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/

contents

- **Chain rule**
- ConvNets
  - Convolution layer
  - Pooling layer
- Overfitting
- Architectures
  - Alexnet (dropout)
  - VGG
  - ResNet (batch norm)

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!

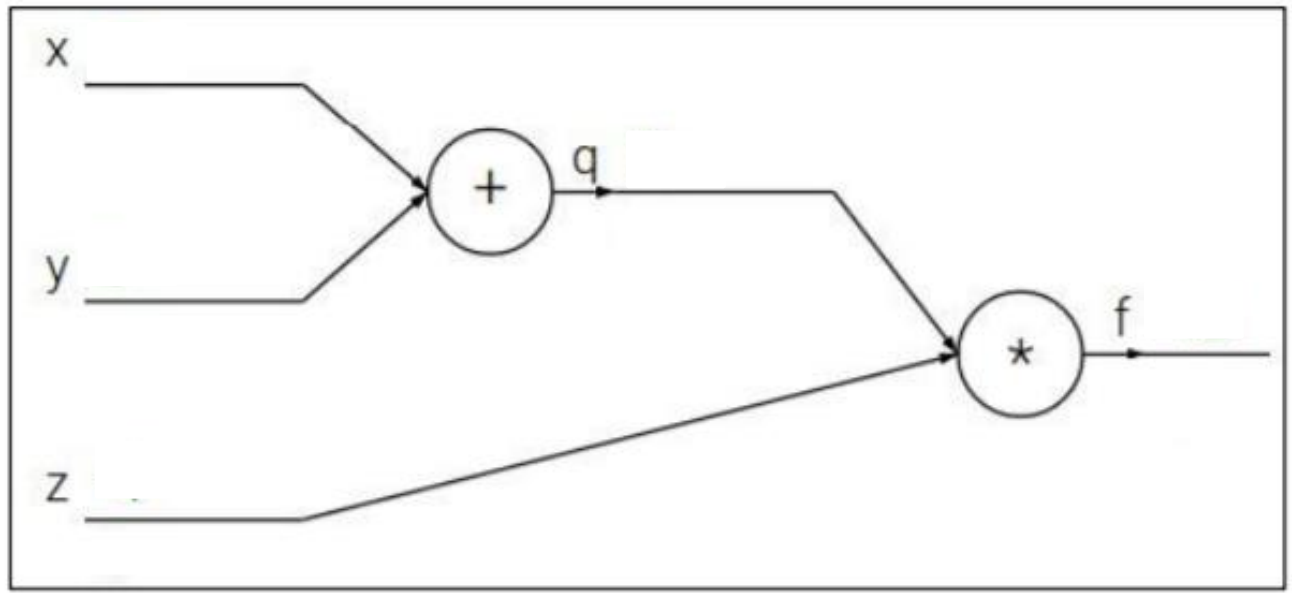
Good idea- back propagation

- We are using the chain rule for gradient calculation:

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

Backpropagation: a simple example

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

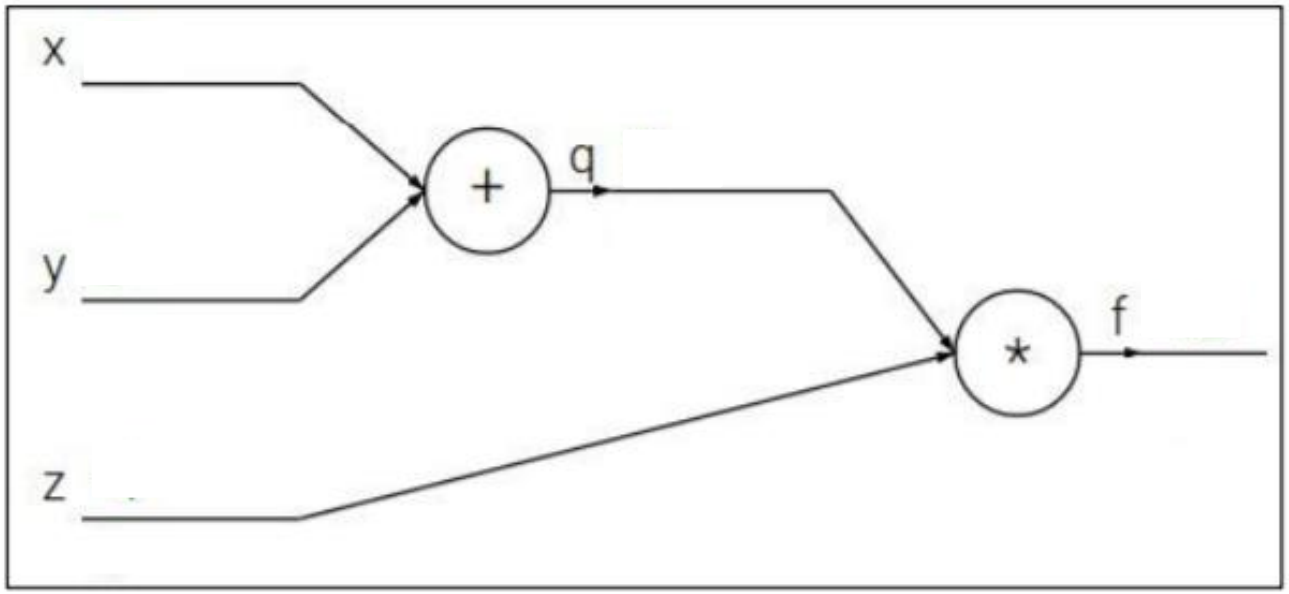


Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation: a simple example

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

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

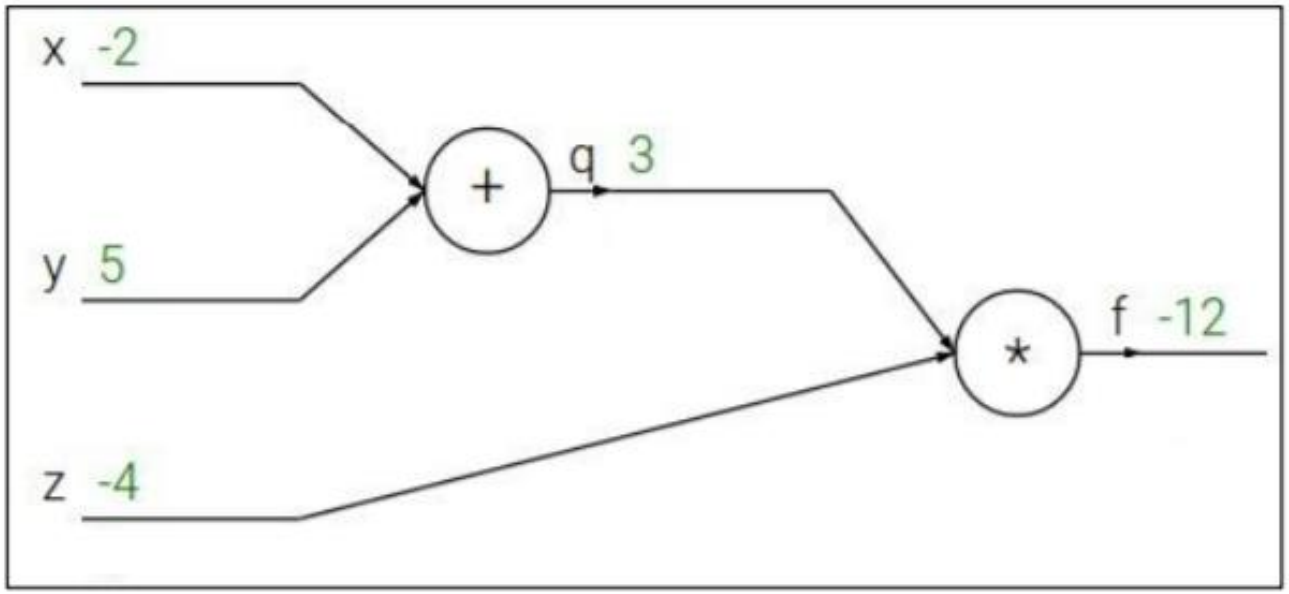


Want: $\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}$

Backpropagation: a simple example

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

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



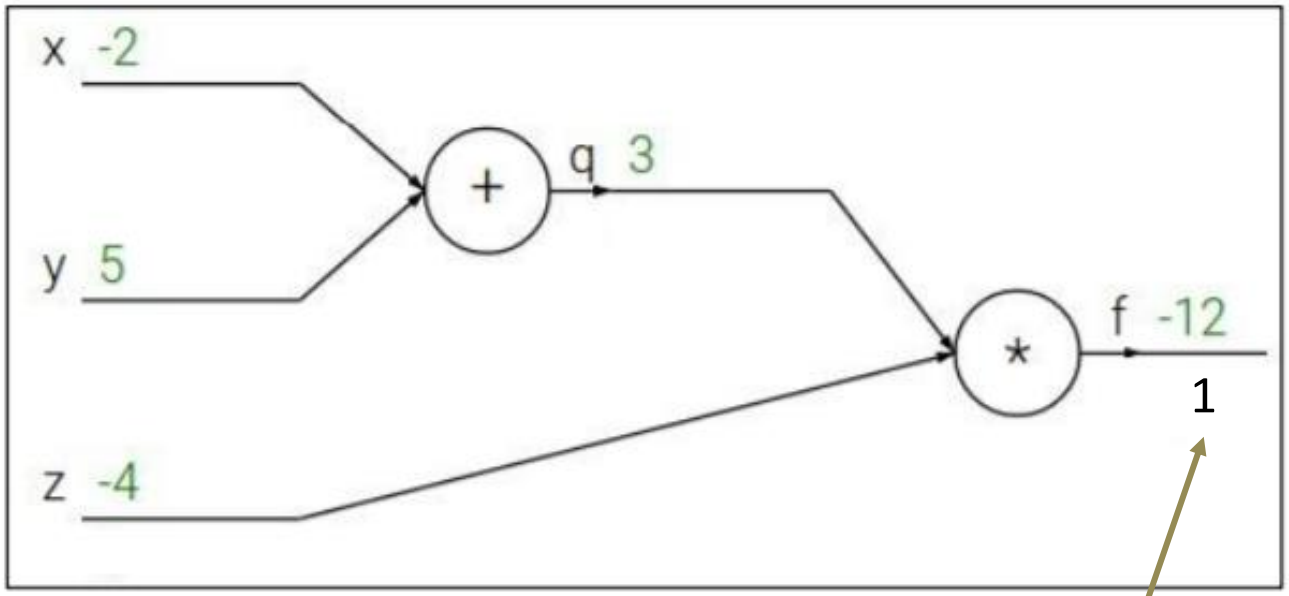
This is called- **the forward pass**

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

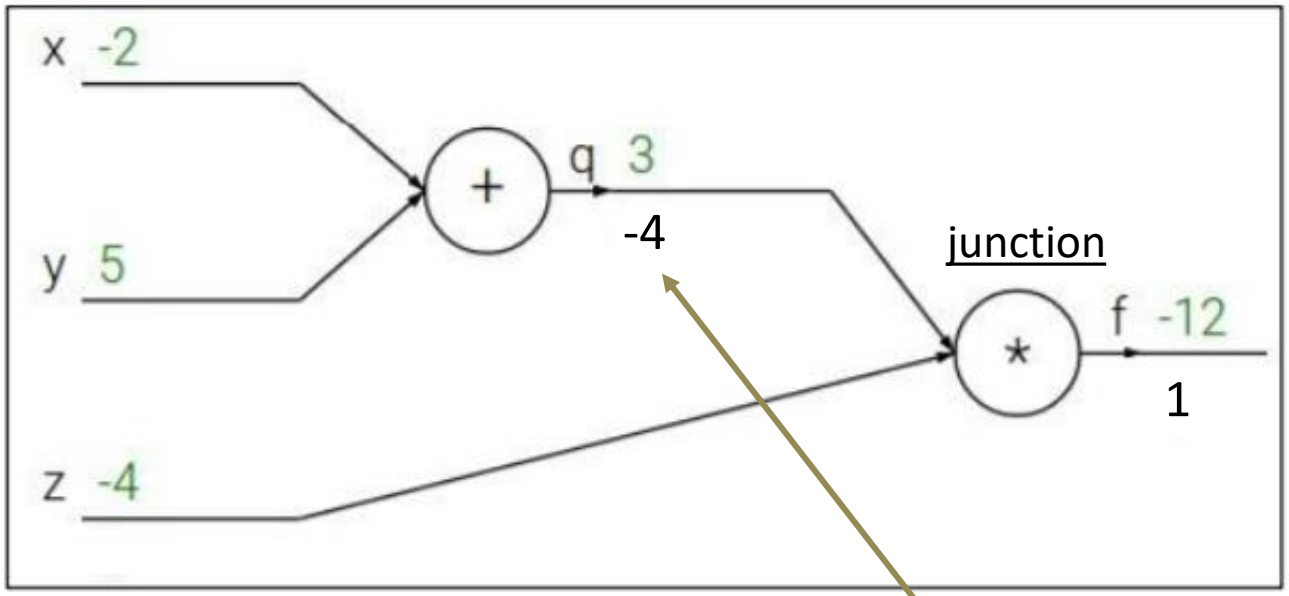
$$\frac{\partial f}{\partial f} = 1$$

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



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

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

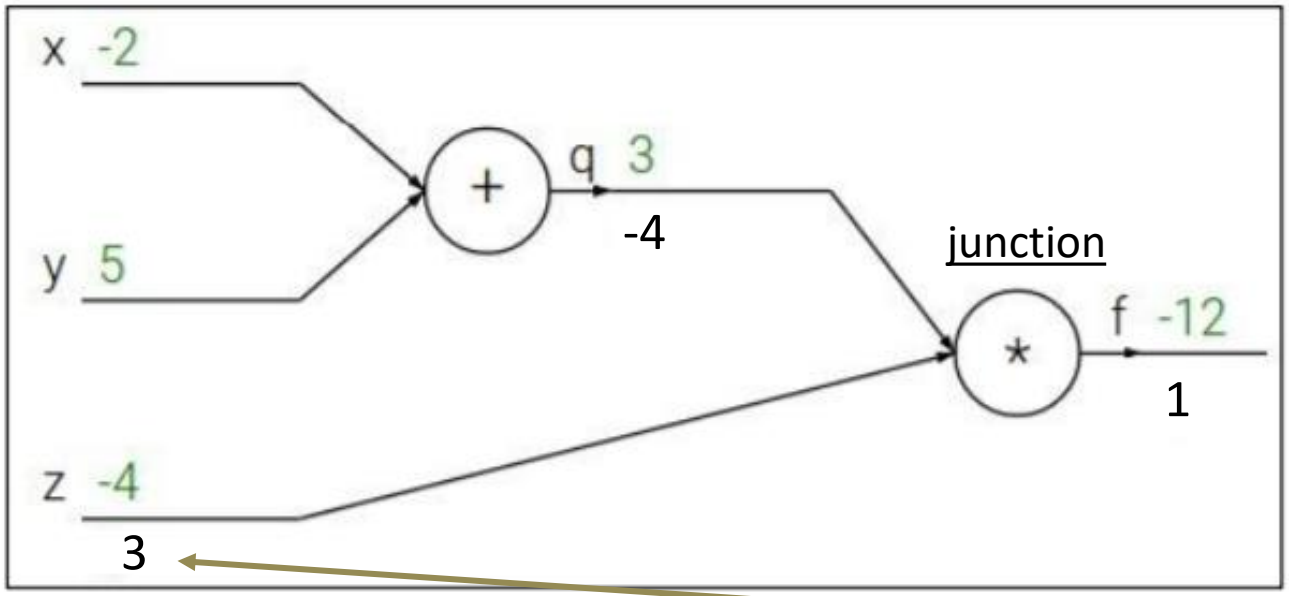
Iterative step: chain rule on locale junction.

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

Backpropagation: a simple example

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

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



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

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

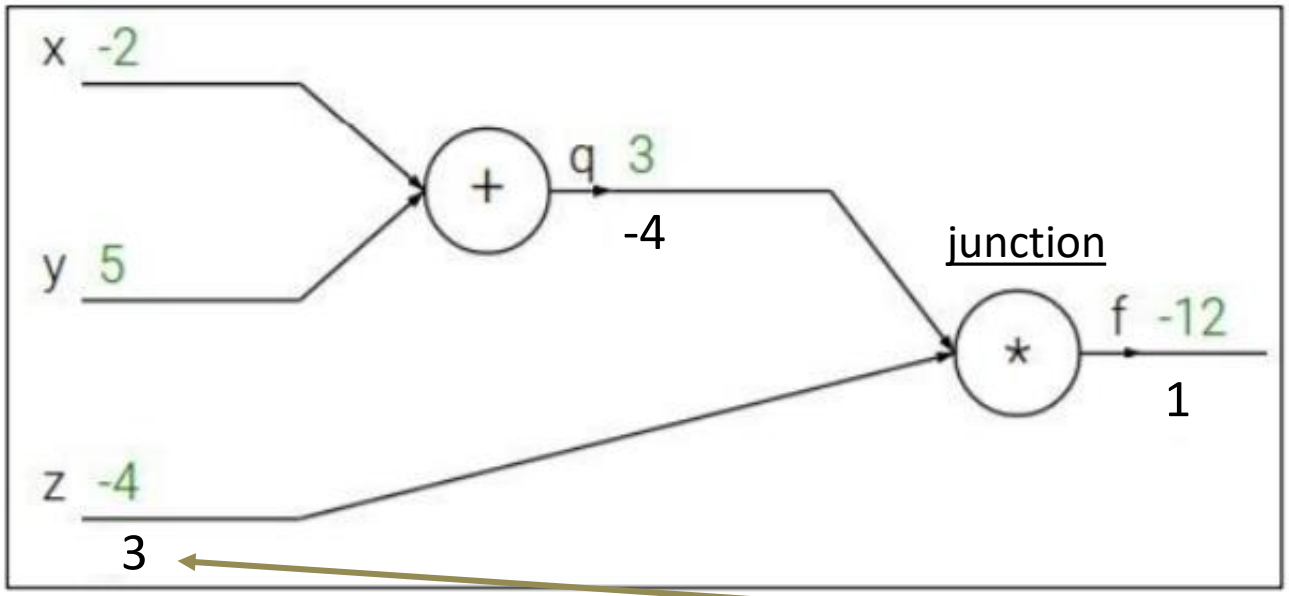
Iterative step: chain rule on locale junction.

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

Backpropagation: a simple example

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

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



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

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

Iterative step: chain rule on locale junction.

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

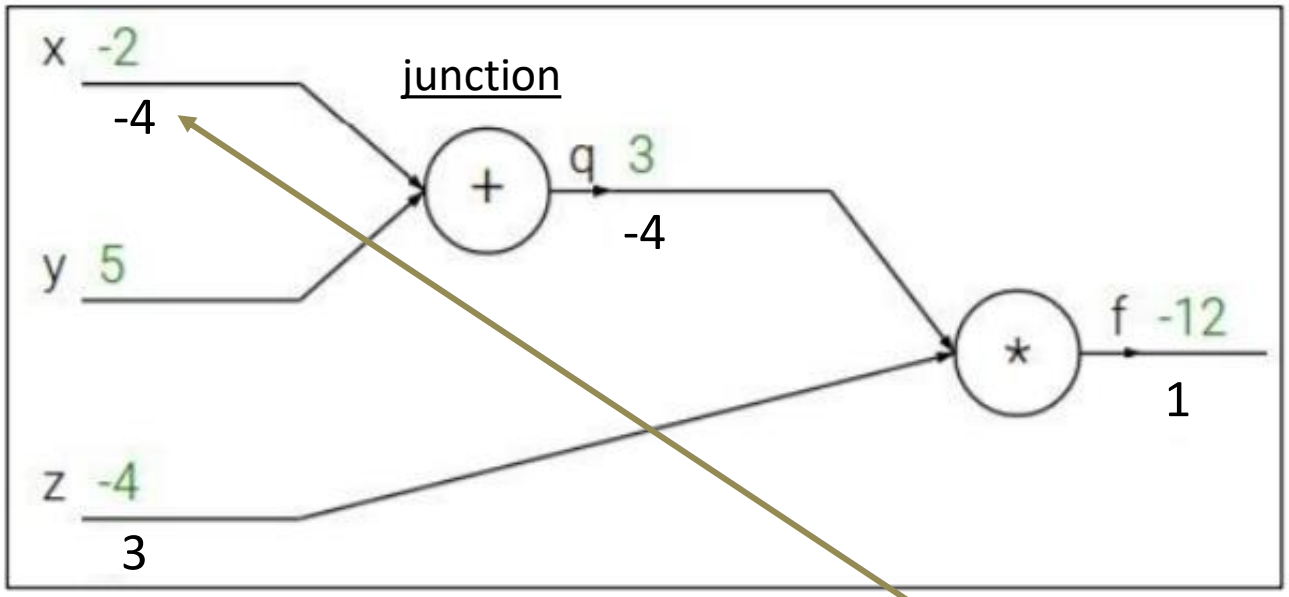
Backpropagation: a simple example

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

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

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

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



Iterative step: chain rule on locale junction.

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

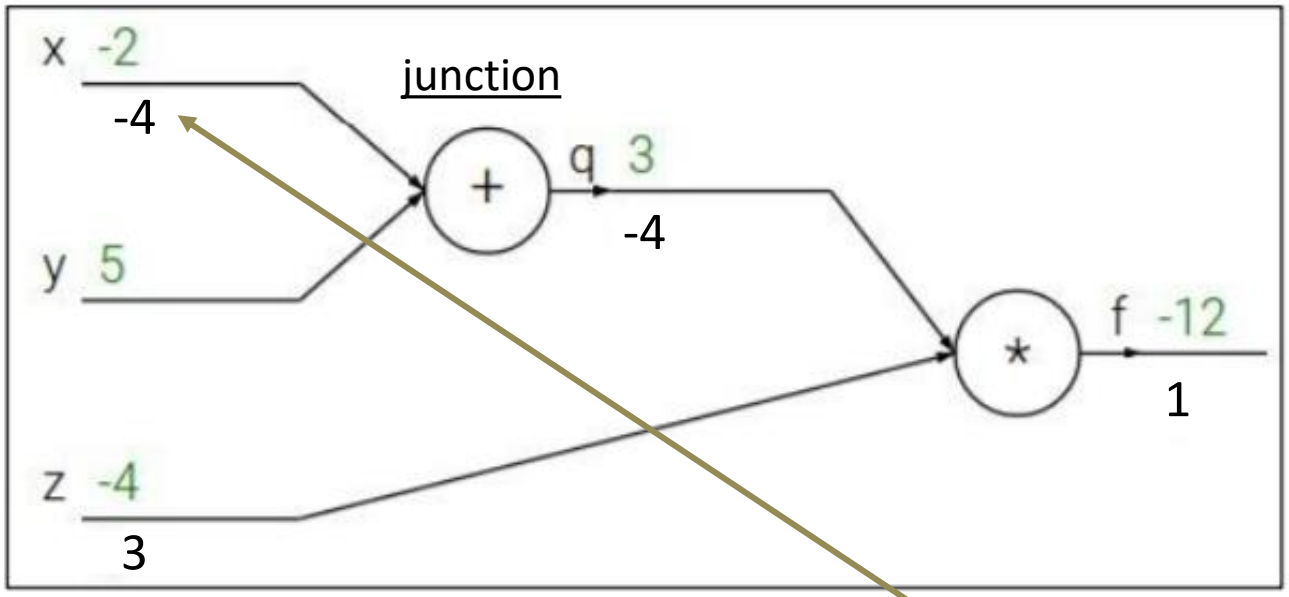
Backpropagation: a simple example

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

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

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

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



Iterative step: chain rule on locale junction.

