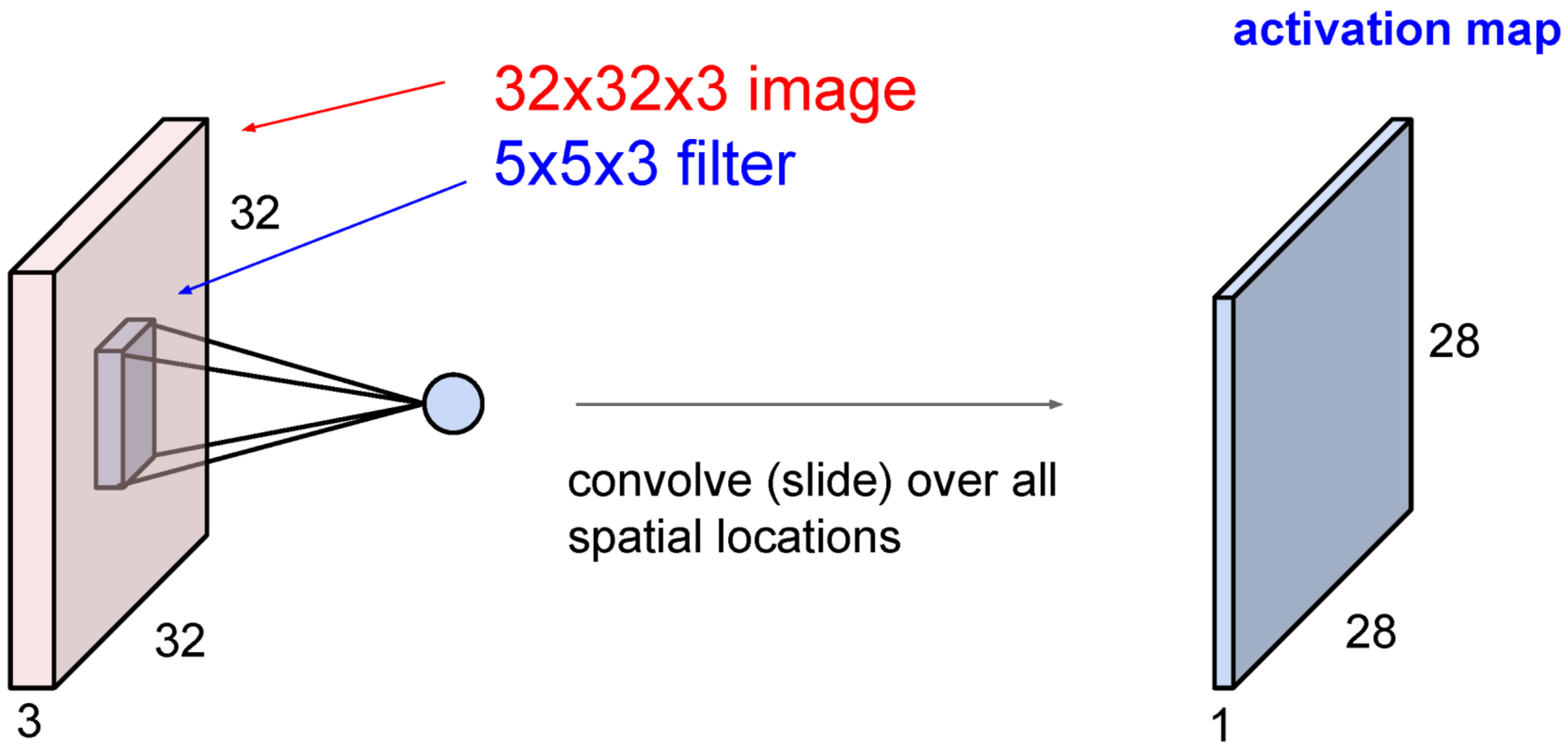
Convolution Layer



1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

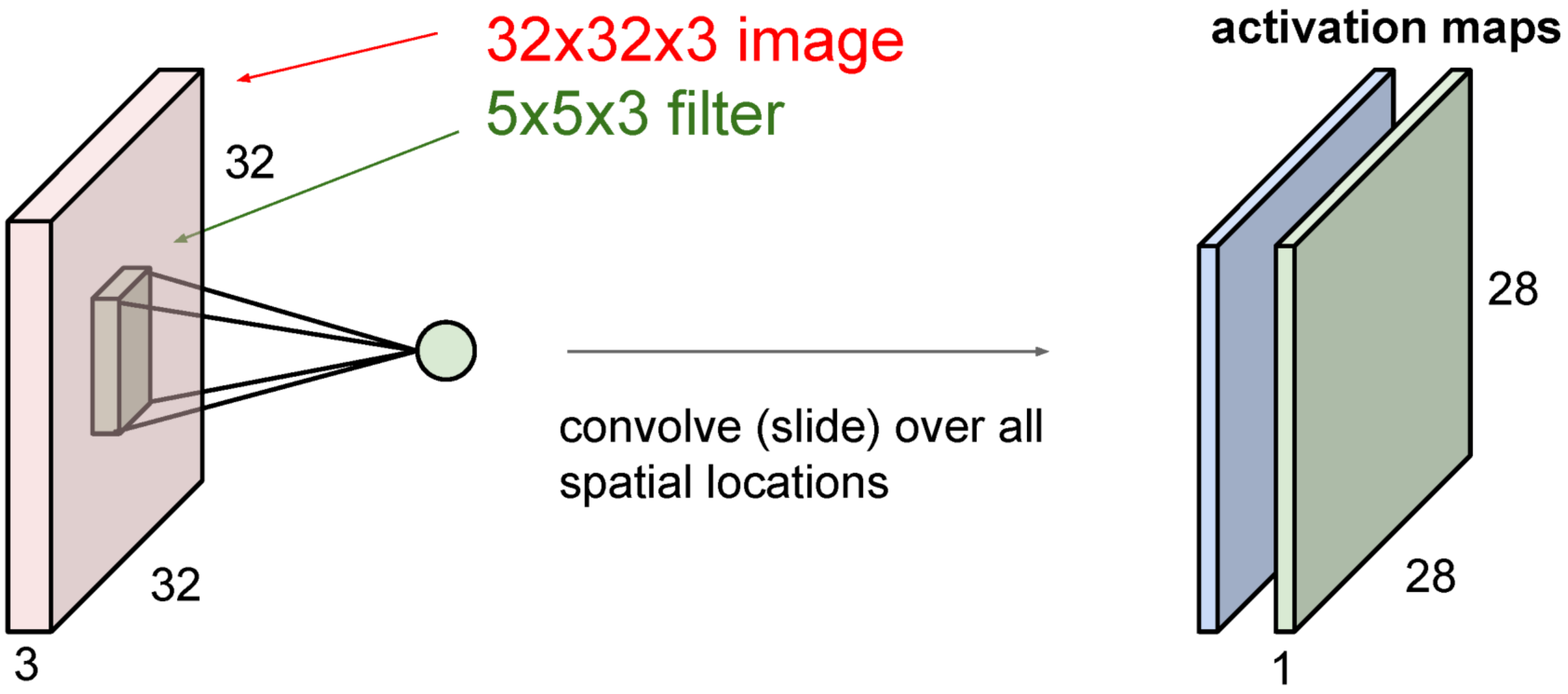
$$w^T x + b$$

Convolution Layer

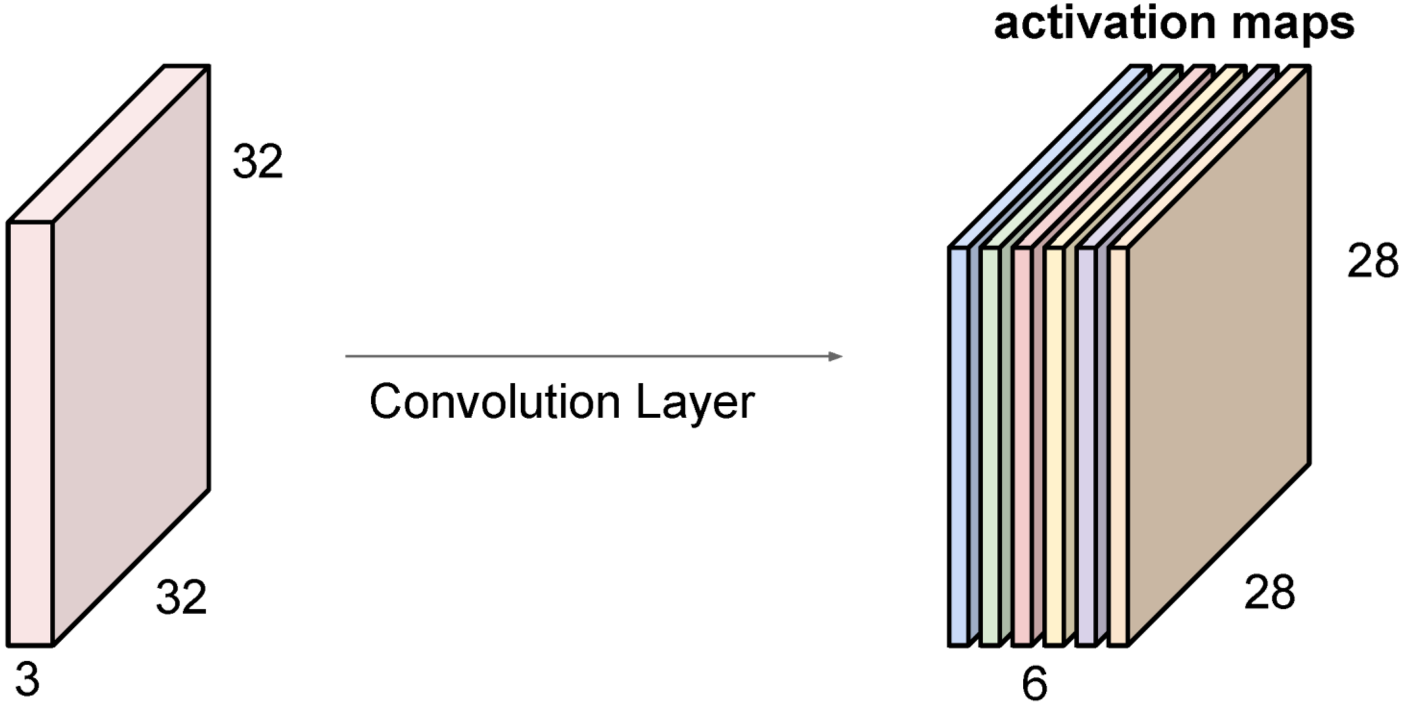


Convolution Layer

consider a second, green filter



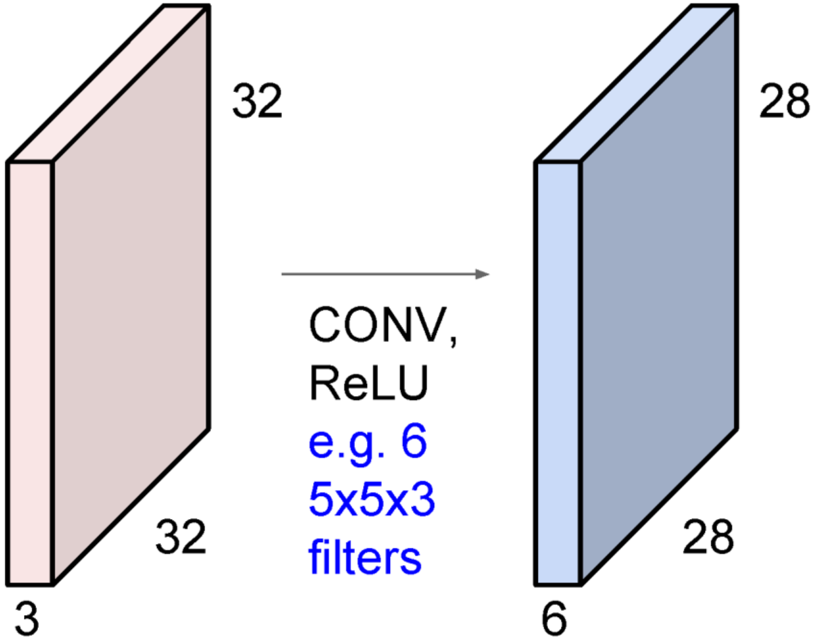
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



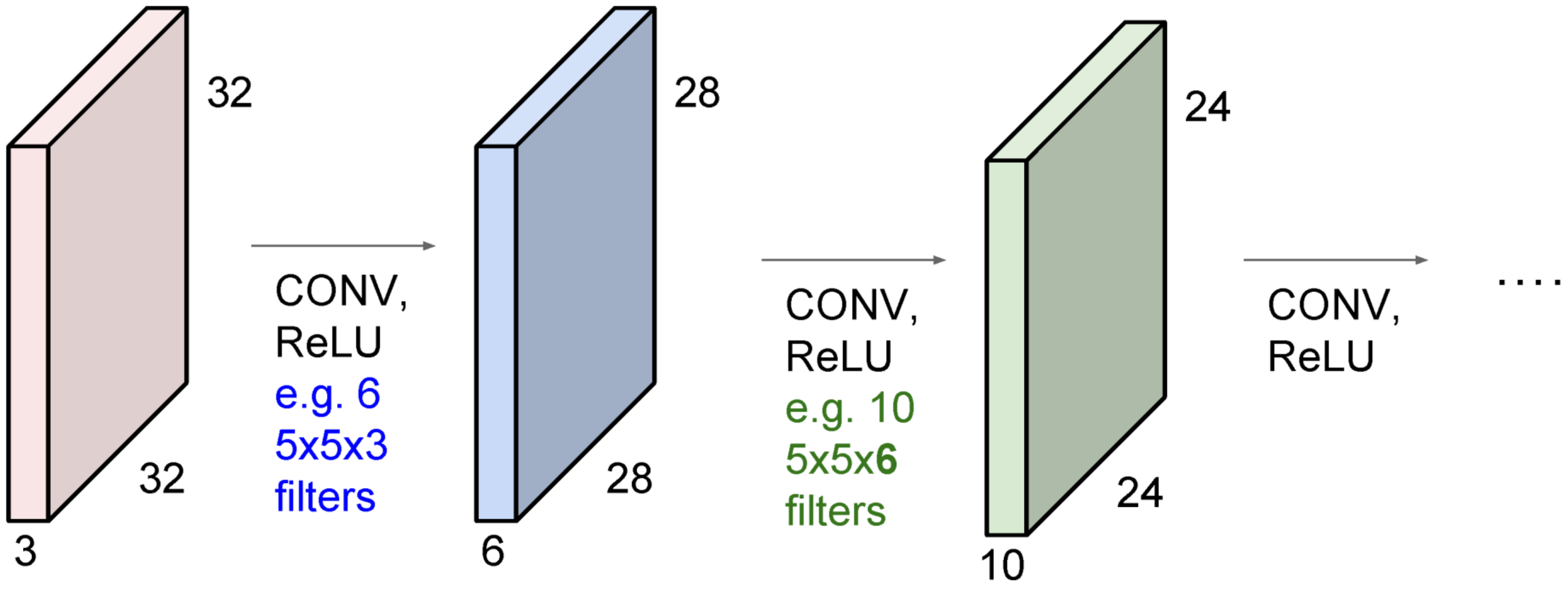
We stack these up to get a "new image" of size 28x28x6!