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

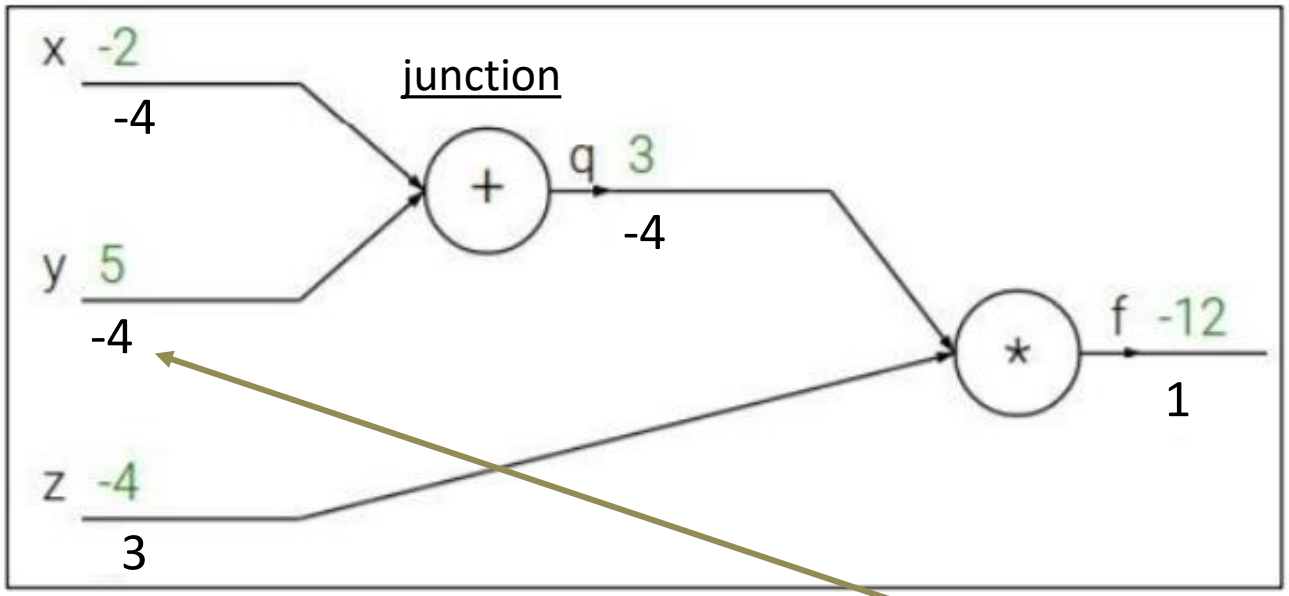
Backpropagation: a simple example

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

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

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

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



Iterative step: chain rule on locale junction.

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

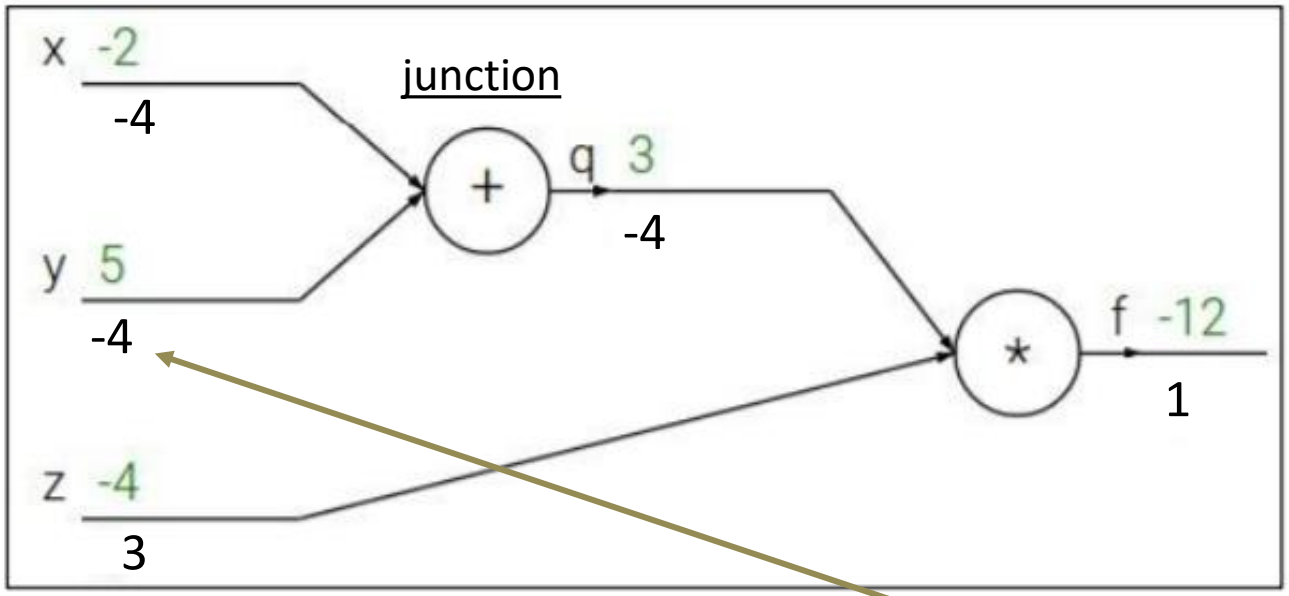
Backpropagation: a simple example

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

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

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

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

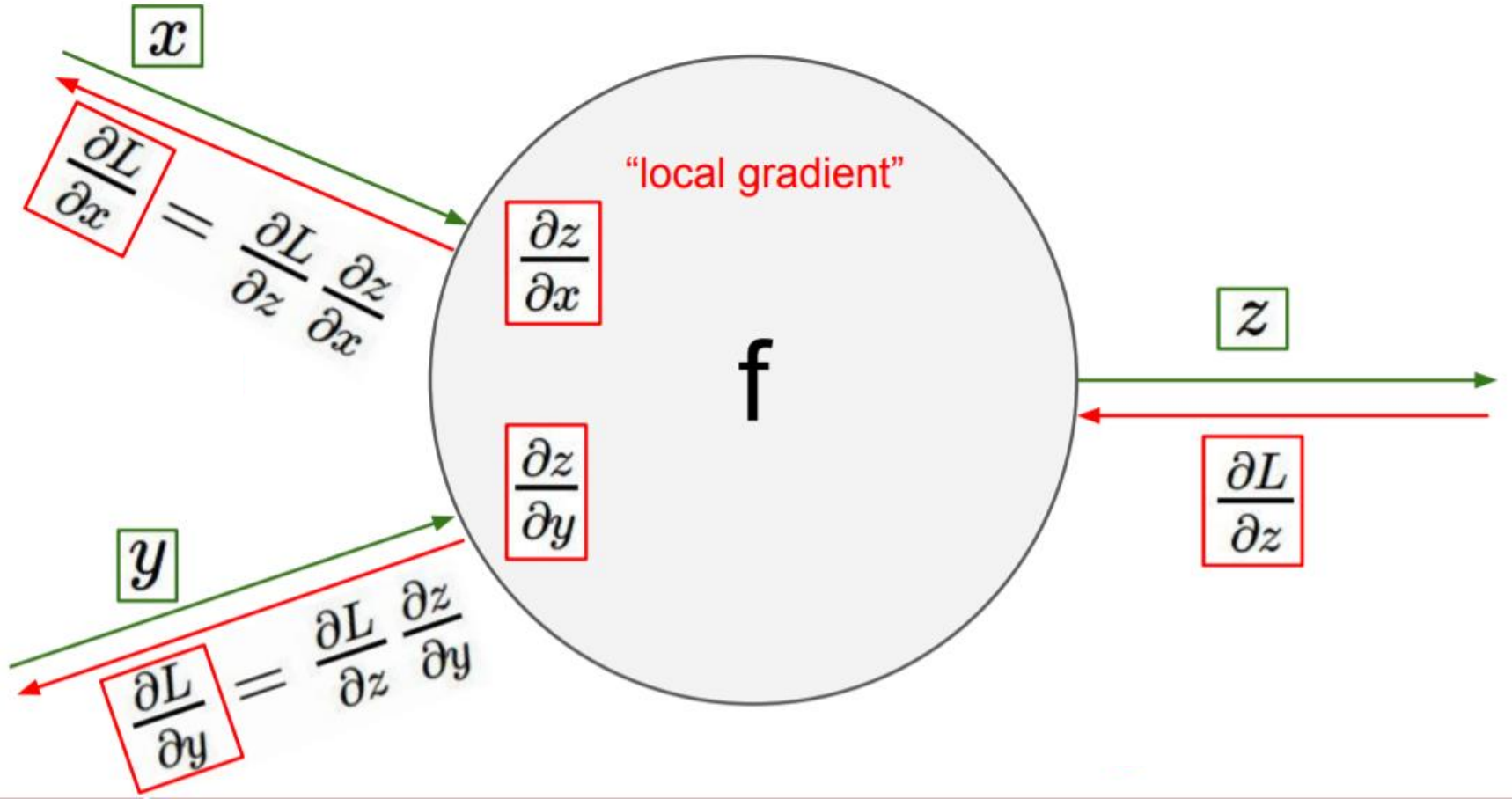


Iterative step: chain rule on locale junction.

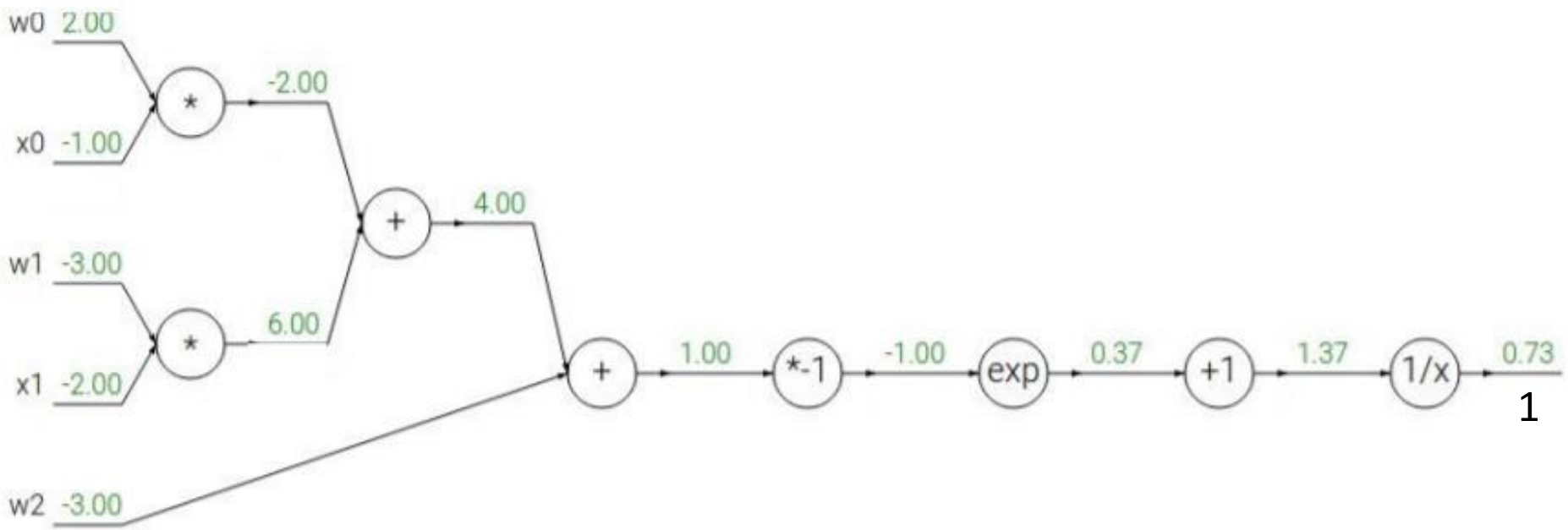
$$\frac{\partial f}{\partial y} = \frac{\partial f}{\partial q} \cdot \frac{\partial q}{\partial y} = -4 \cdot 1 = -4$$

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.



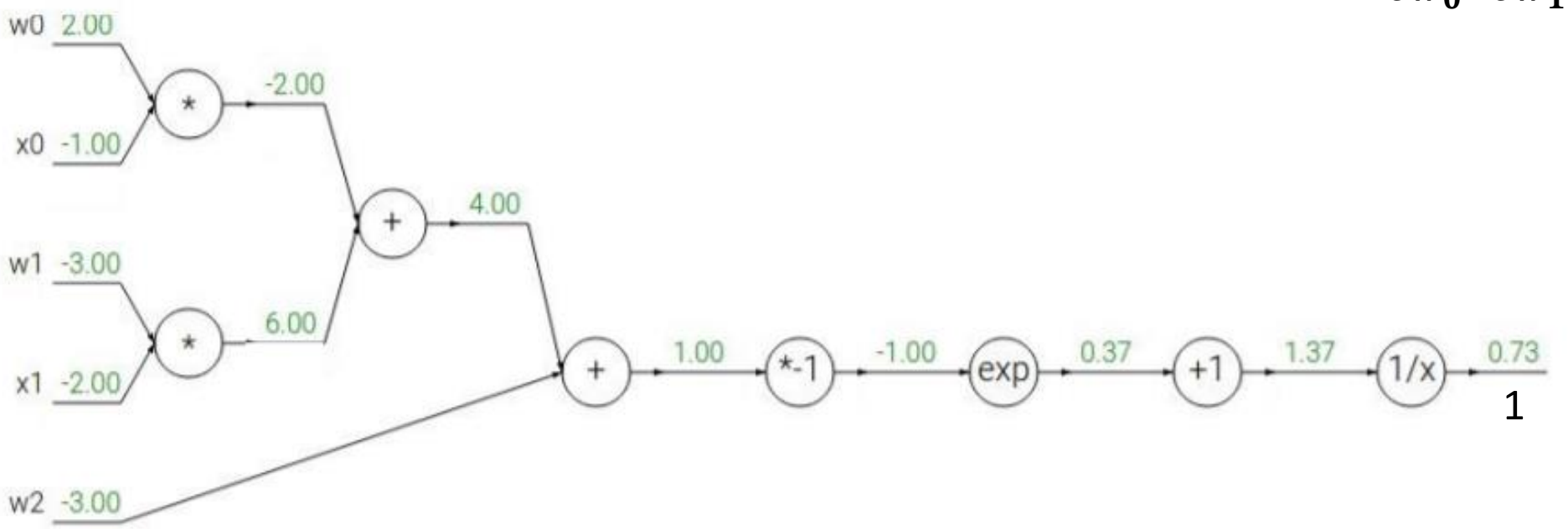
Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$



1

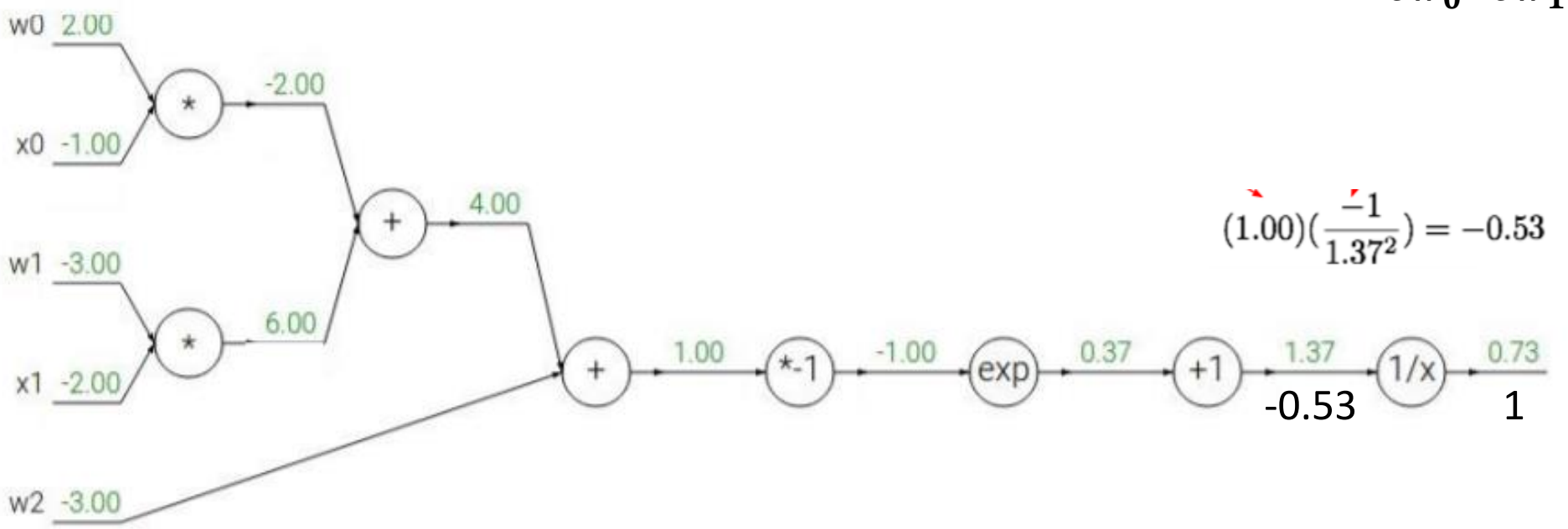
Another example: $$f(w,x) = \frac{1}{1+e^{-(w_0x_0+w_1x_1+w_2)}}$$