(total number of parameters: 6 x (75 + 1) = **456**)

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



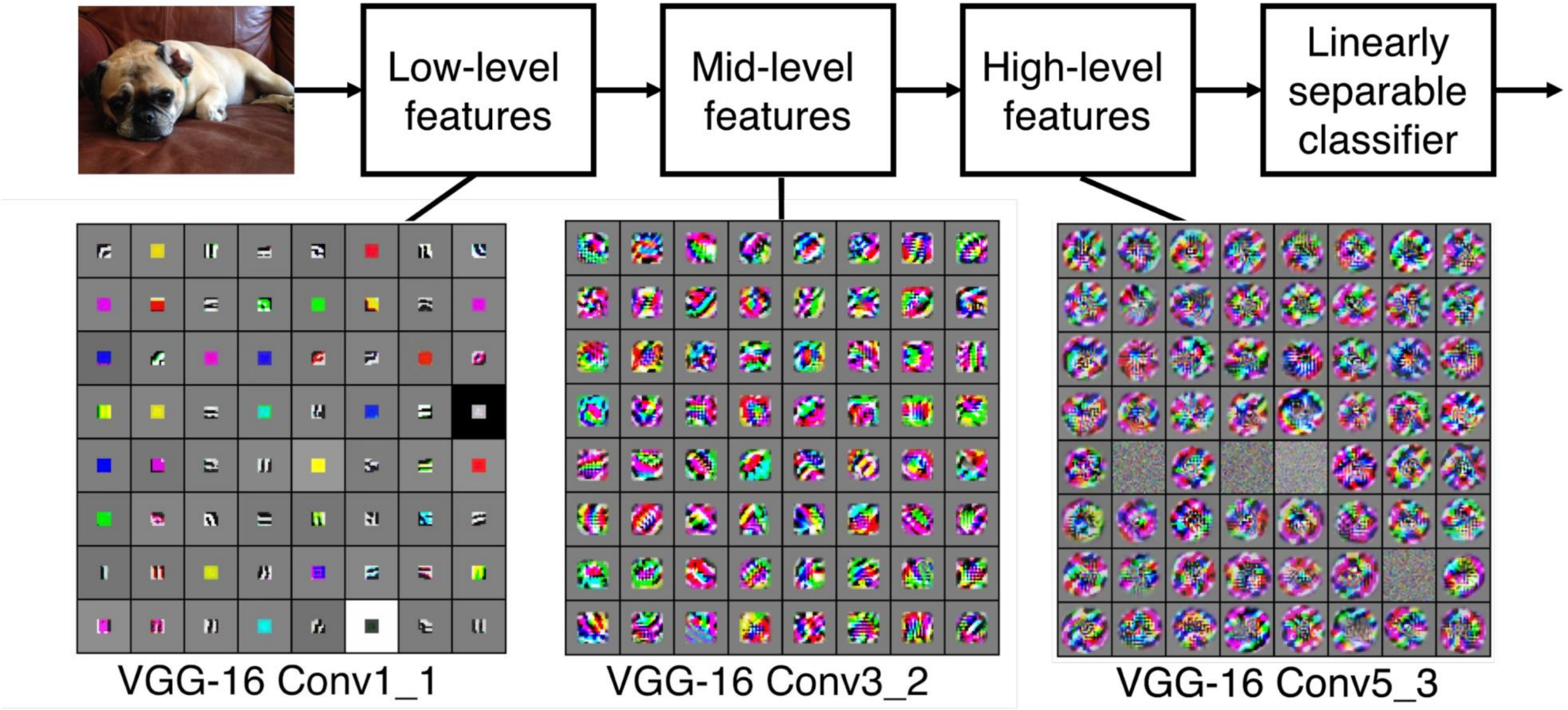
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



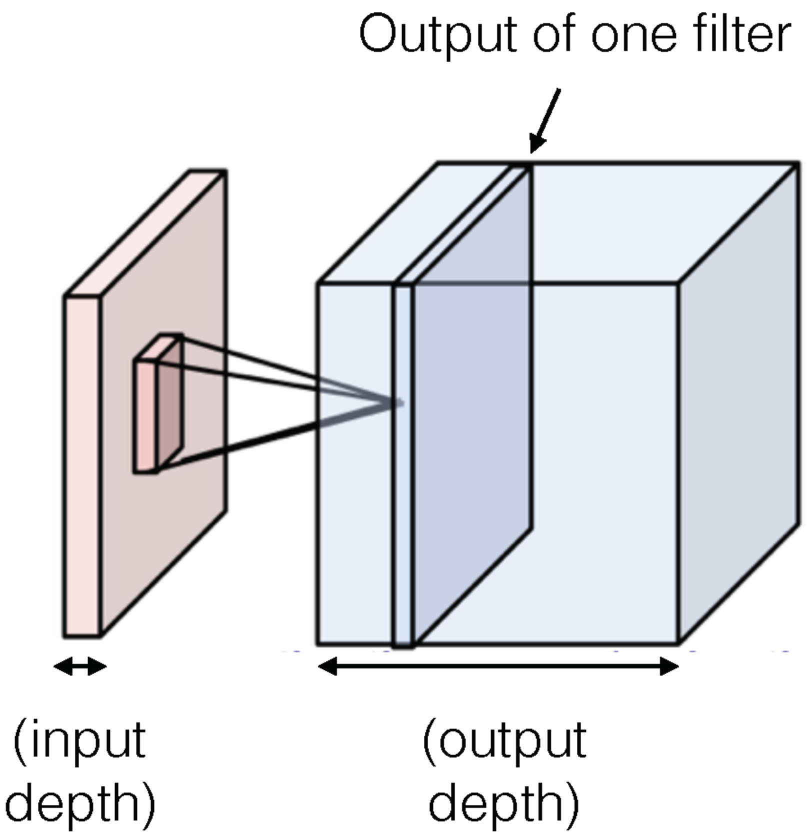
Preview

[Zeiler and Fergus 2013]

Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].



3D Activations



One set of weights gives one slice in the output

To get a 3D output of depth *D*, use *D* different filters

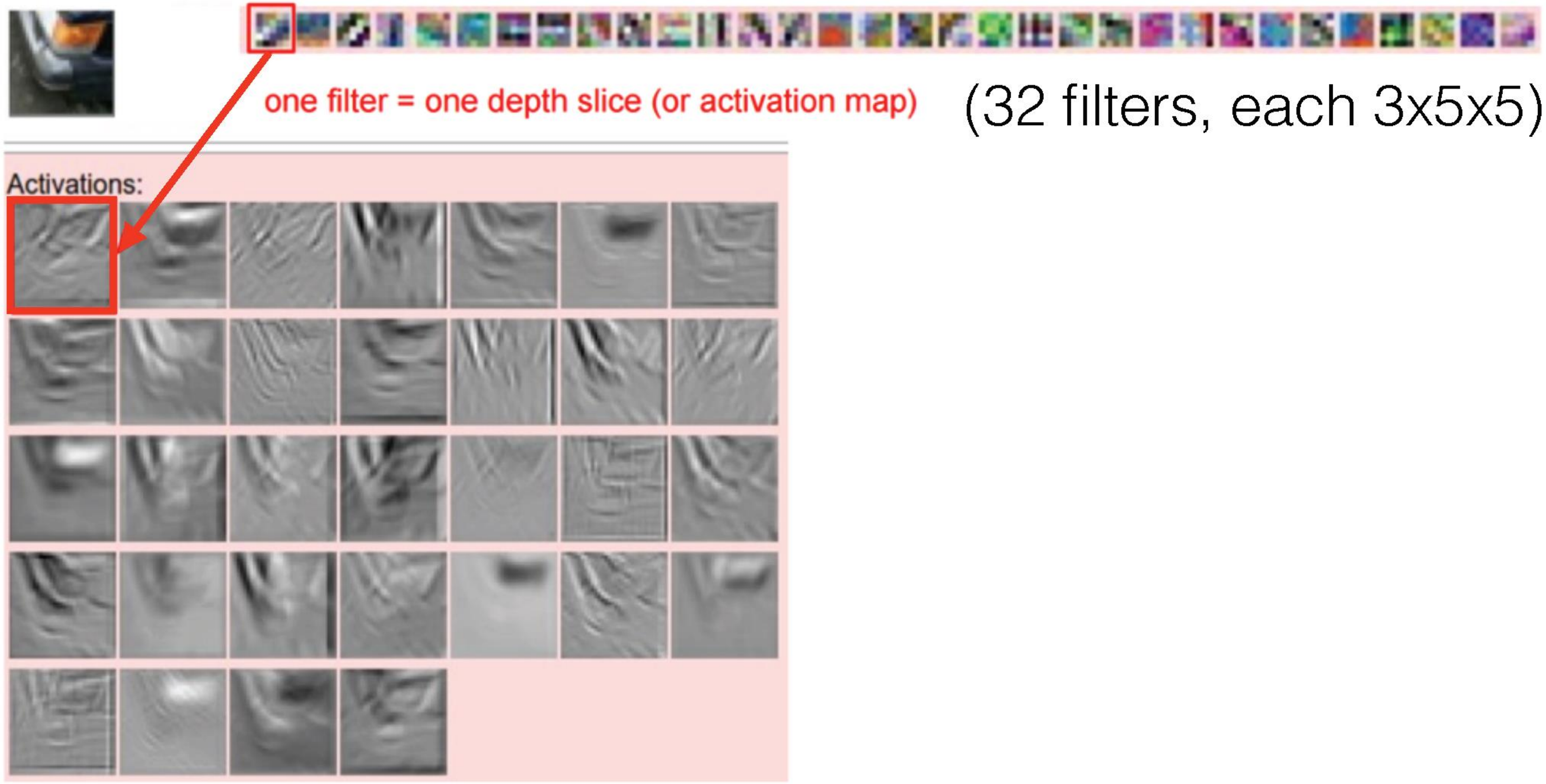
In practice, ConvNets use many filters (~64 to 1024)

All together, the weights are **4** dimensional:
(output depth, input depth, kernel height, kernel width)

3D Activations

We can unravel the 3D cube and show each layer separately:

(Input)



(32 filters, each 3x5x5)

Figure: Andrej Karpathy

3D Activations

We can unravel the 3D cube and show each layer separately:

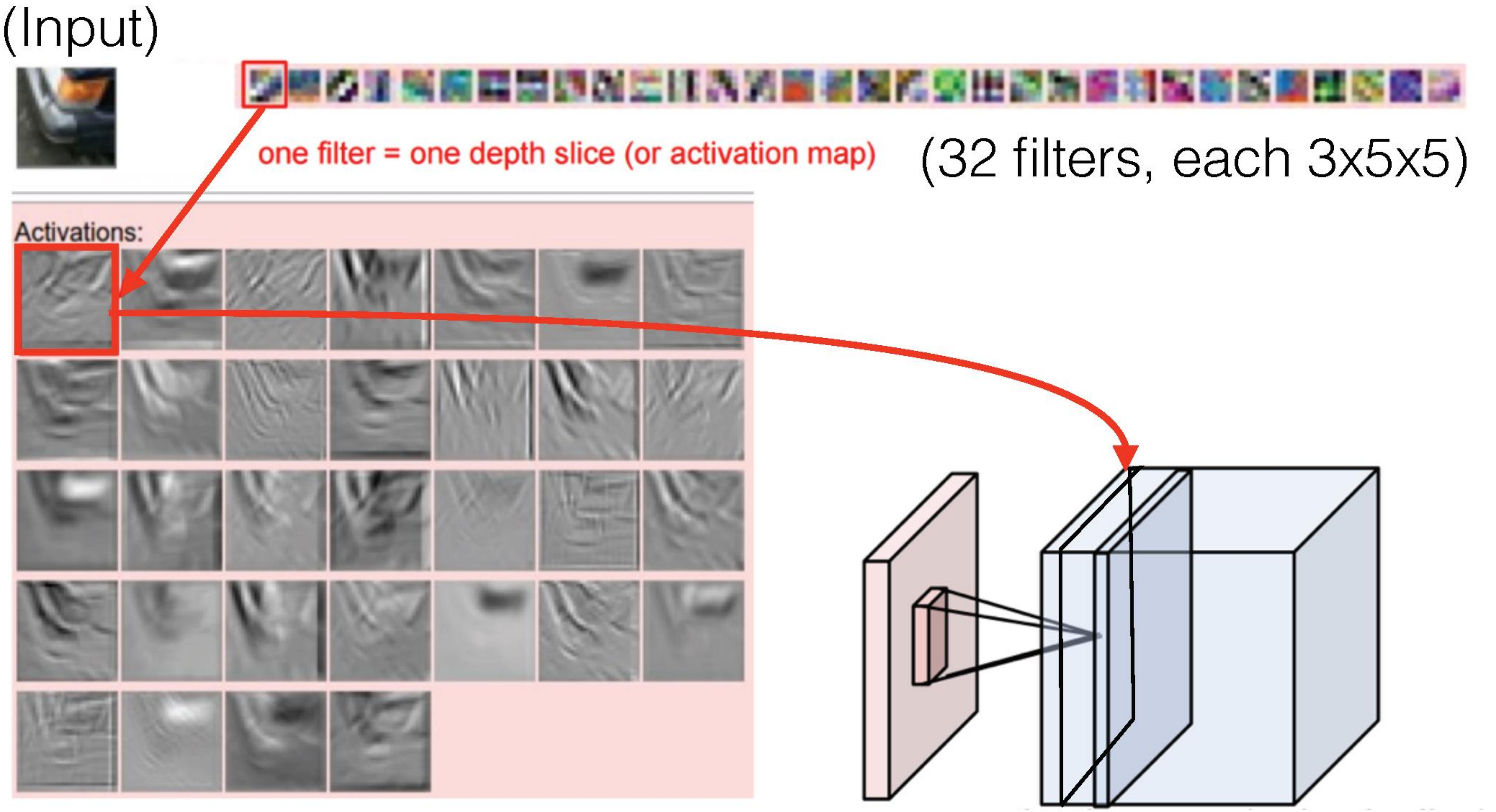


Figure: Andrej Karpathy

3D Activations

We can unravel the 3D cube and show each layer separately:

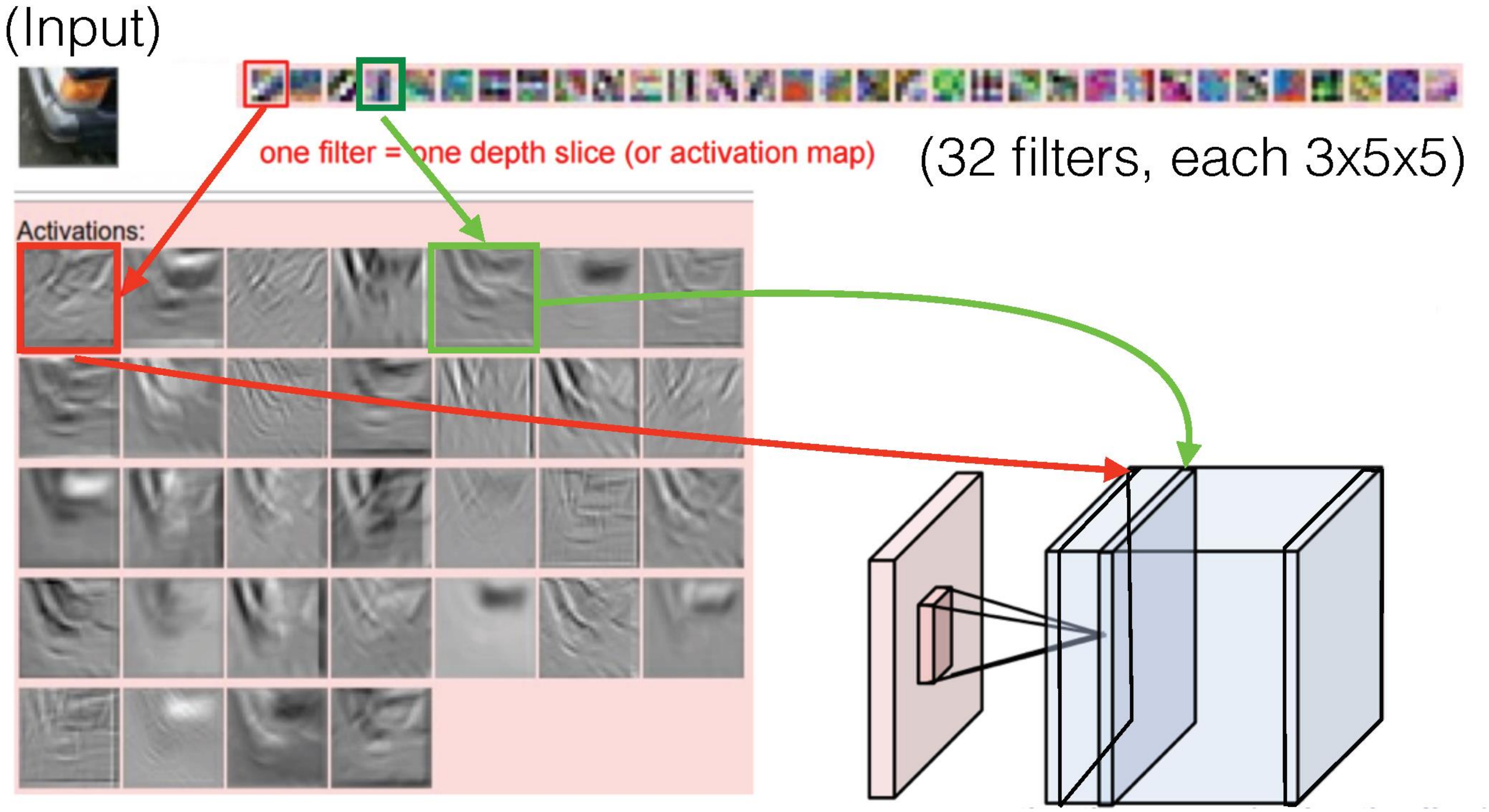


Figure: Andrej Karpathy

3D Activations

We can unravel the 3D cube and show each layer separately:

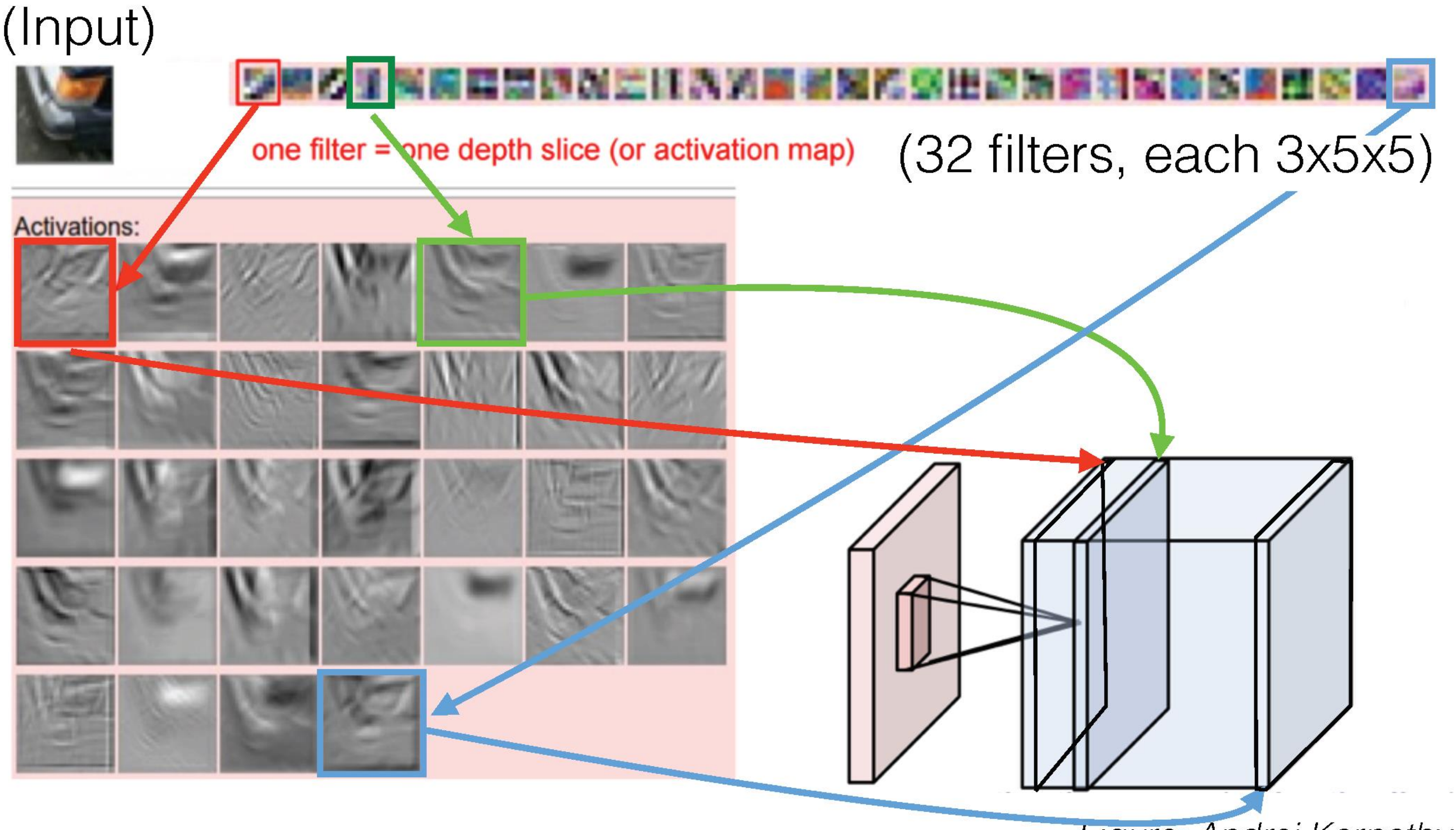
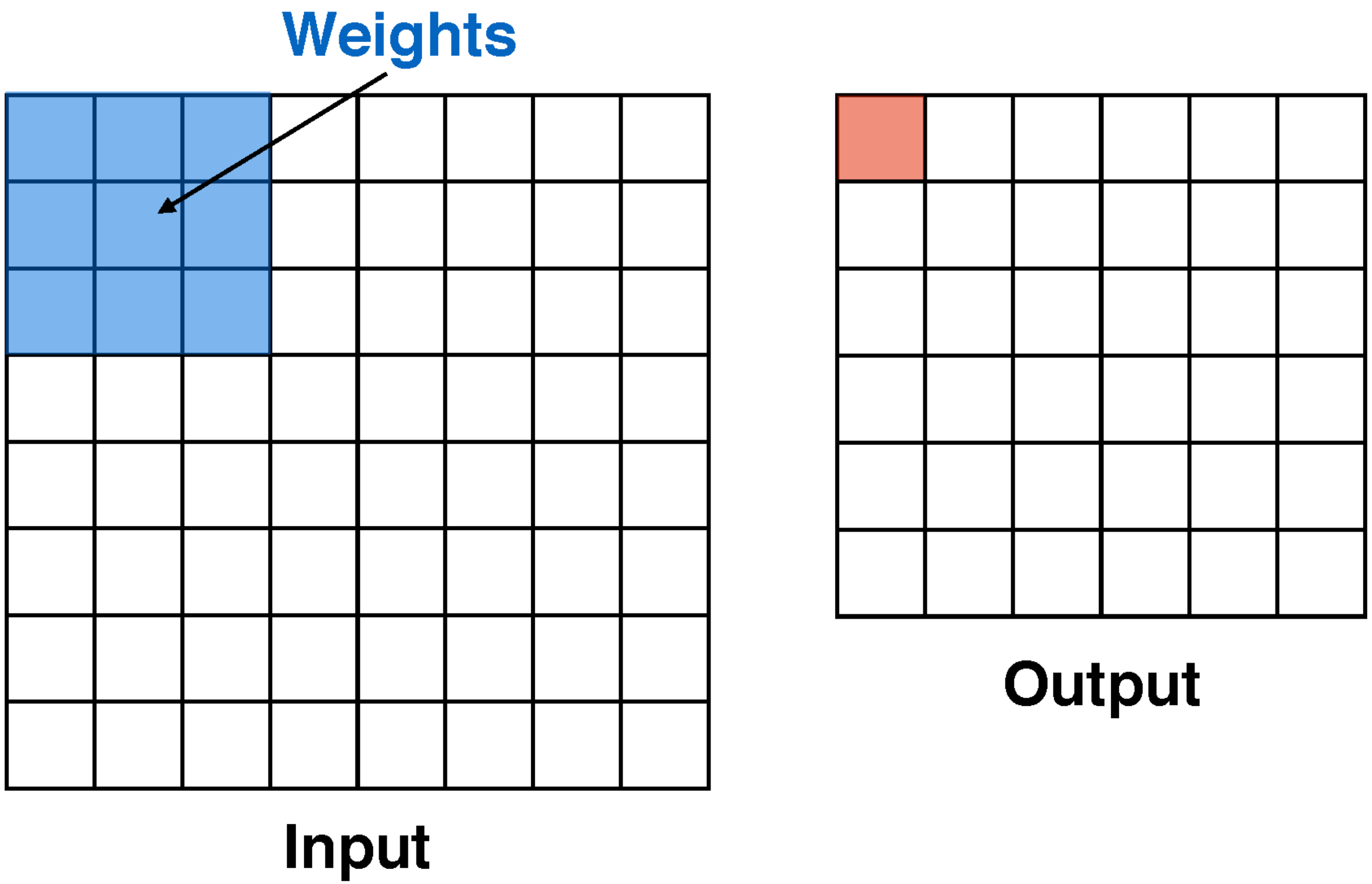