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



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

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

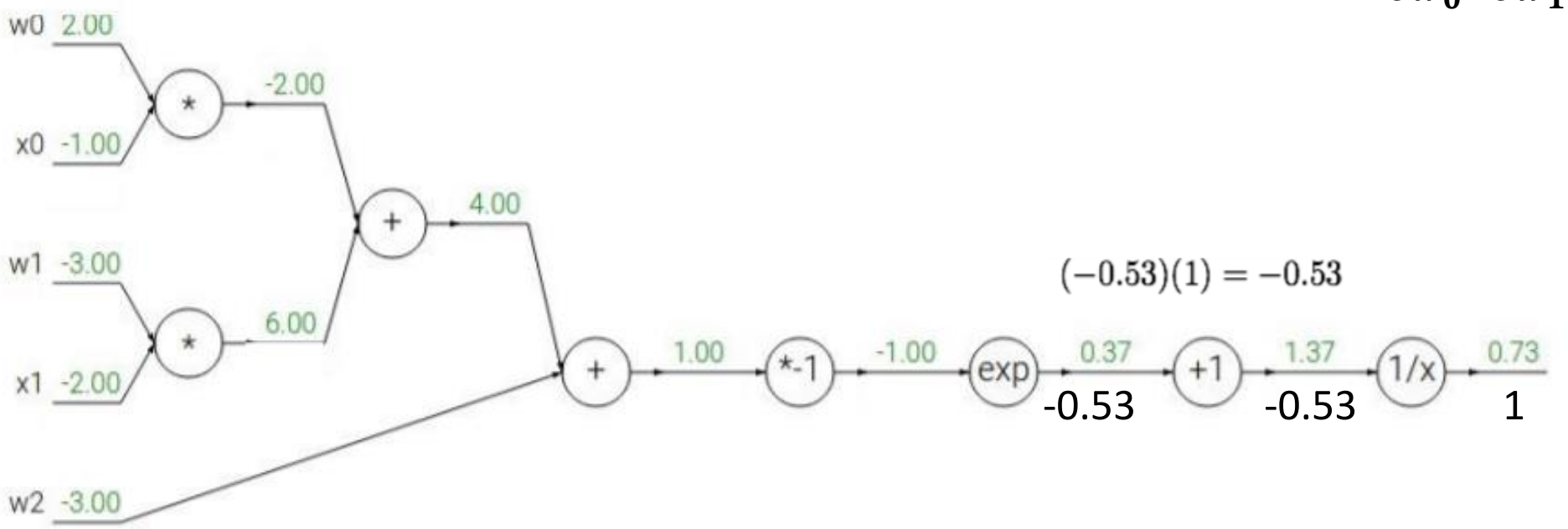


$$(1.00)(\frac{-1}{1.37^2}) = -0.53$$

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

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

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

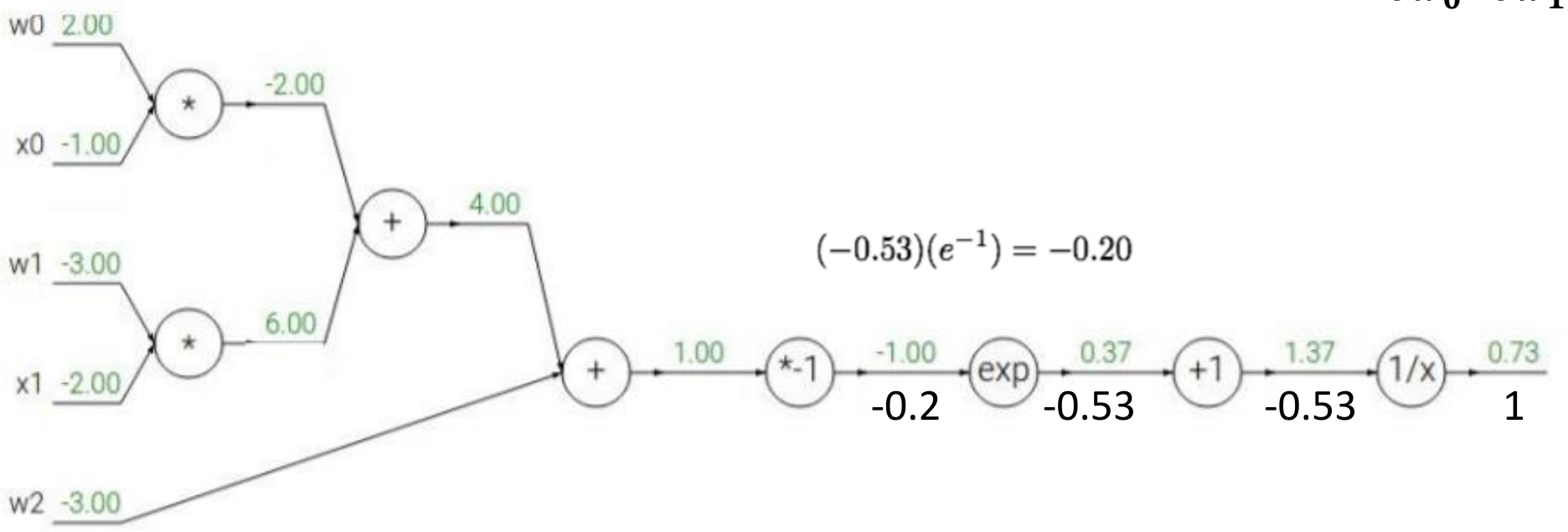


$$(-0.53)(1) = -0.53$$

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

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

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

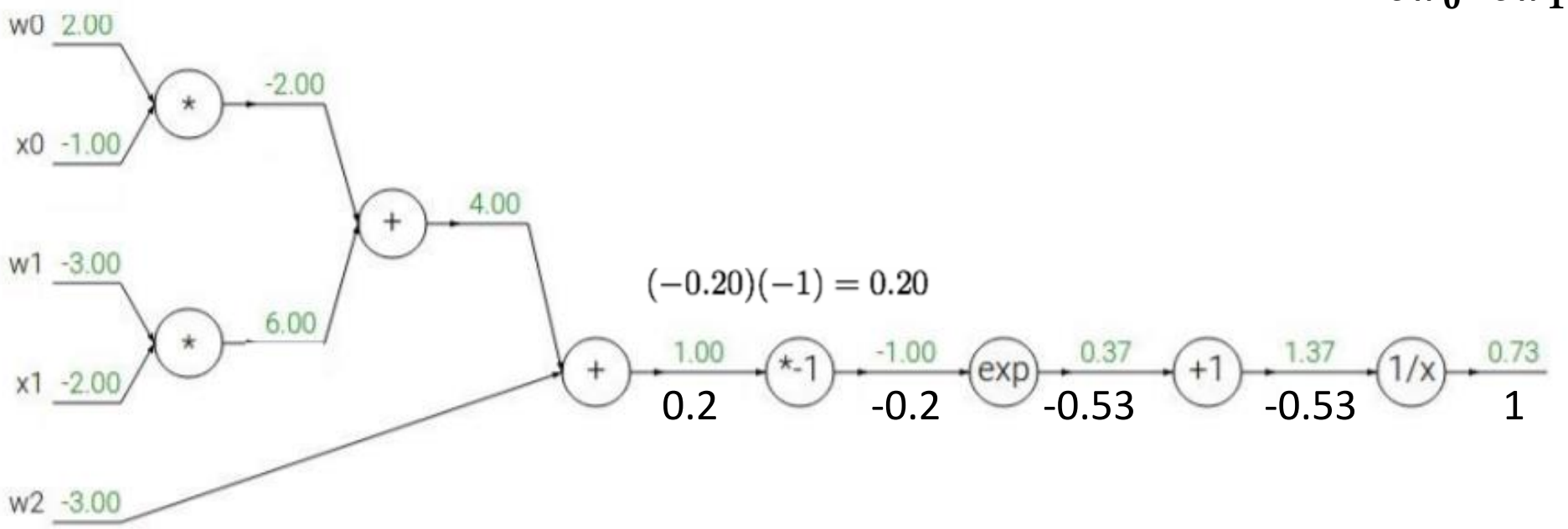


$$(-0.53)(e^{-1}) = -0.20$$

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

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

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

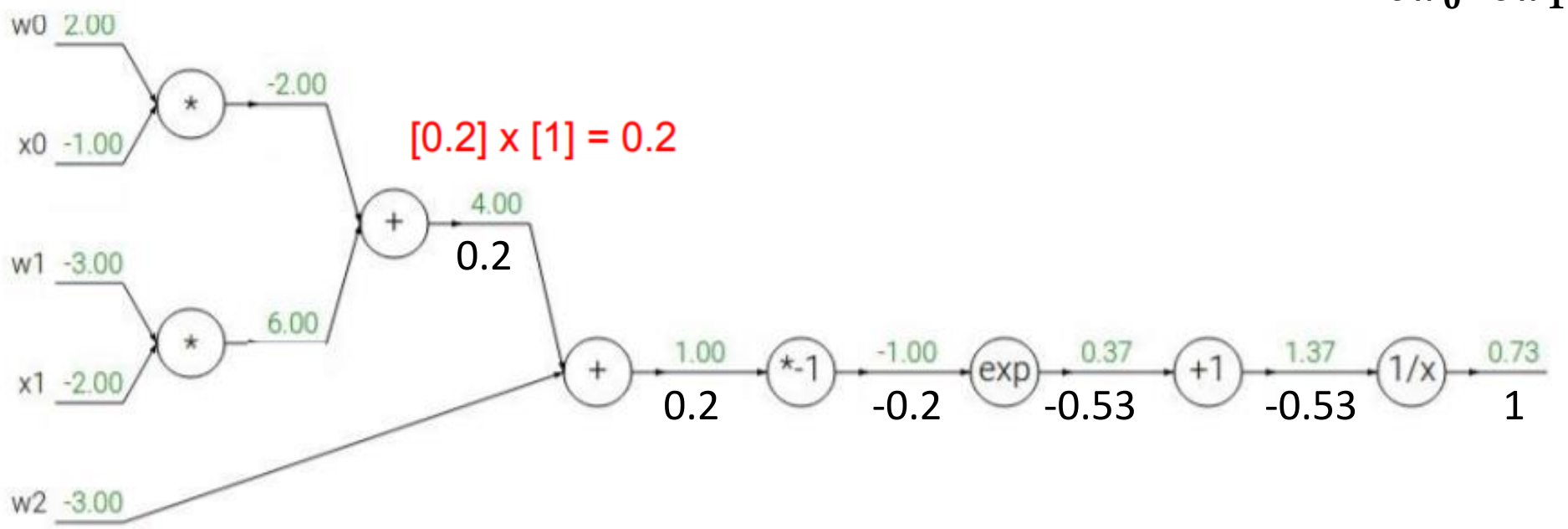


$(-0.20)(-1) = 0.20$

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

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

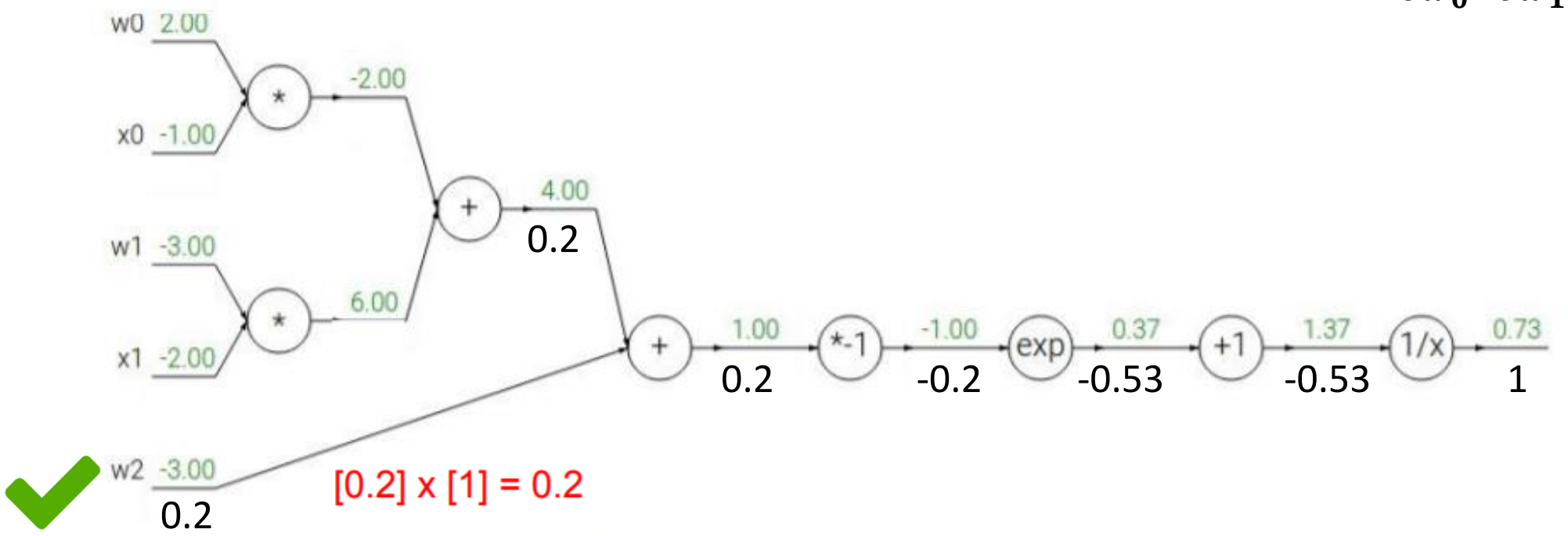
We want to find:
$\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$



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

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

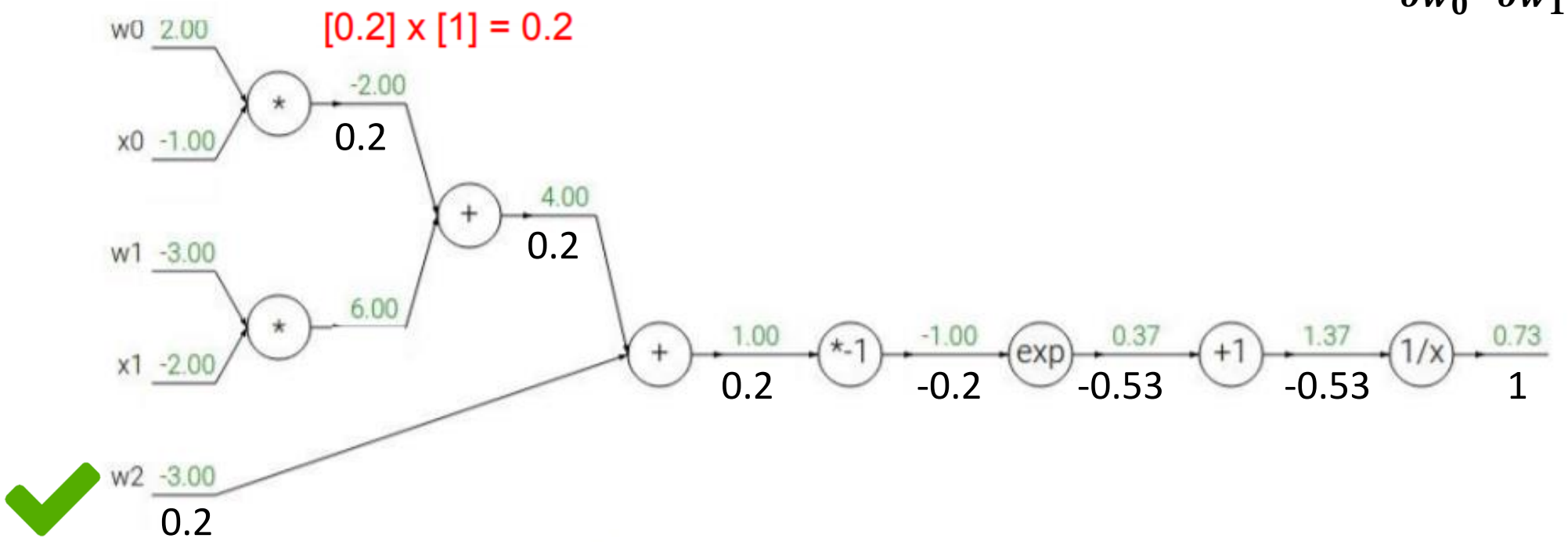
We want to find:
$\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$



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

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

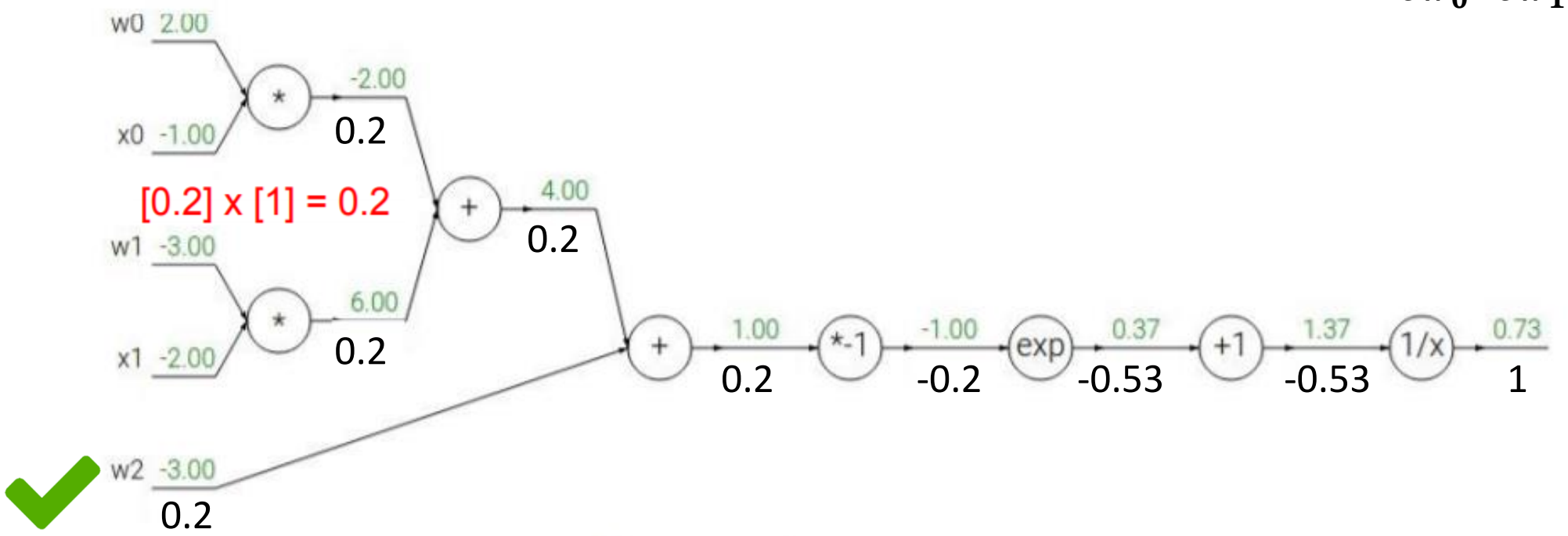
We want to find:
$\frac{\partial f}{\partial w_0}, \frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}$