Figure: Andrej Karpathy

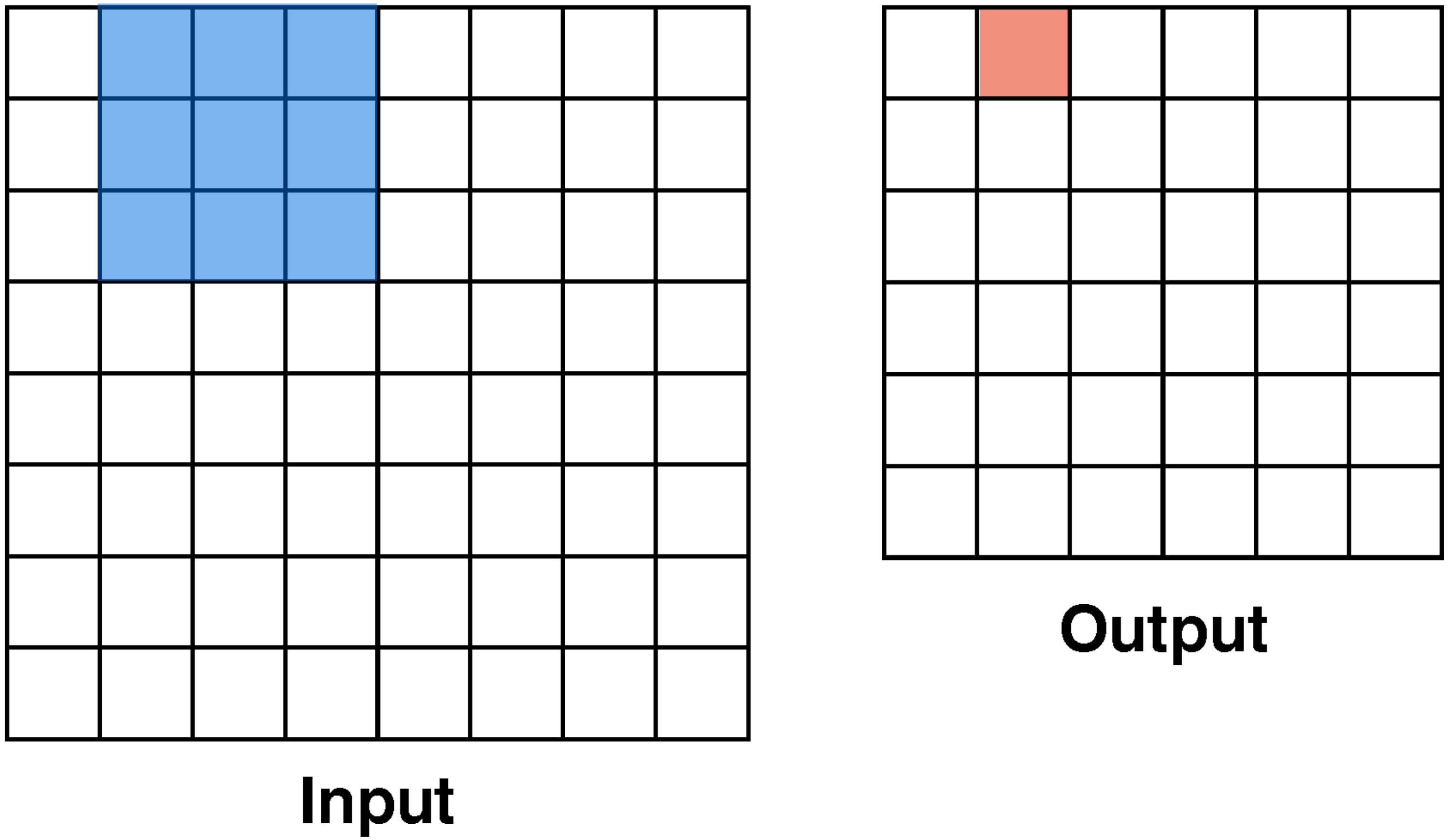
Convolution: Stride

During convolution, the weights "slide" along the input to generate each output



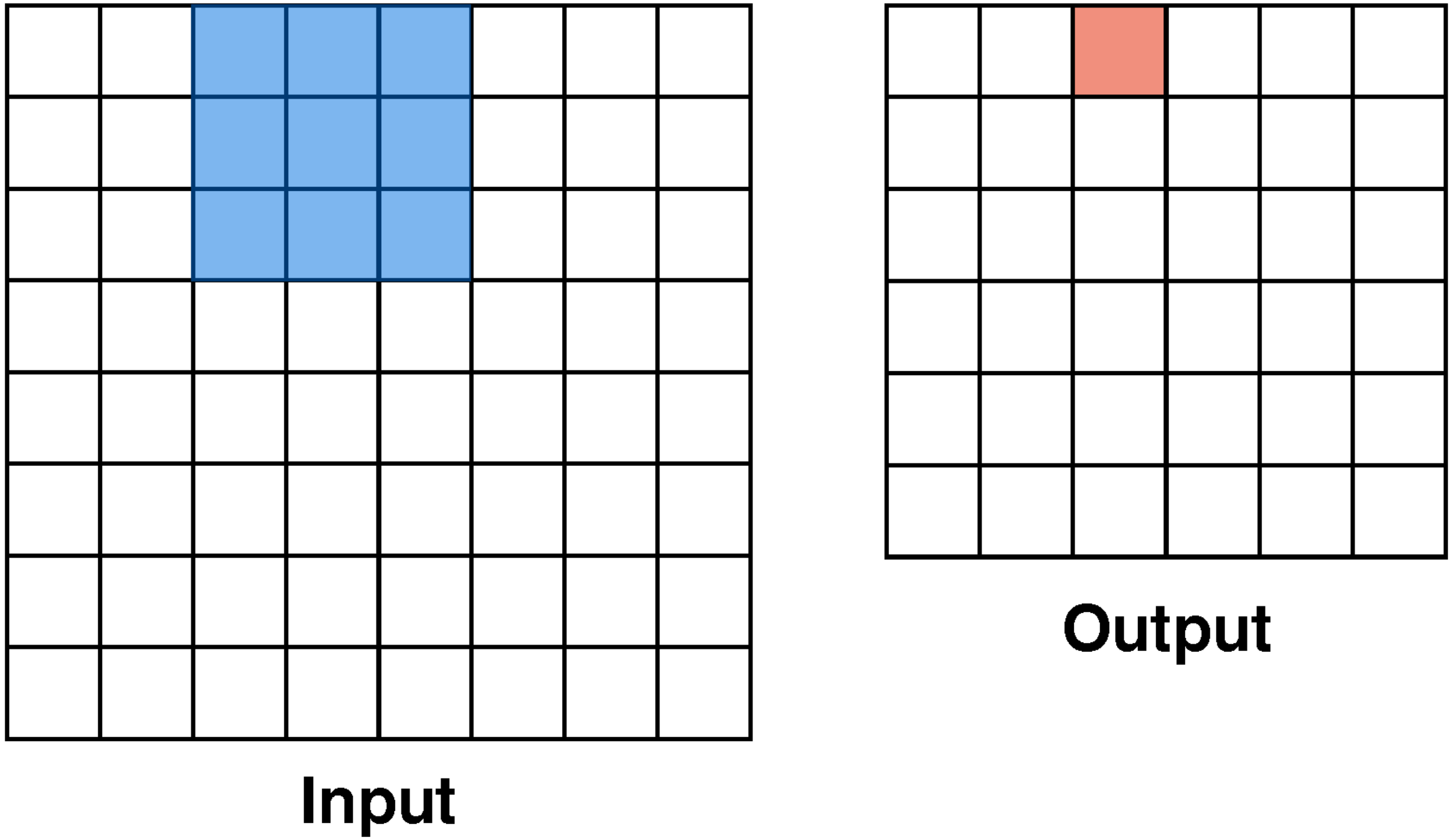
Convolution: Stride

During convolution, the weights “slide” along the input to generate each output



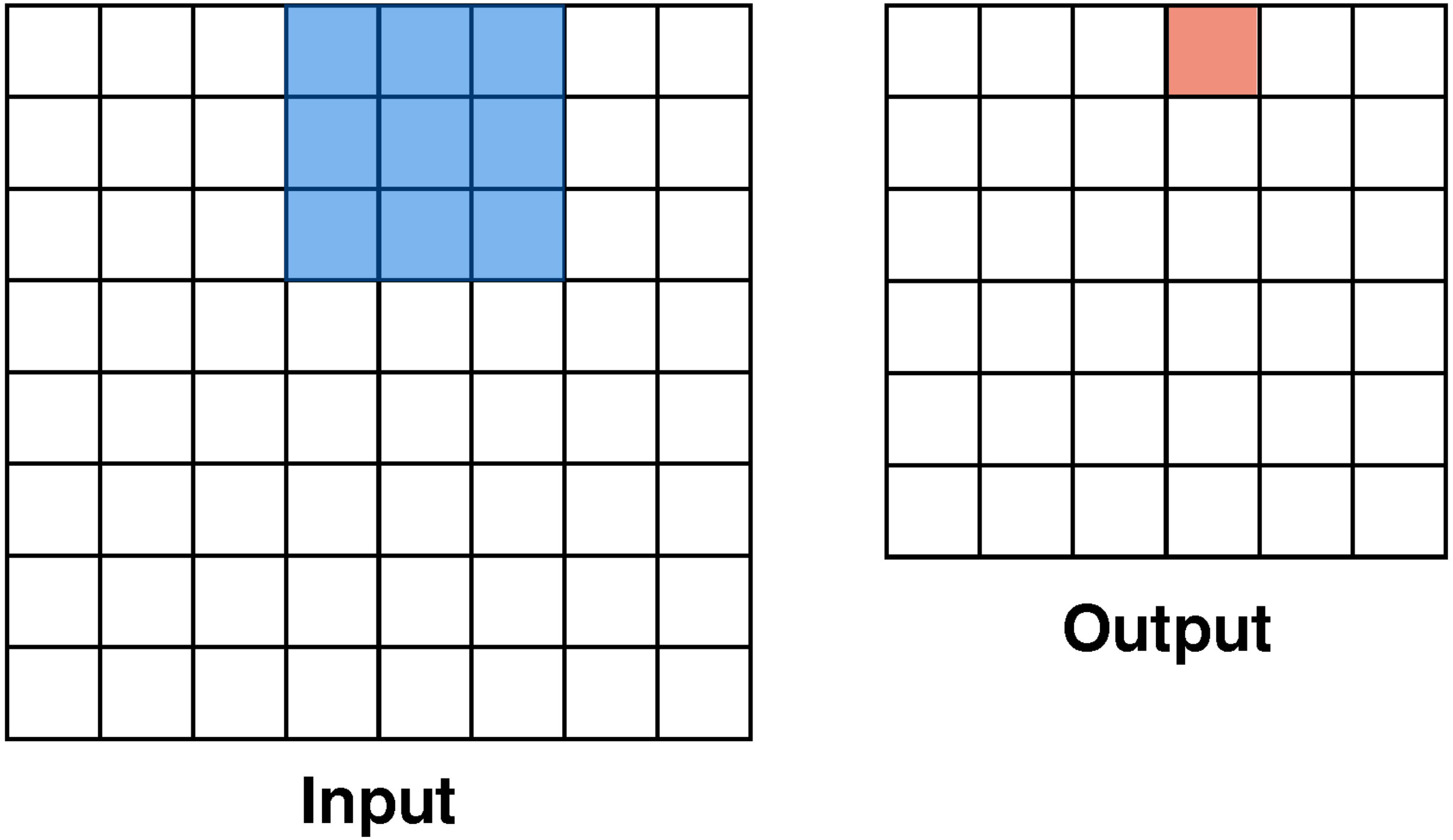
Convolution: Stride

During convolution, the weights “slide” along the input to generate each output



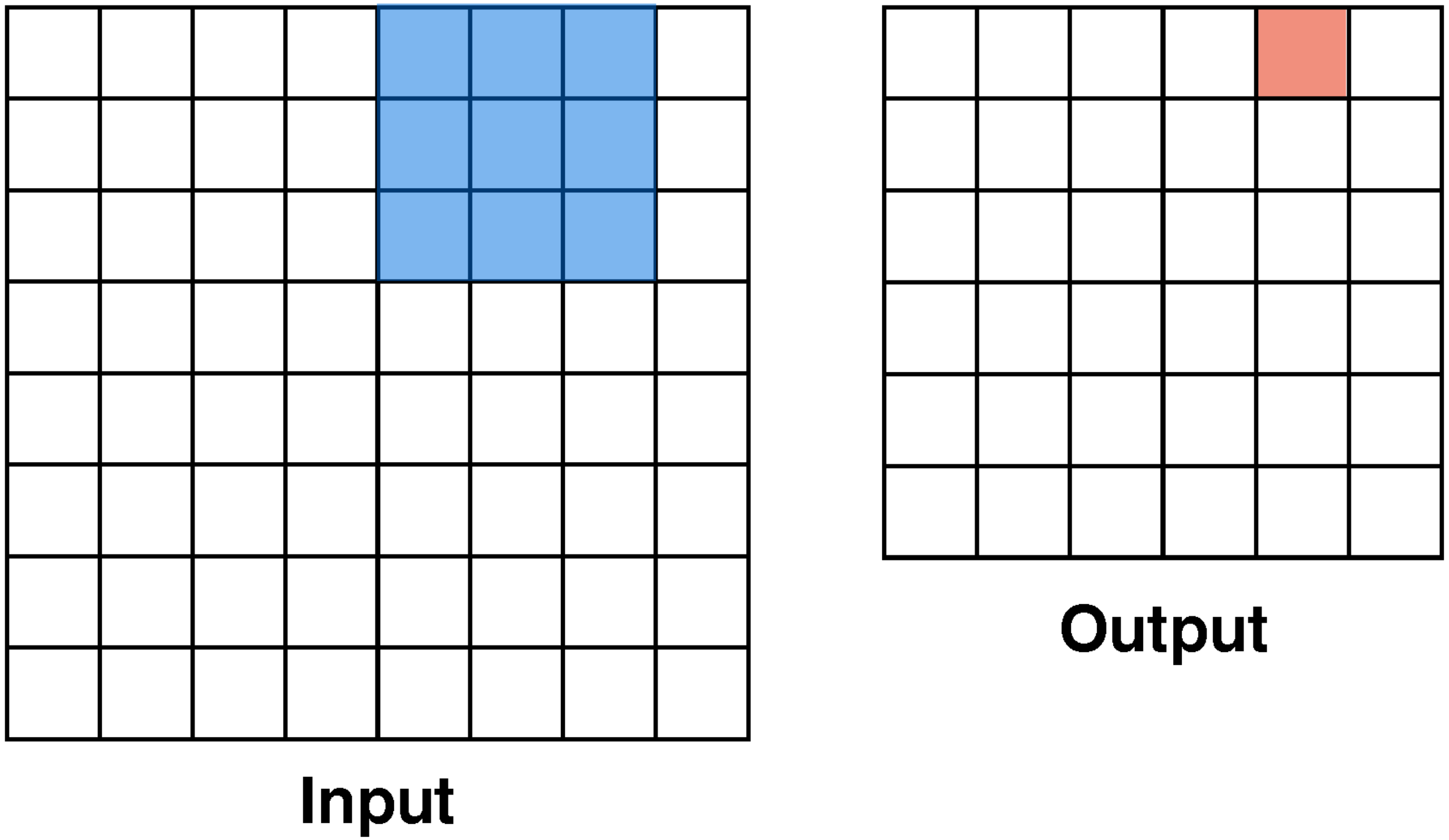
Convolution: Stride

During convolution, the weights “slide” along the input to generate each output



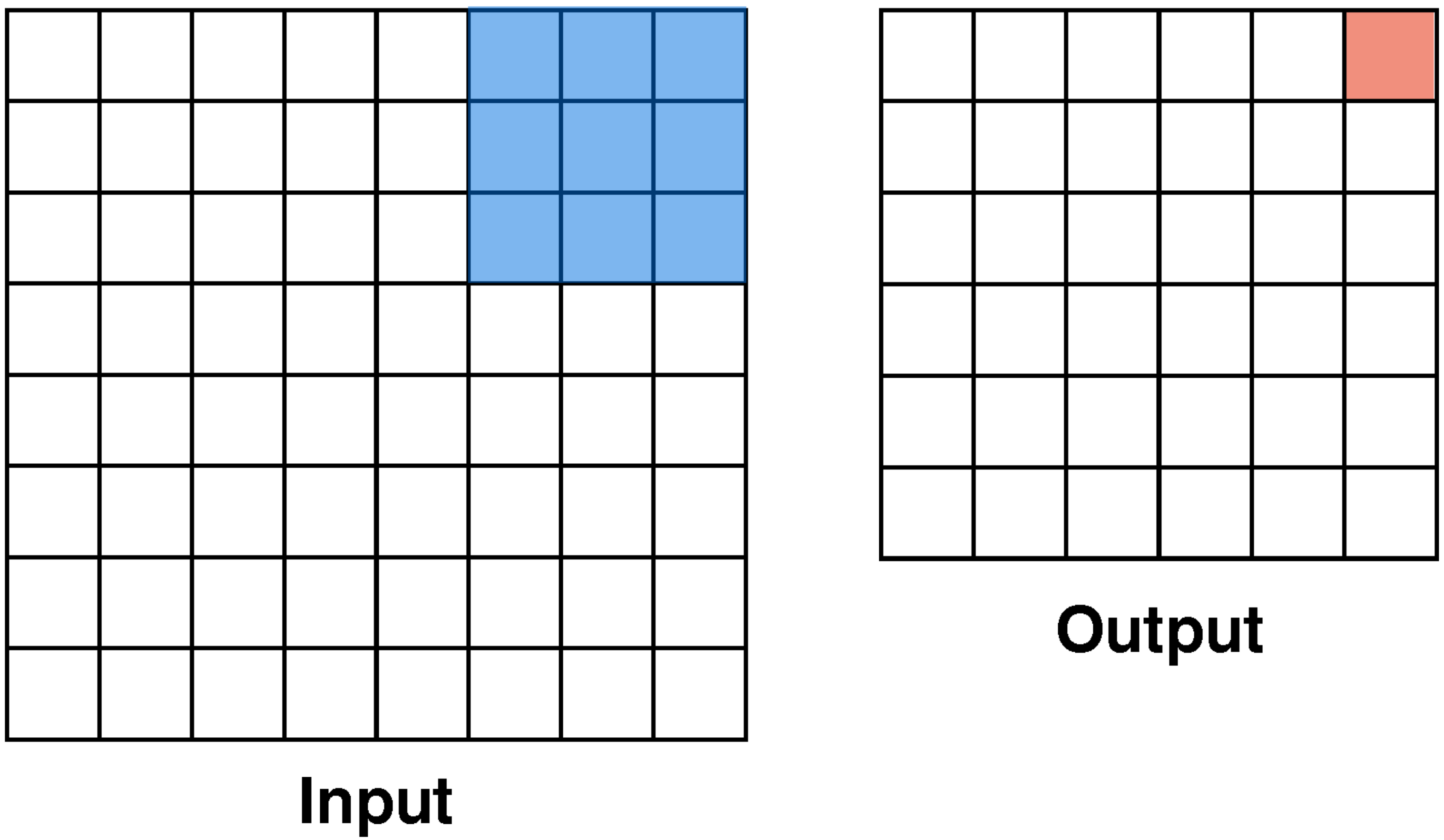
Convolution: Stride

During convolution, the weights “slide” along the input to generate each output



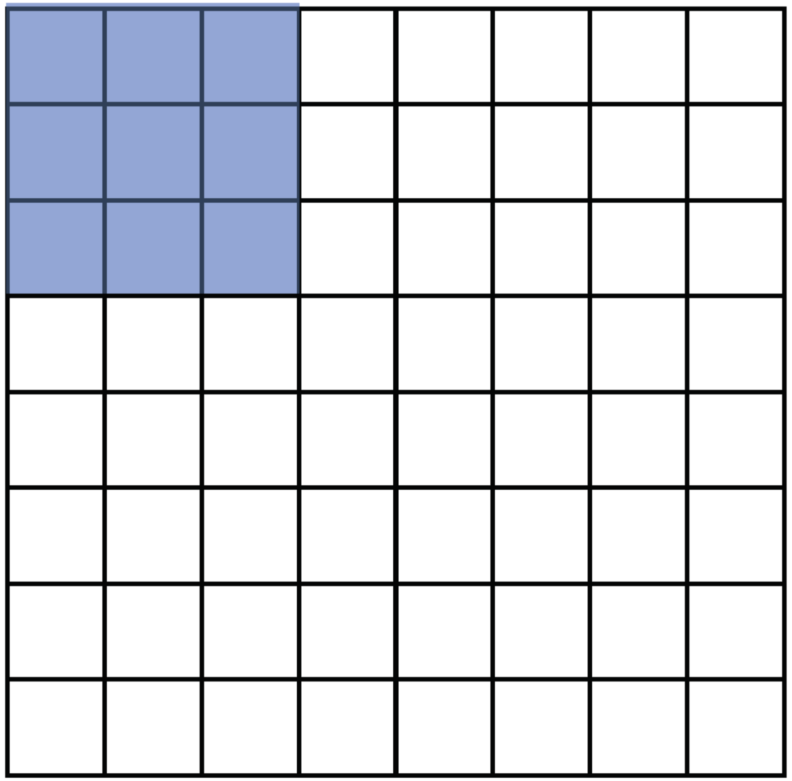
Convolution: Stride

During convolution, the weights “slide” along the input to generate each output



Convolution: Stride

During convolution, the weights “slide” along the input to generate each output



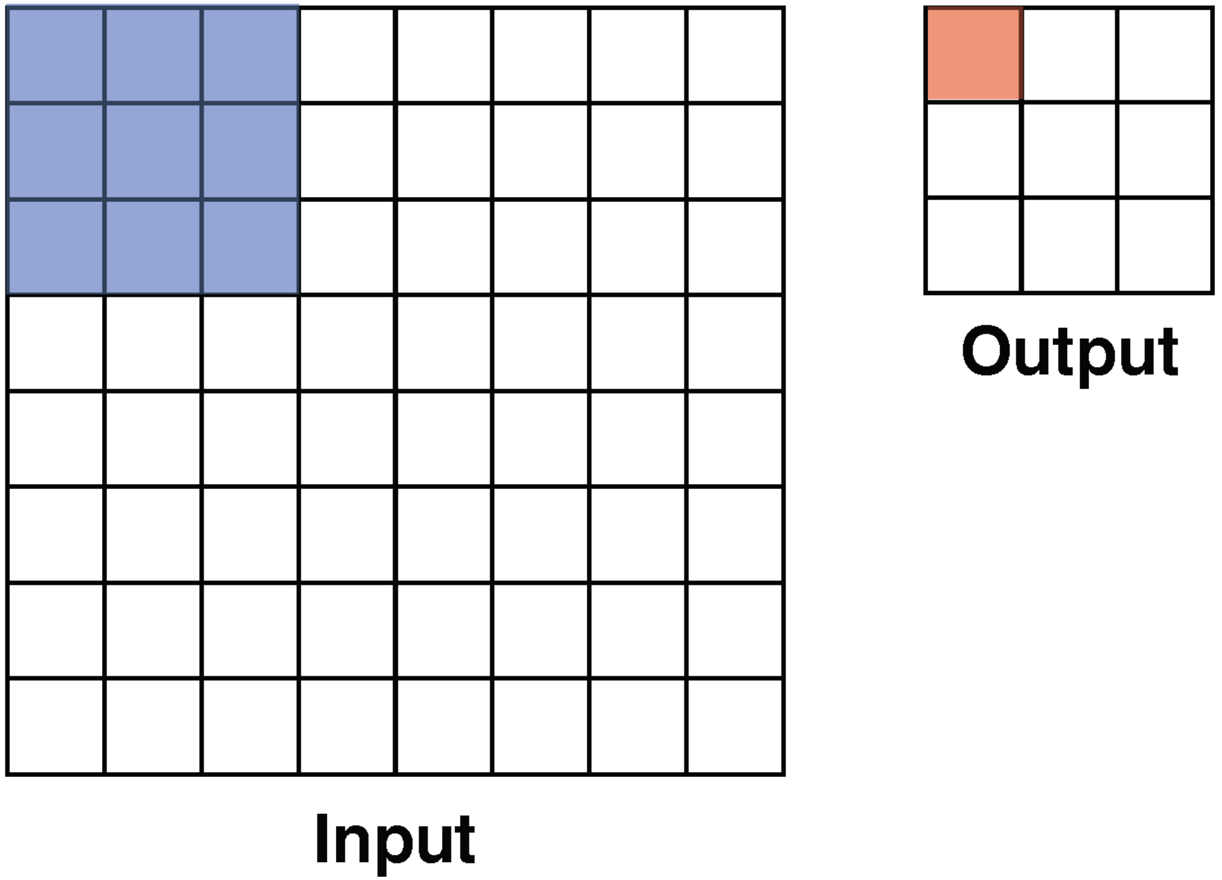
Recall that at each position, we are doing a **3D** sum:

$$h^r = \sum_{ijk} x^r{}_{ijk} W_{ijk} + b$$

(channel, row, column)