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

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

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



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

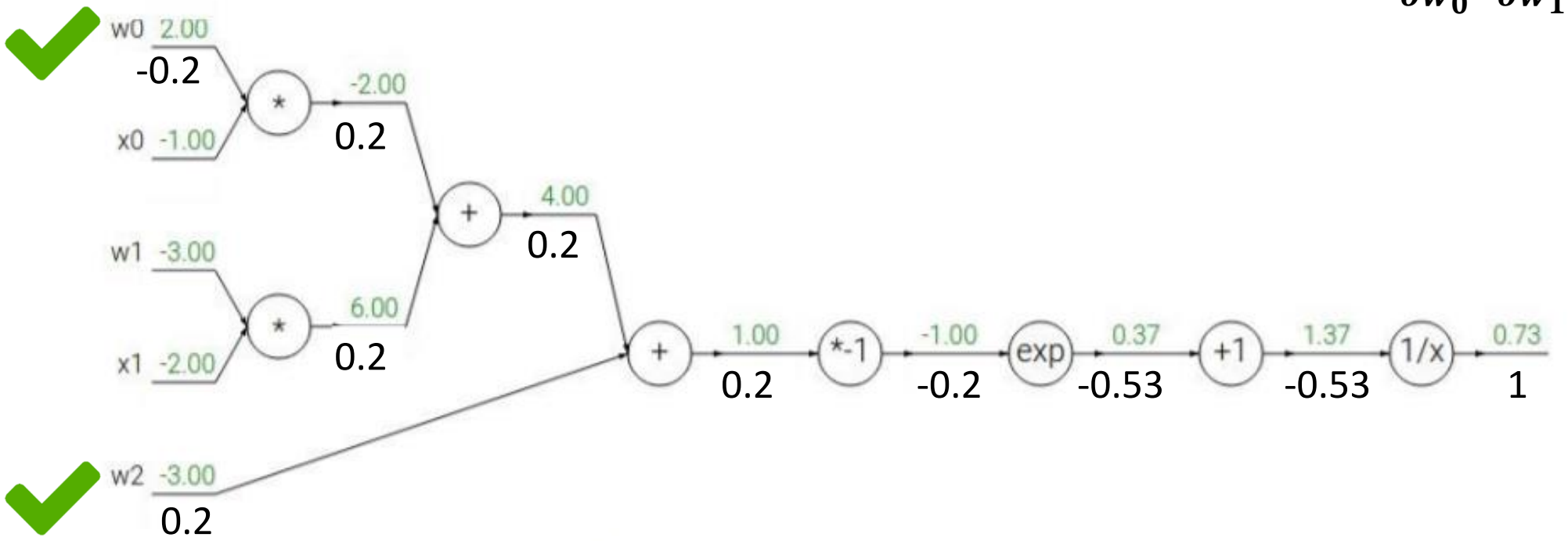
Another example:

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

We want to find:

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

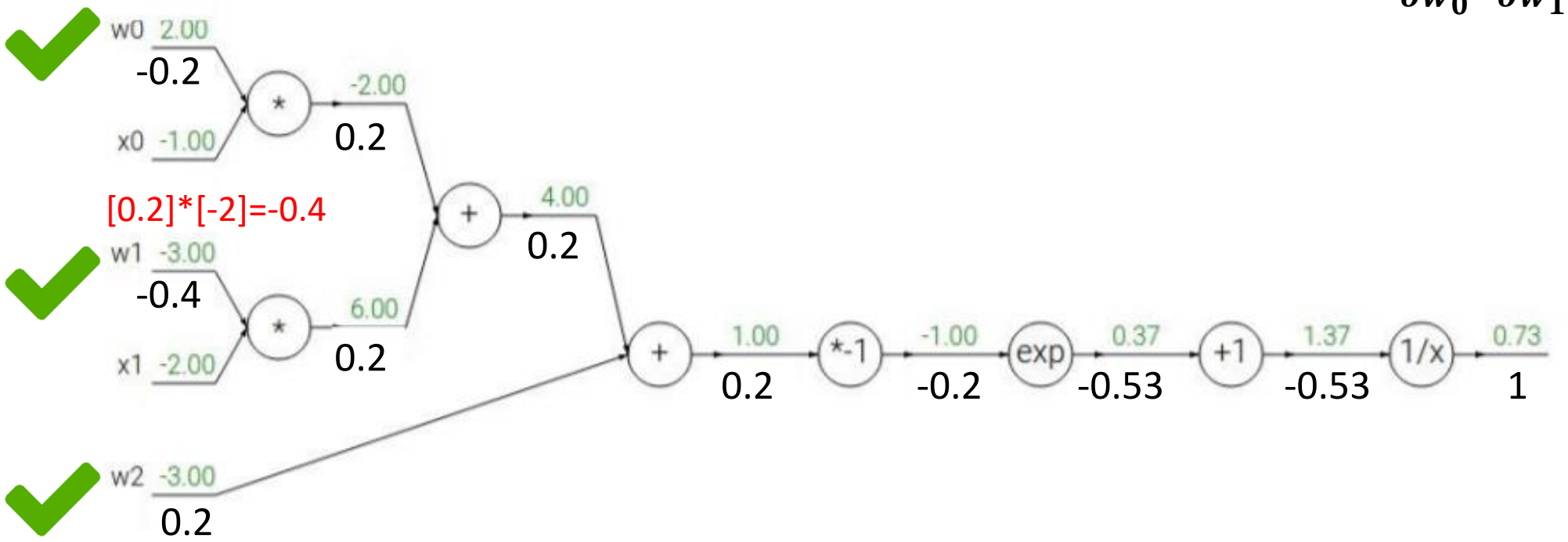
[0.2] x [-1] = -0.2



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

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

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



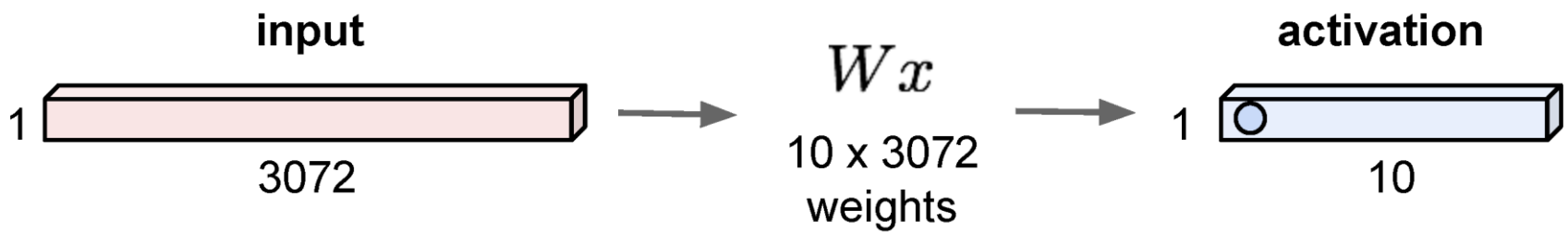
$$f_a(x) = ax \quad \rightarrow \quad \frac{df}{dx} = a$$

contents

- Chain rule
- **ConvNets**
  - **Convolution layer**
  - Pooling layer
- Overfitting
- Architectures
  - Alexnet (dropout)
  - VGG
  - ResNet (batch norm)

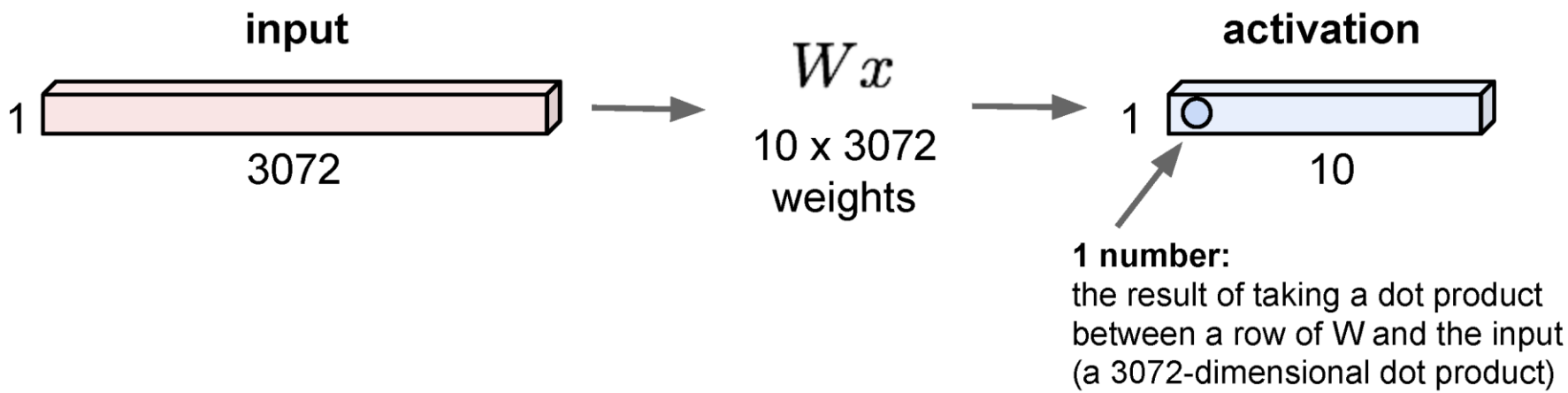
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



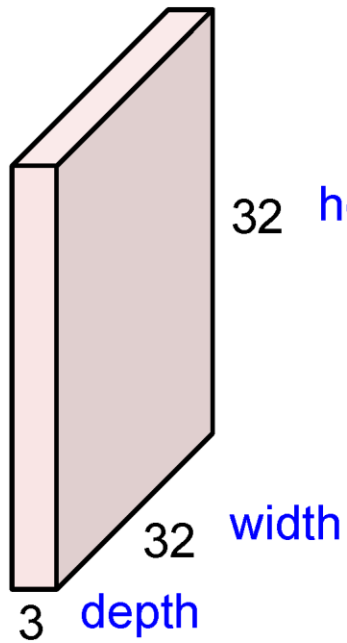
Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



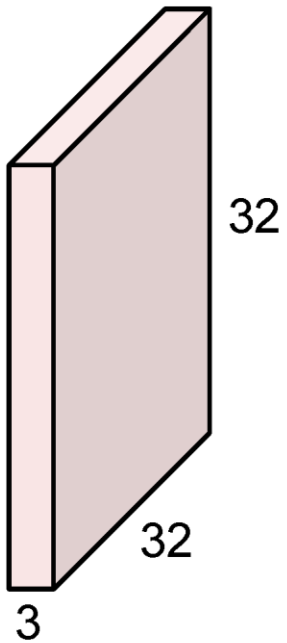
Convolution Layer

32x32x3 image -> preserve spatial structure

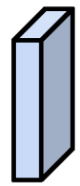


Convolution Layer

32x32x3 image



5x5x3 filter

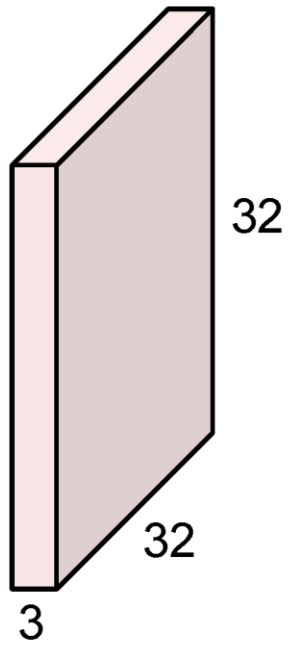


Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

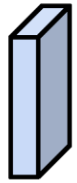
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



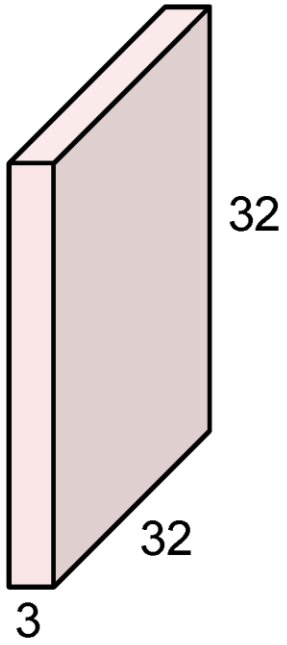
5x5x3 filter



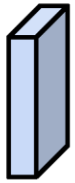
Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Convolution Layer