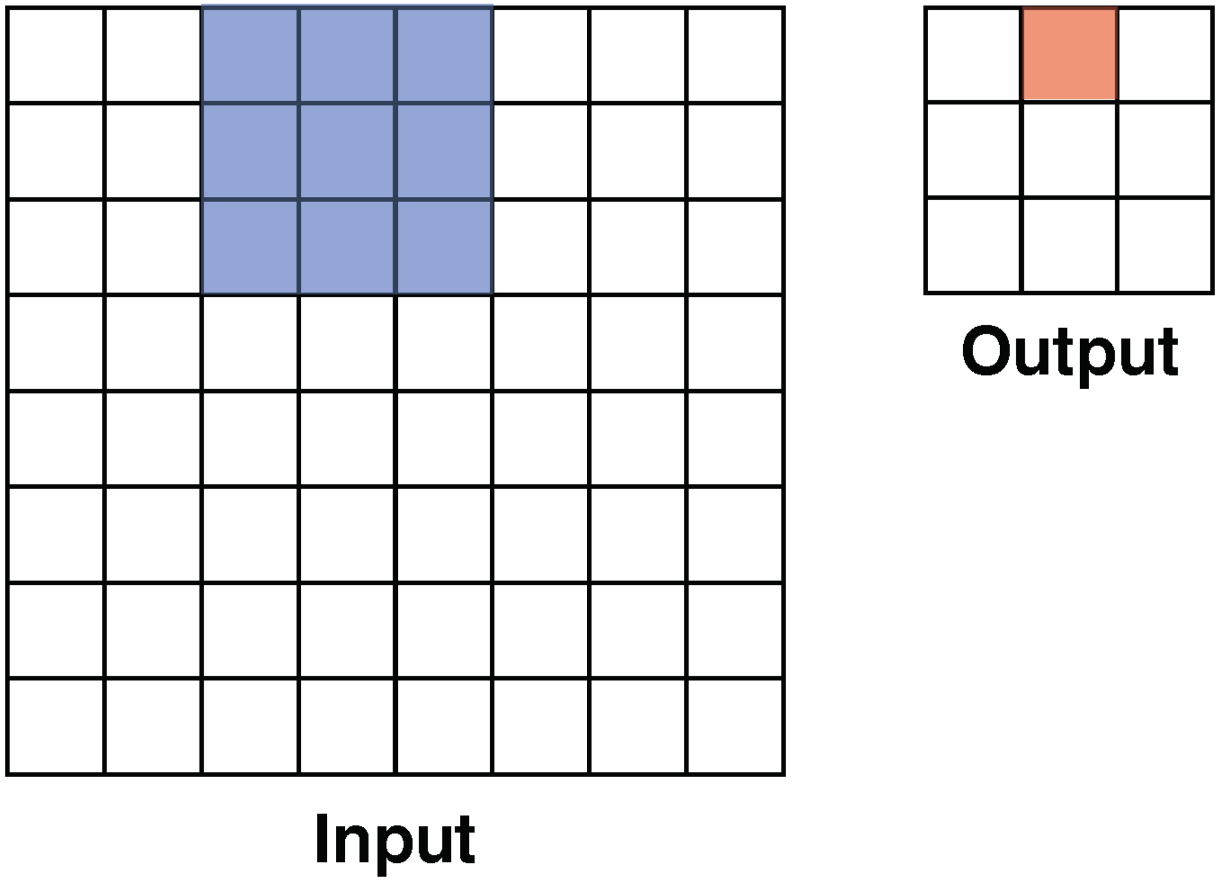
Convolution: Stride

But we can also convolve with a **stride**, e.g. stride = 2



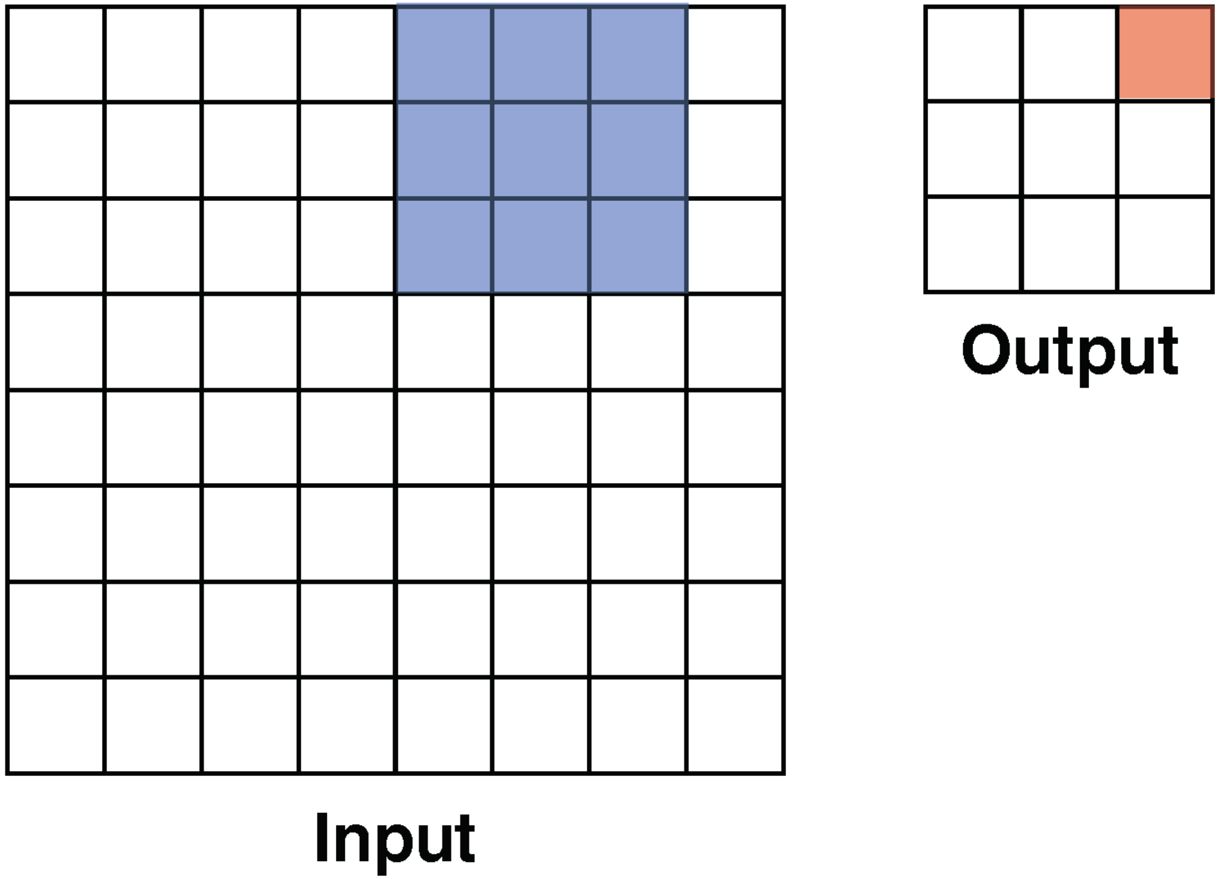
Convolution: Stride

But we can also convolve with a **stride**, e.g. stride = 2



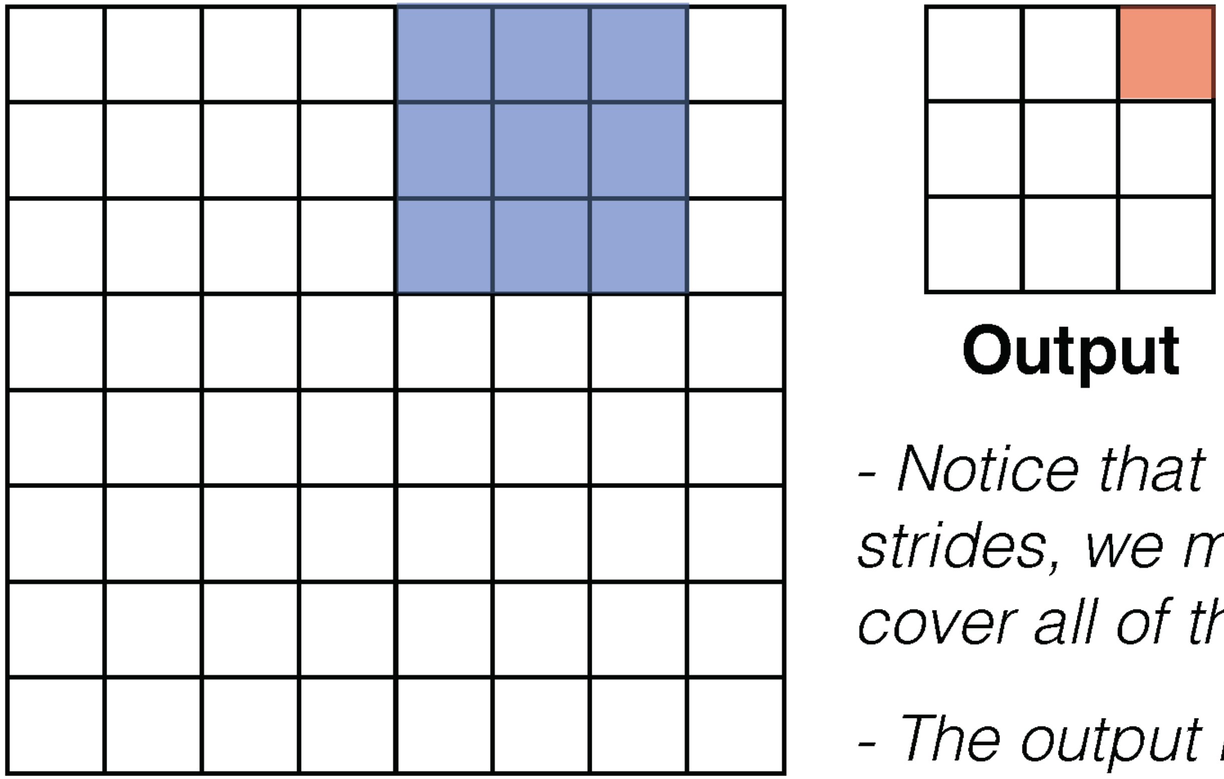
Convolution: Stride

But we can also convolve with a **stride**, e.g. stride = 2



Convolution: Stride

But we can also convolve with a **stride**, e.g. stride = 2



Input

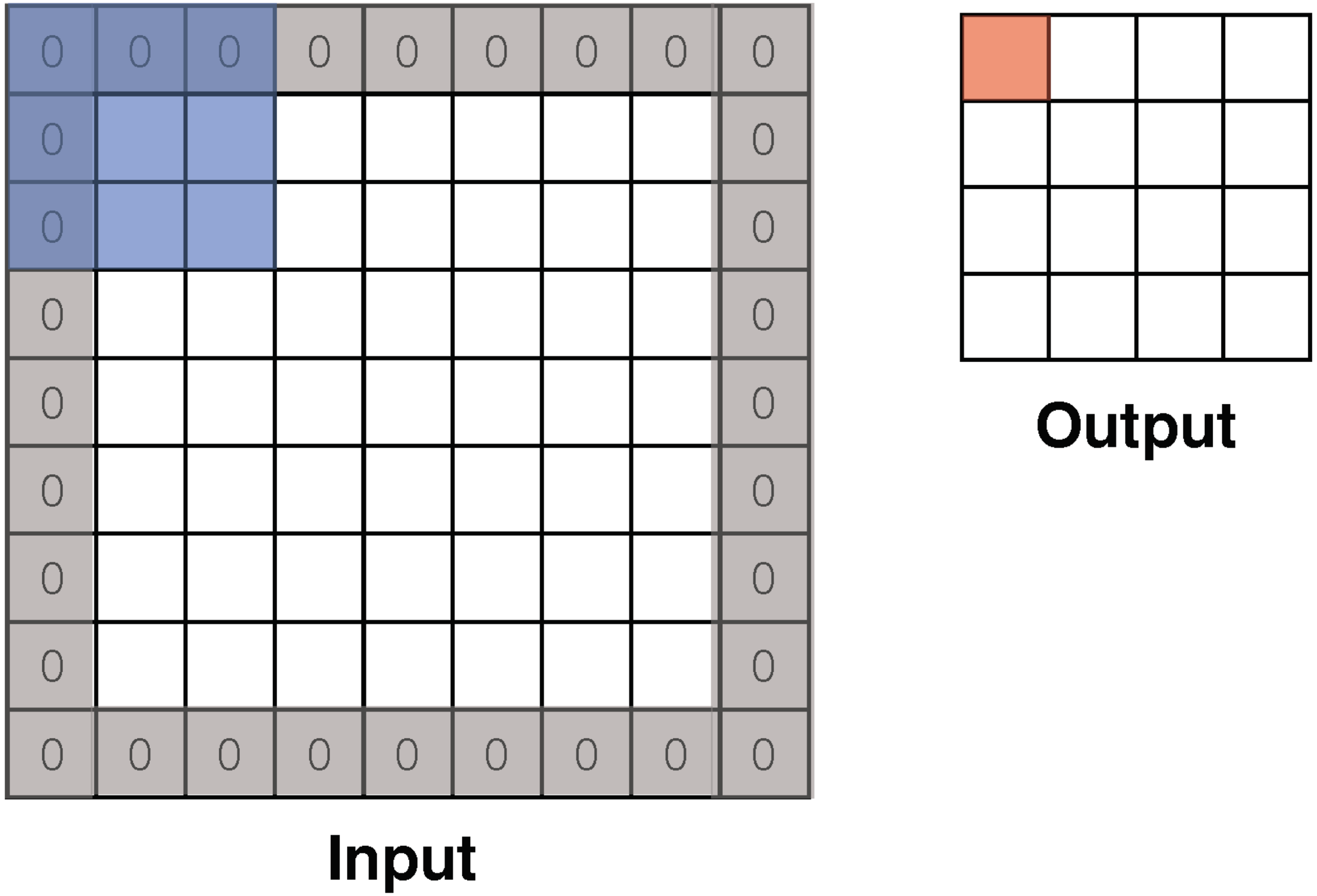
Output

- *Notice that with certain strides, we may not be able to cover all of the input*

- *The output is also half the size of the input*

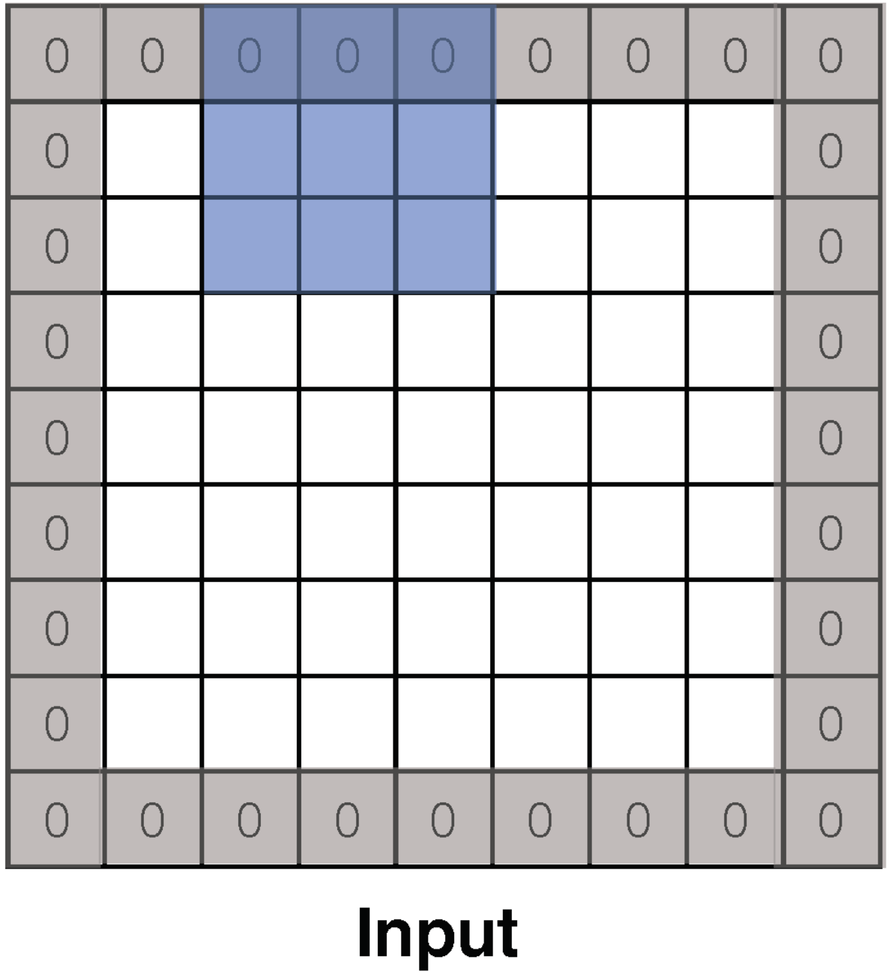
Convolution: Padding

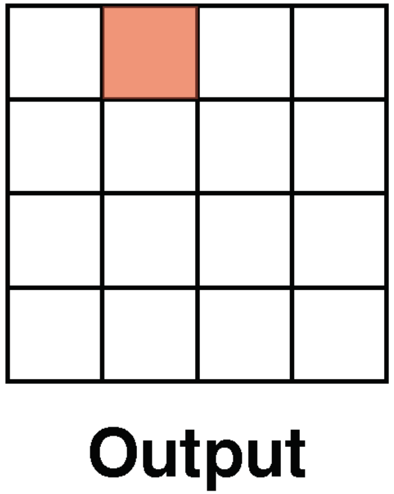
We can also pad the input with zeros.
Here, **pad = 1, stride = 2**



Convolution: Padding

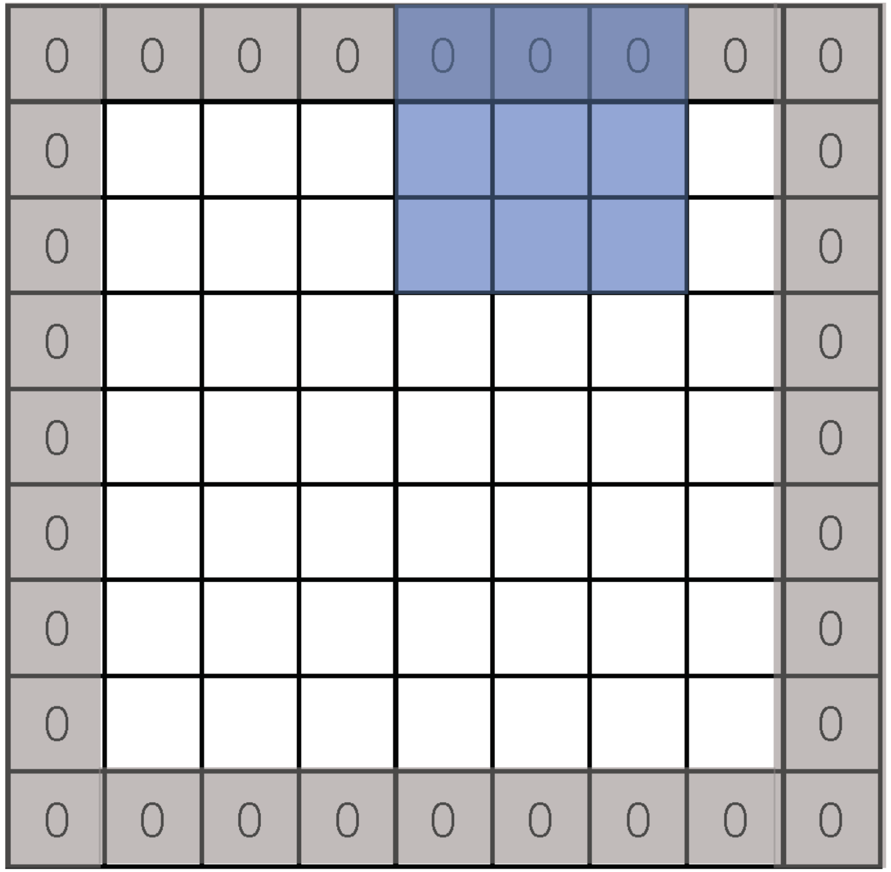
We can also pad the input with zeros.
Here, **pad = 1, stride = 2**



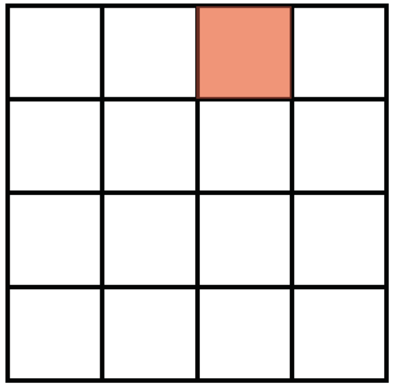


Convolution: Padding

We can also pad the input with zeros.
Here, **pad = 1, stride = 2**



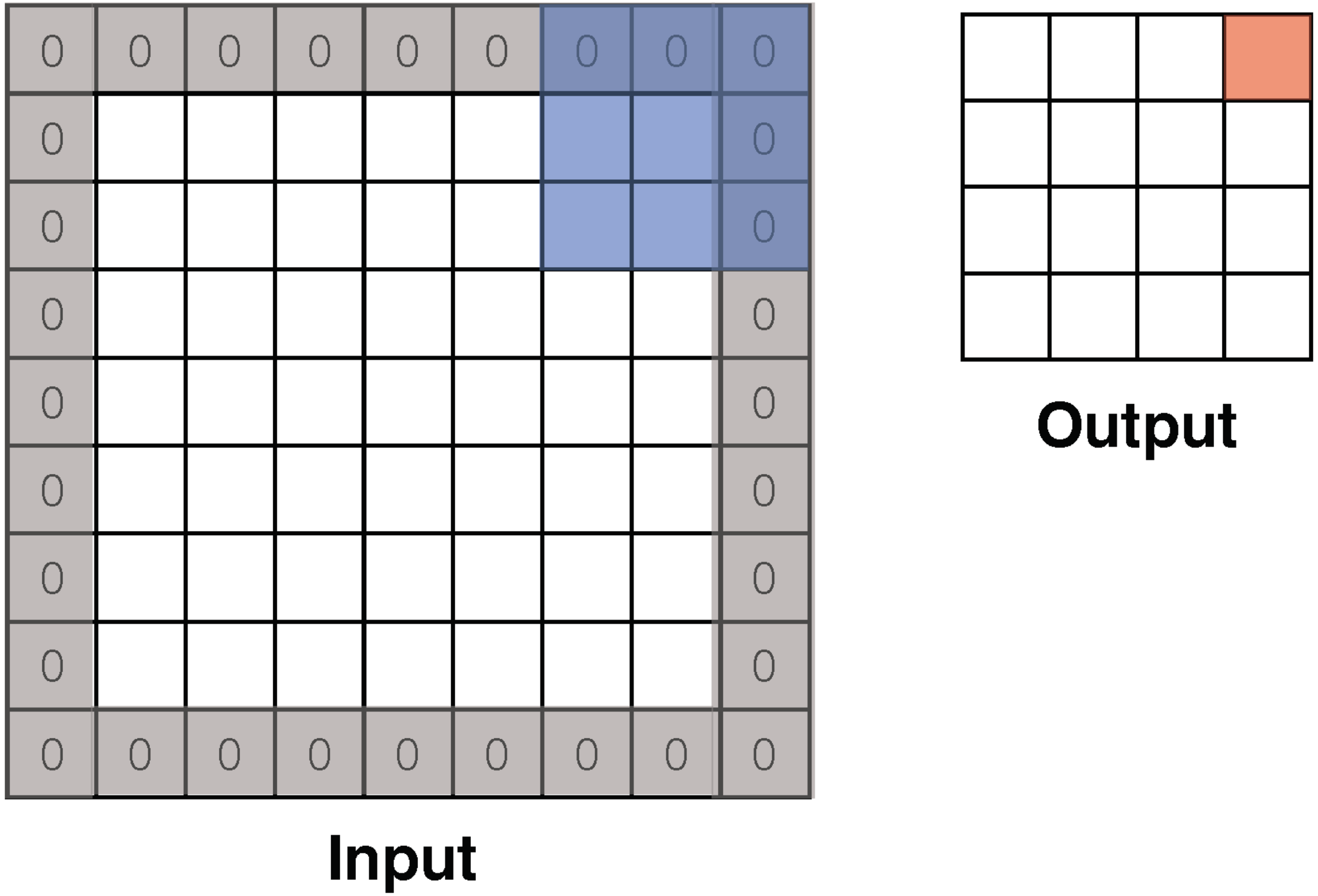
Input



Output

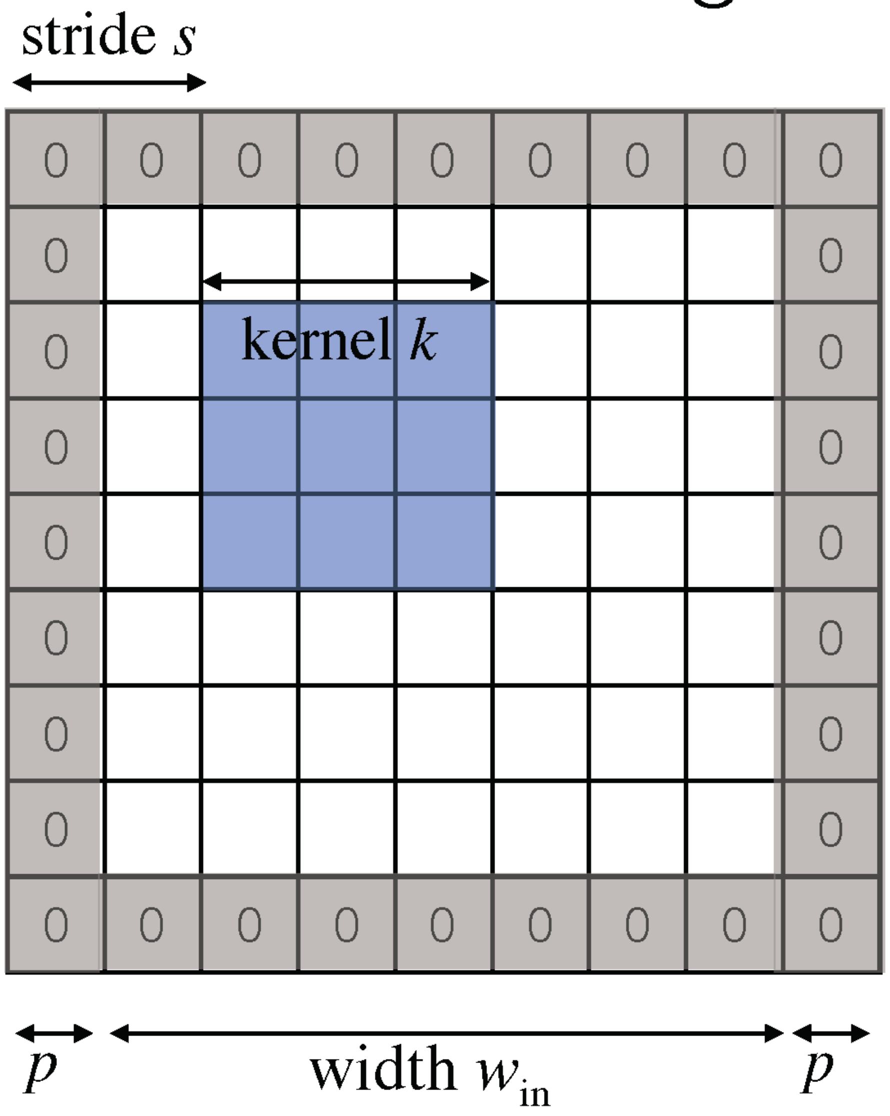
Convolution: Padding

We can also pad the input with zeros.
Here, **pad = 1, stride = 2**



Convolution:

How big is the output?



We can choose different parameters for the convolution:

$$w_{out} = \frac{w_{in} + 2p - k}{s} + 1$$

Such that w_{out} is an int.

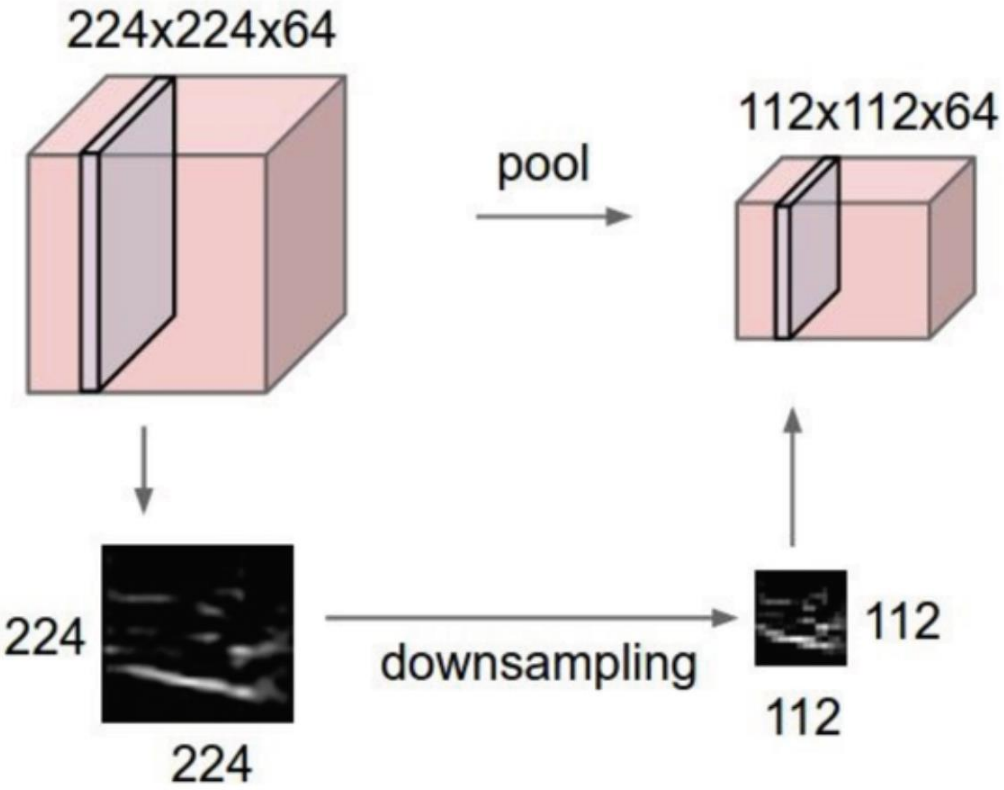
Pooling

For most ConvNets, **convolution** is often followed by **pooling**:

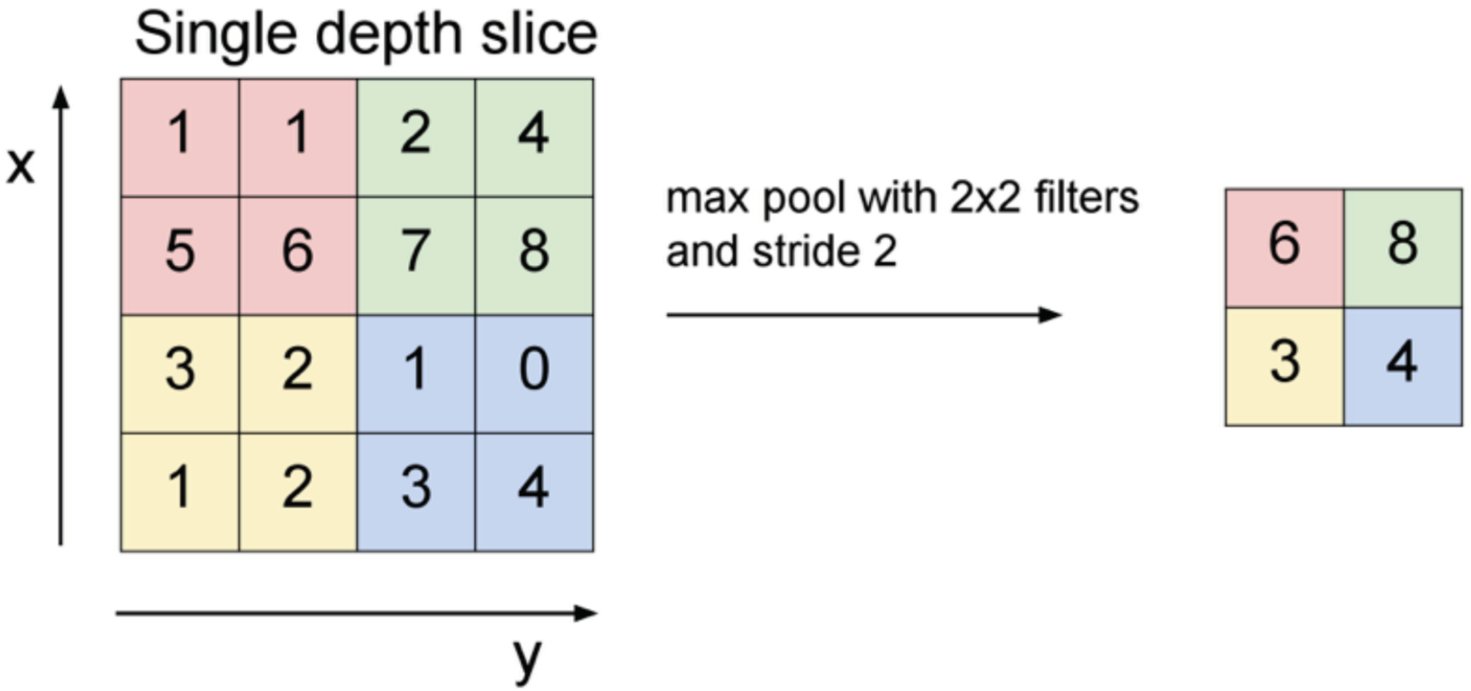
- Creates a smaller representation while retaining the most important information
- The “max” operation is the most common

Pooling

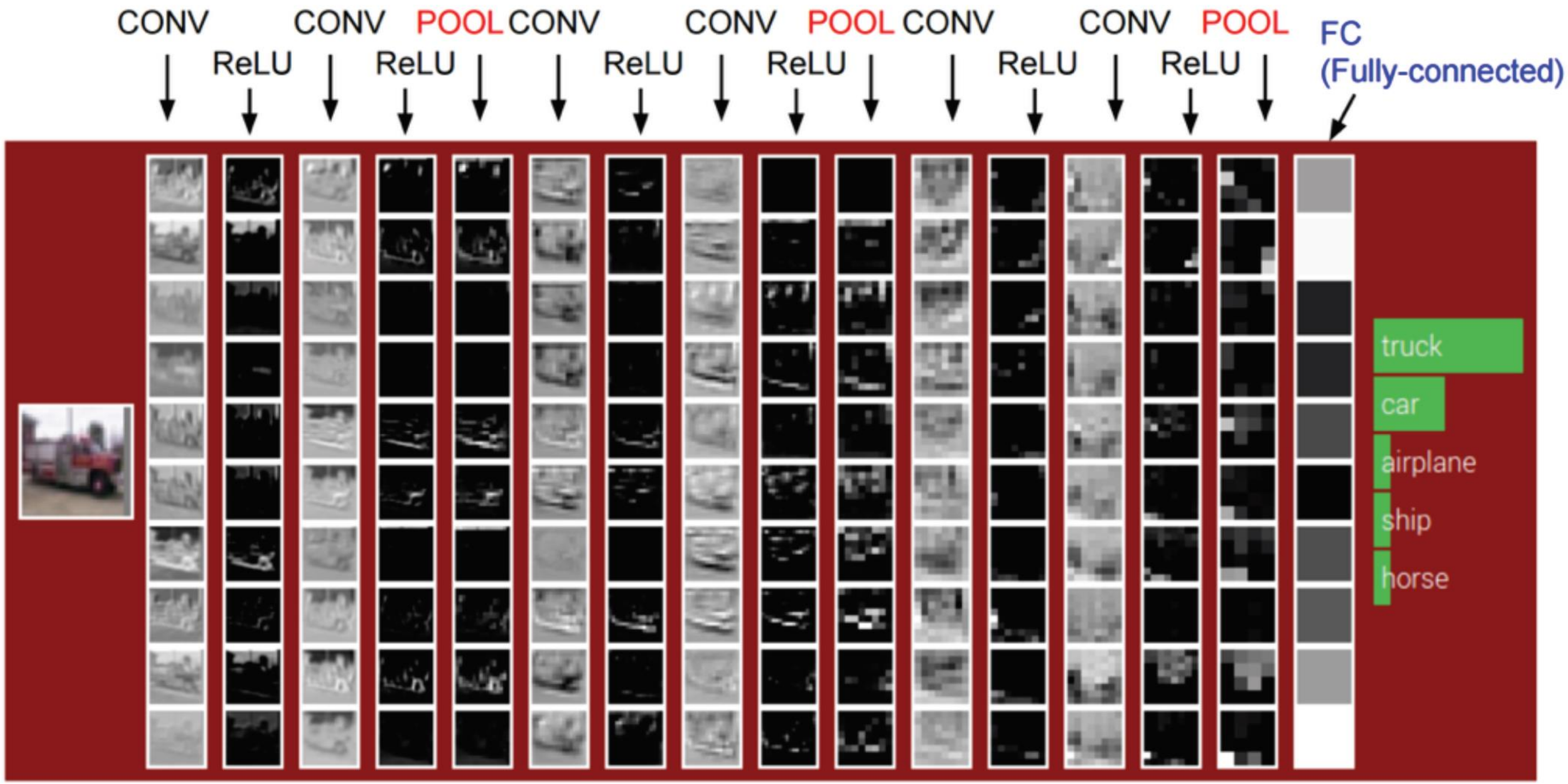
- makes the representations smaller and more manageable
- operates over each activation map independently:



Max Pooling


Single depth slice
x
y
1 1 2 4
5 6 7 8
3 2 1 0
1 2 3 4
max pool with 2x2 filters
and stride 2
6 8
3 4

Example ConvNet



10x3x3 conv filters, stride 1, pad 1

2x2 pool filters, stride 2

Figure: Andrej Karpathy

Visualization example

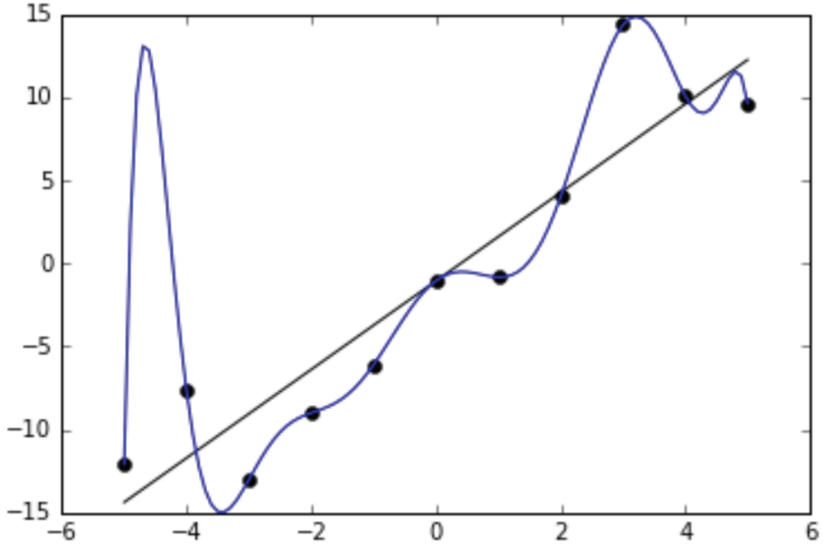
- https://www.youtube.com/watch?v=AgkfIQ4IGaM