32x32x3 image



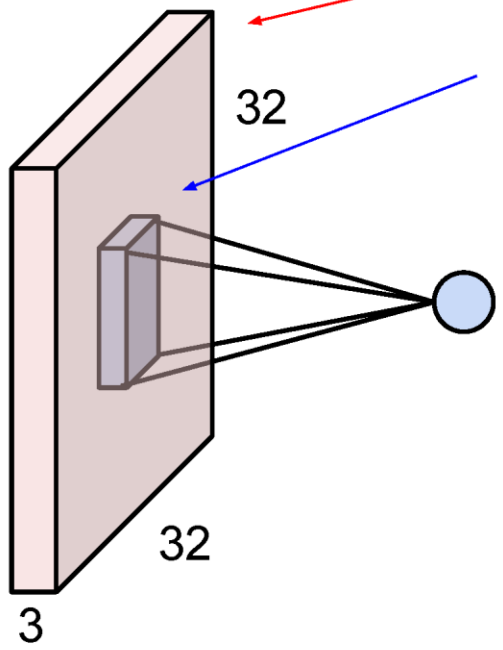
5x5x3 filter



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



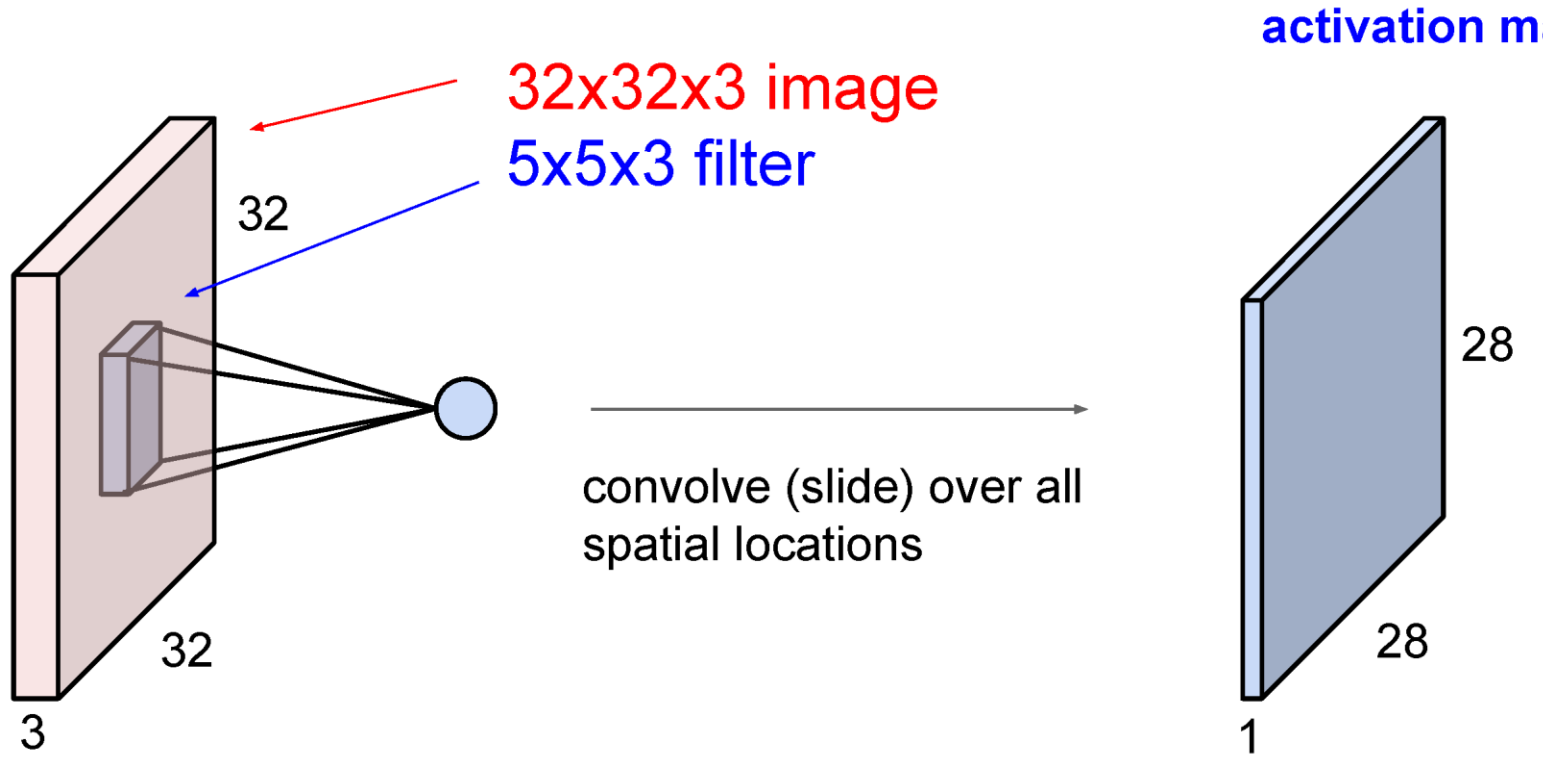
32x32x3 image

5x5x3 filter w

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)