Overfitting

Overfitting: modeling noise in the training set instead of the “true” underlying relationship

Underfitting: insufficiently modeling the relationship in the training set

General rule: models that are “bigger” or have more capacity are more likely to overfit



[Image: https://en.wikipedia.org/wiki/File:Overfitted_Data.png]

Plot the loss

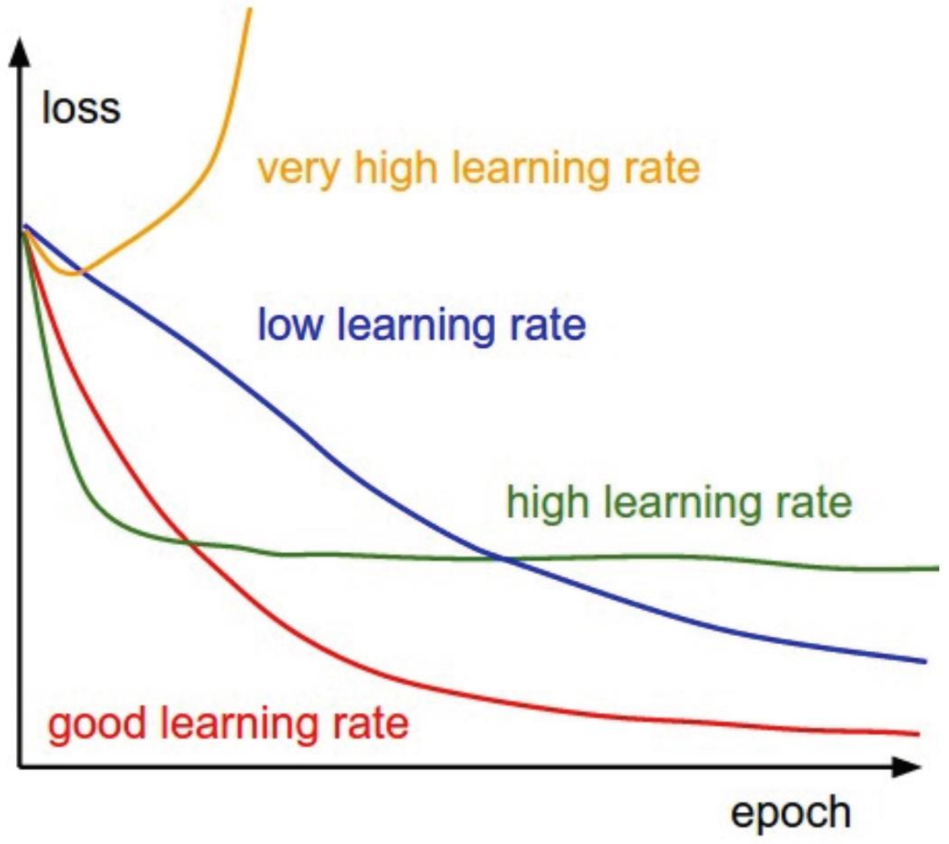
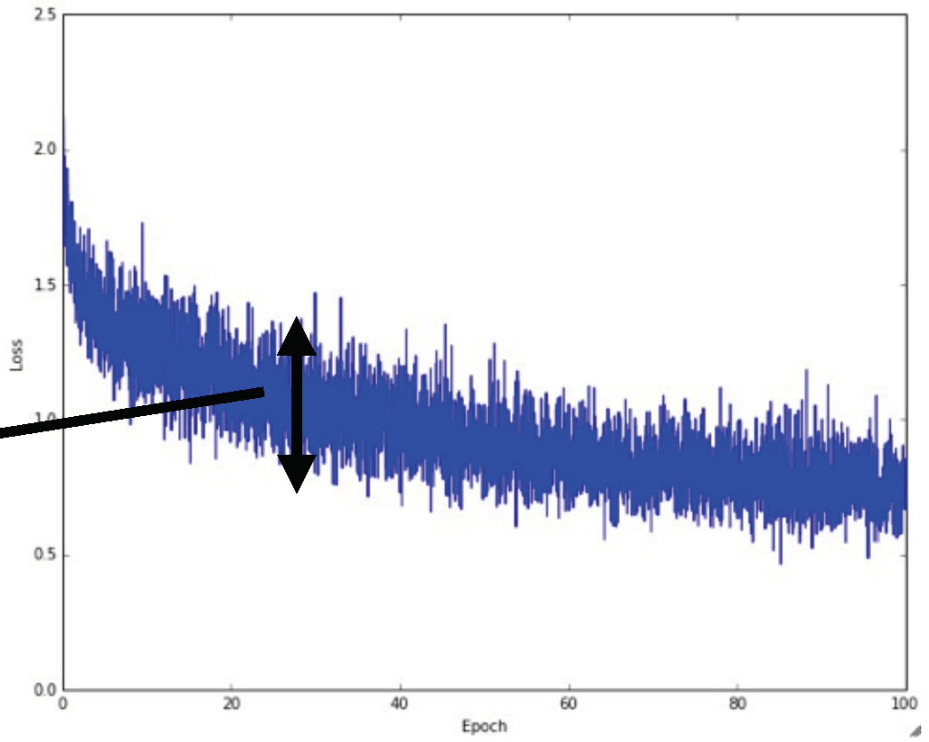


Figure: Andrej Karpathy

Typical training loss:

Why is it varying so rapidly?

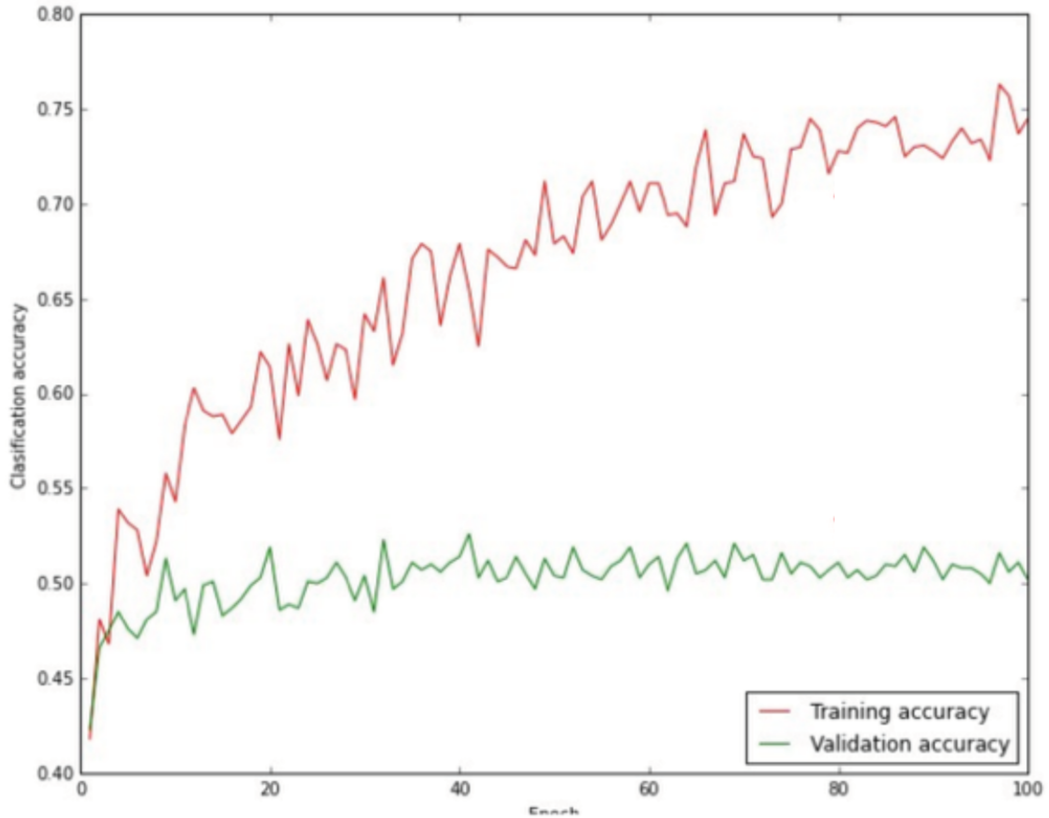
The width of the curve is related to the batchsize — if too noisy, increase the batch size



Possibly too linear
(learning rate too small)

Figure: Andrej Karpathy

Visualize the accuracy

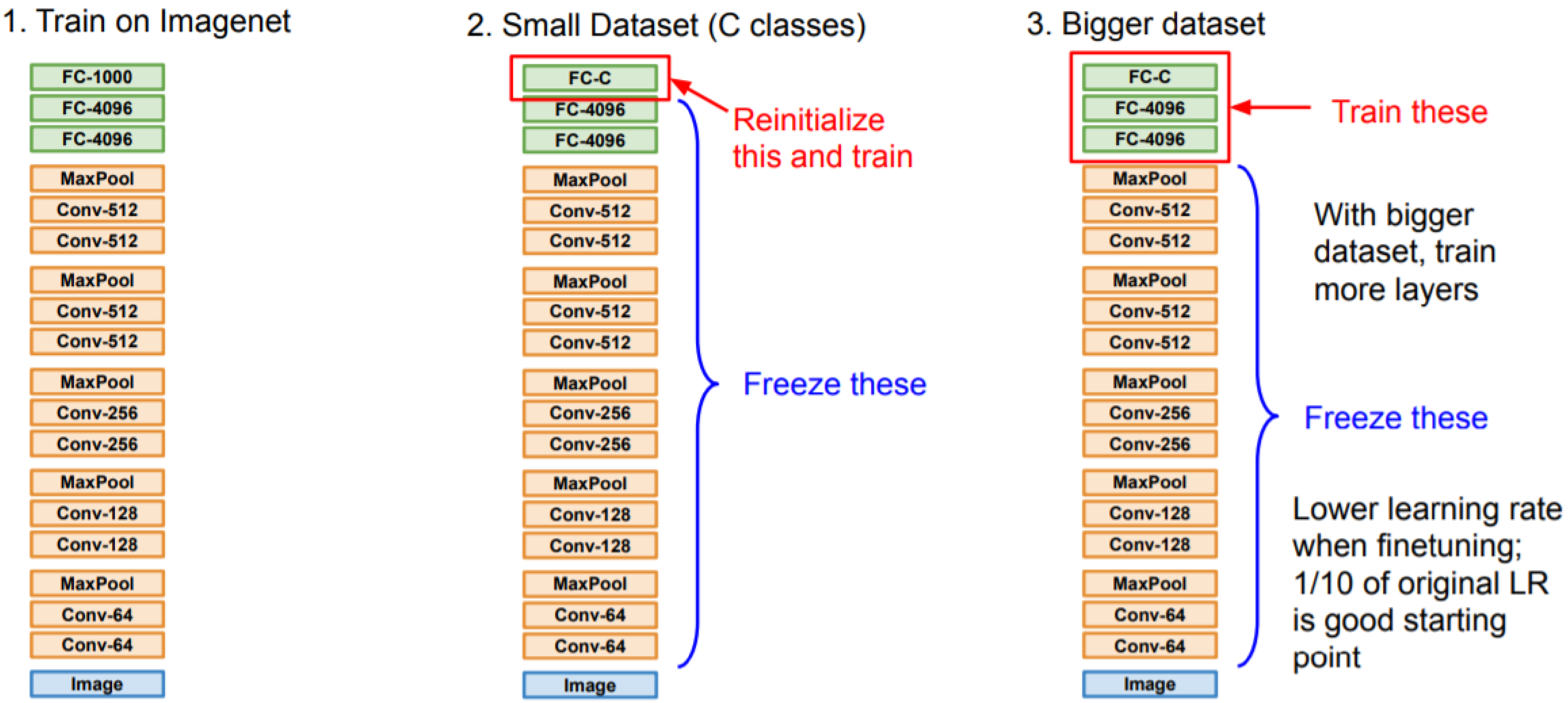


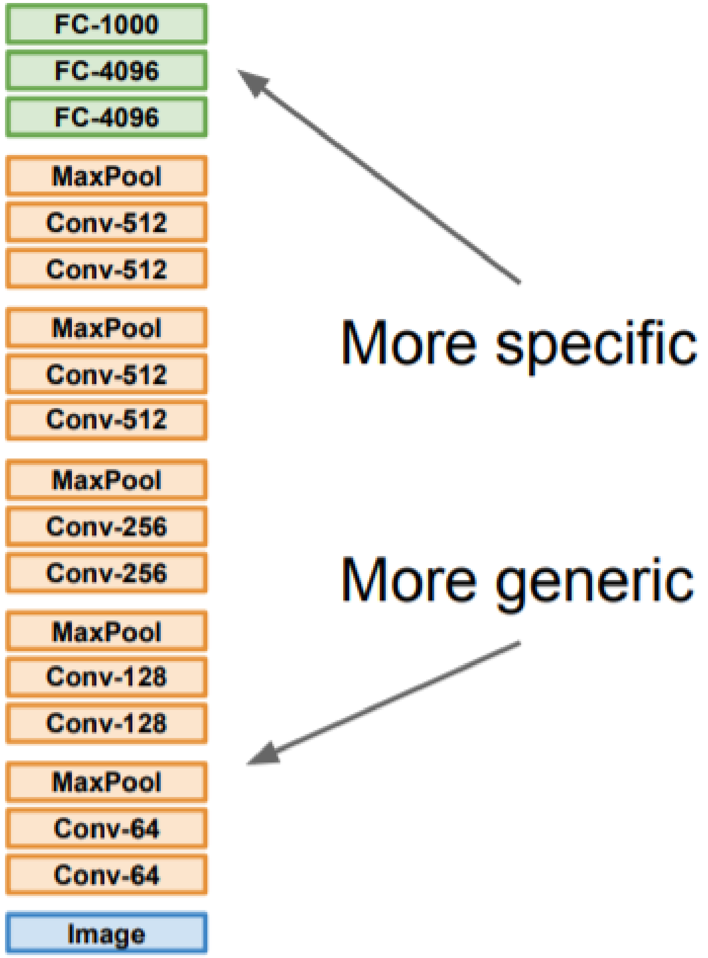
Big gap: overfitting
(increase regularization)

No gap: underfitting
(increase model capacity,
make layers bigger
or decrease regularization)

Figure: Andrej Karpathy

Transfer learning





	very similar dataset	**very different dataset**
very little data	Use Linear Classifier on top layer	You're in trouble… Try linear classifier from different stages
quite a lot of data	Finetune a few layers	Finetune a larger number of layers