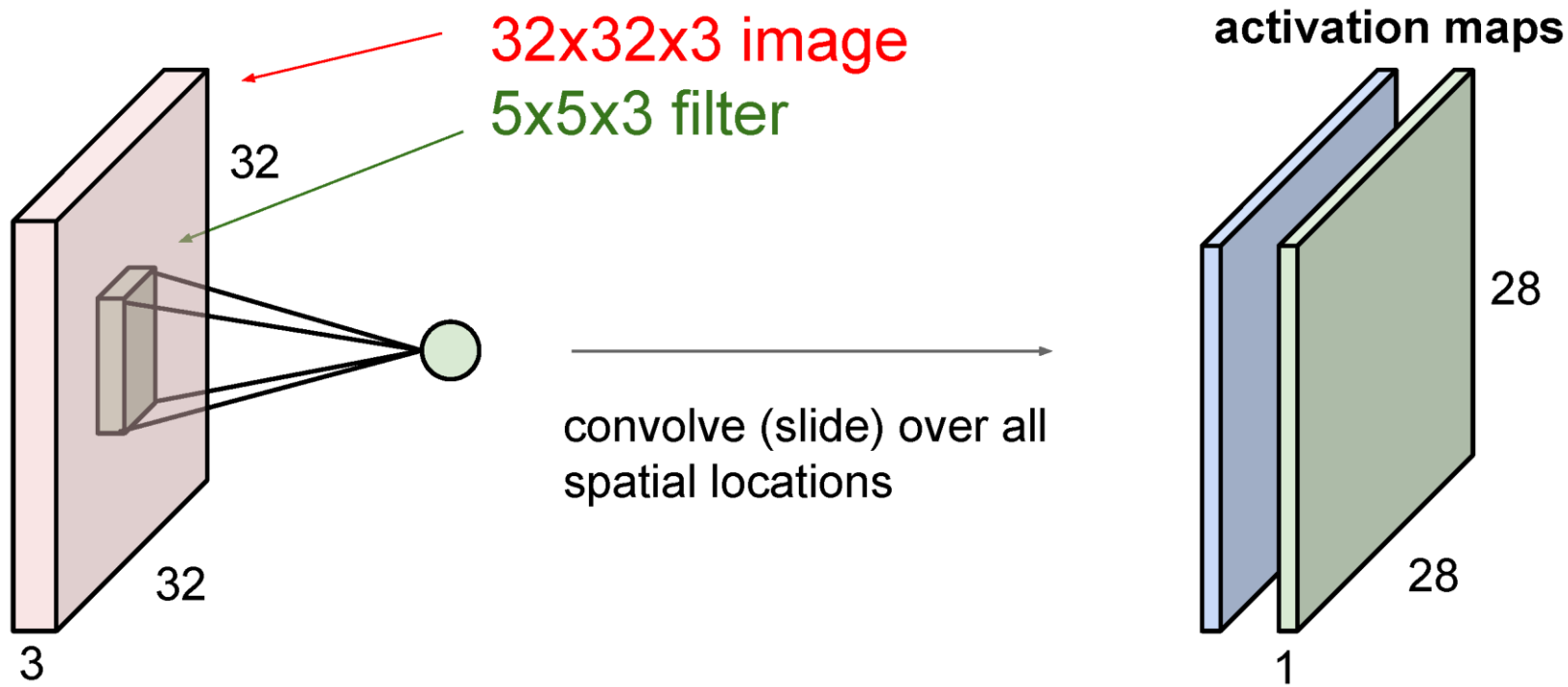
$$w^T x + b$$

Convolution Layer

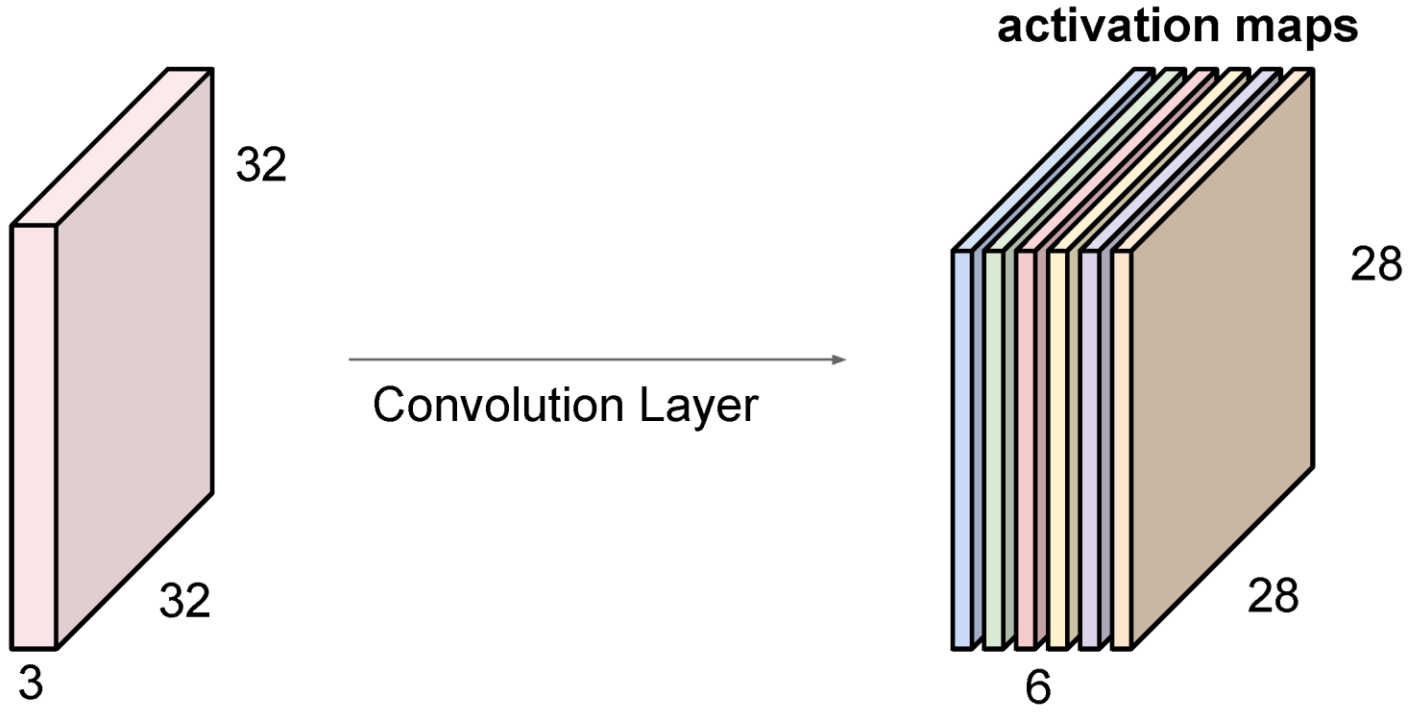


Convolution Layer

consider a second, green filter



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

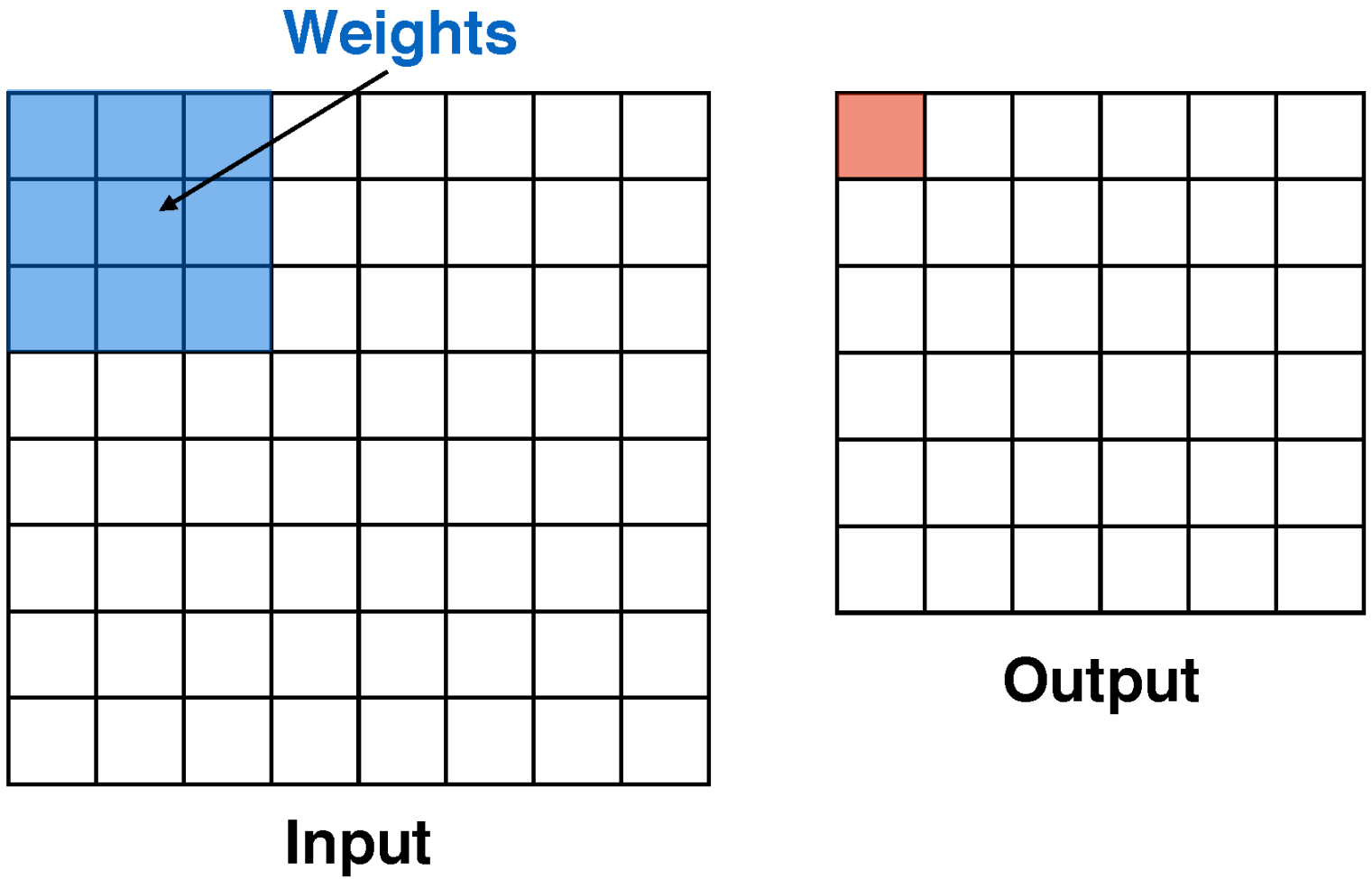


We stack these up to get a "new image" of size 28x28x6!

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

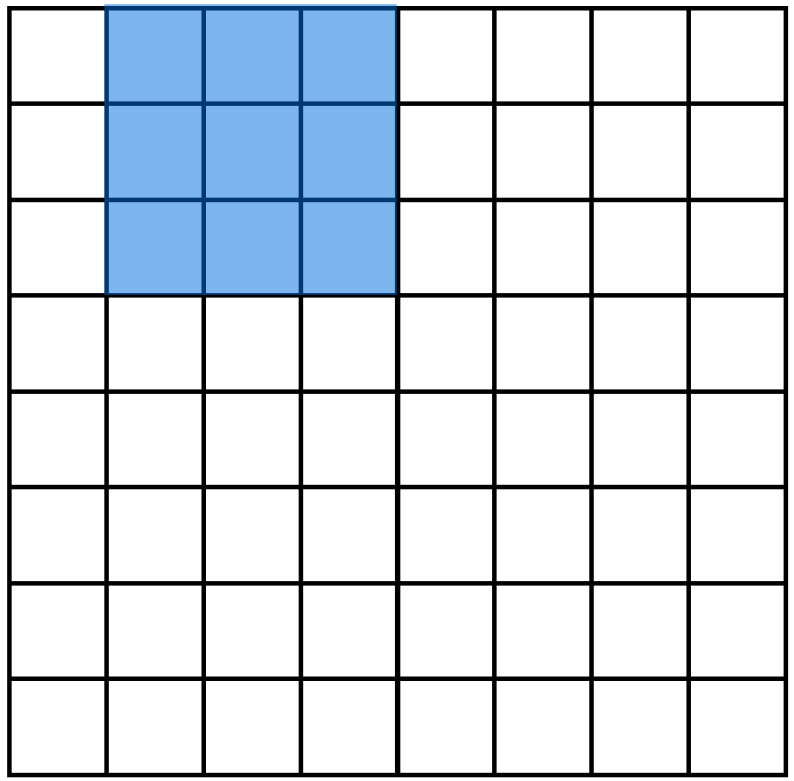
Convolution: Stride

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

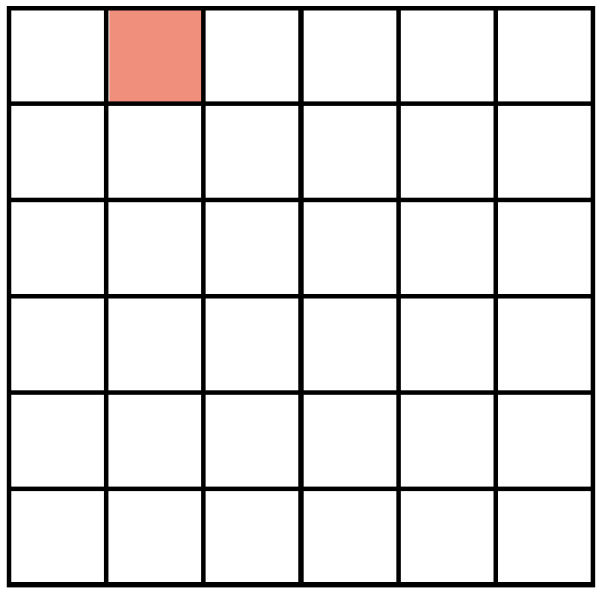


Convolution: Stride

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



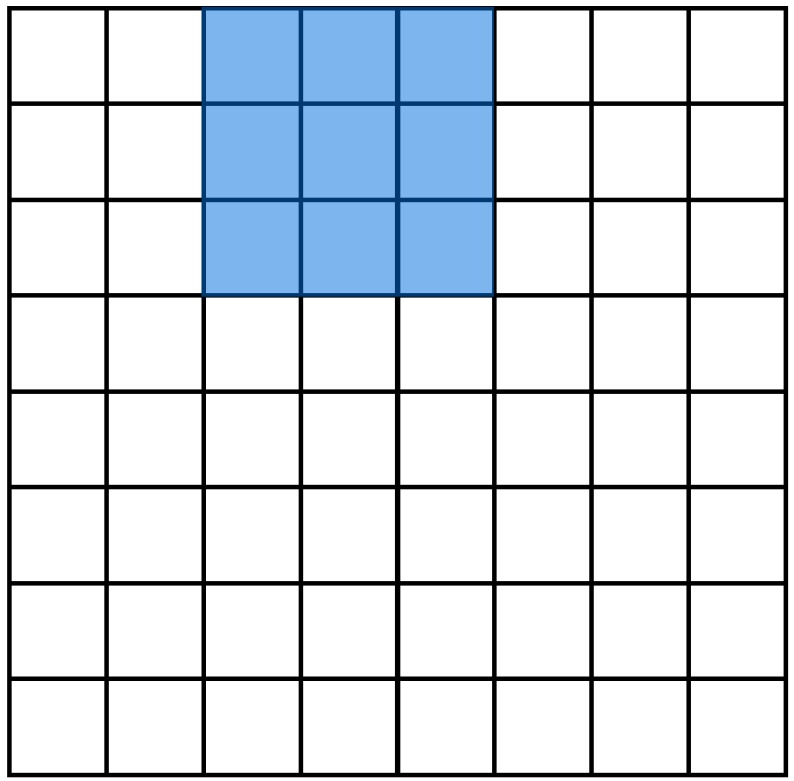
Input



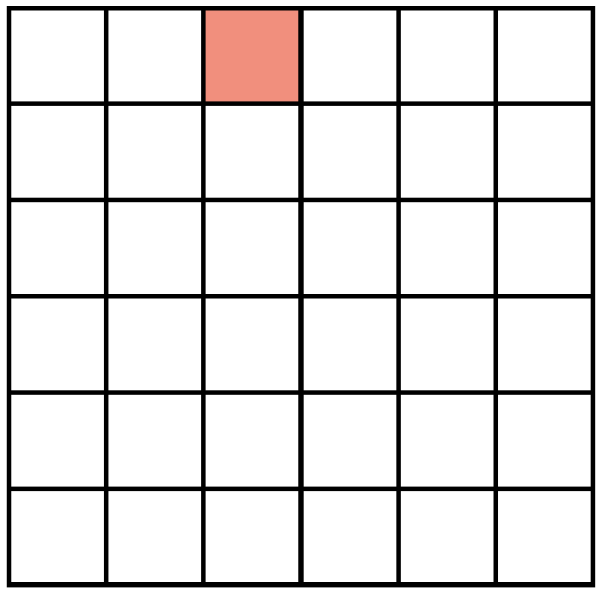
Output

Convolution: Stride

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



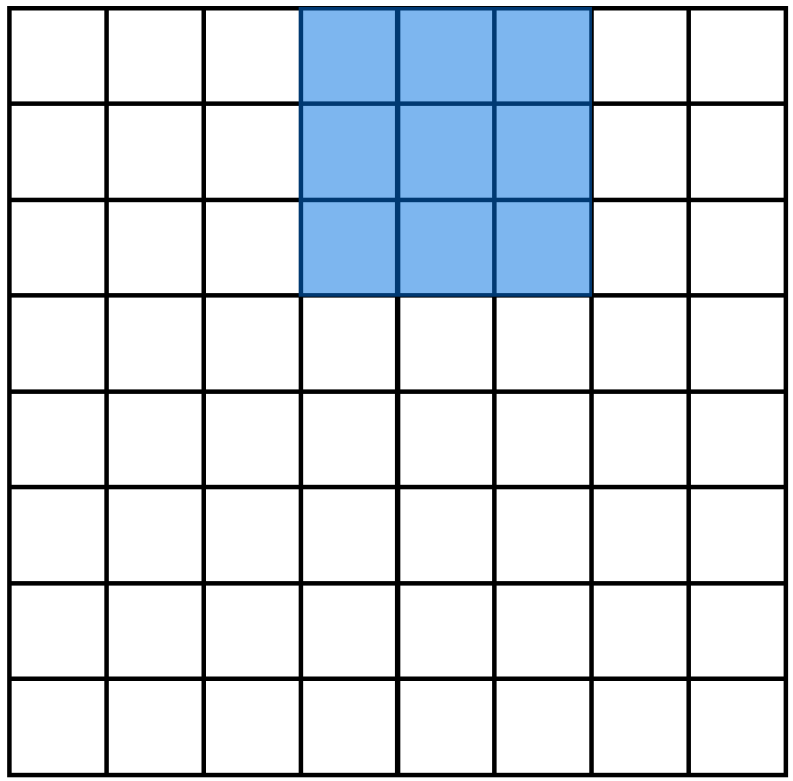
Input



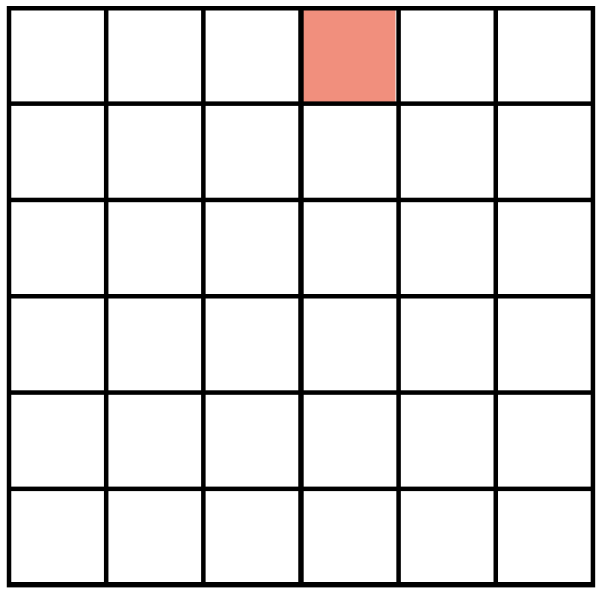
Output

Convolution: Stride

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



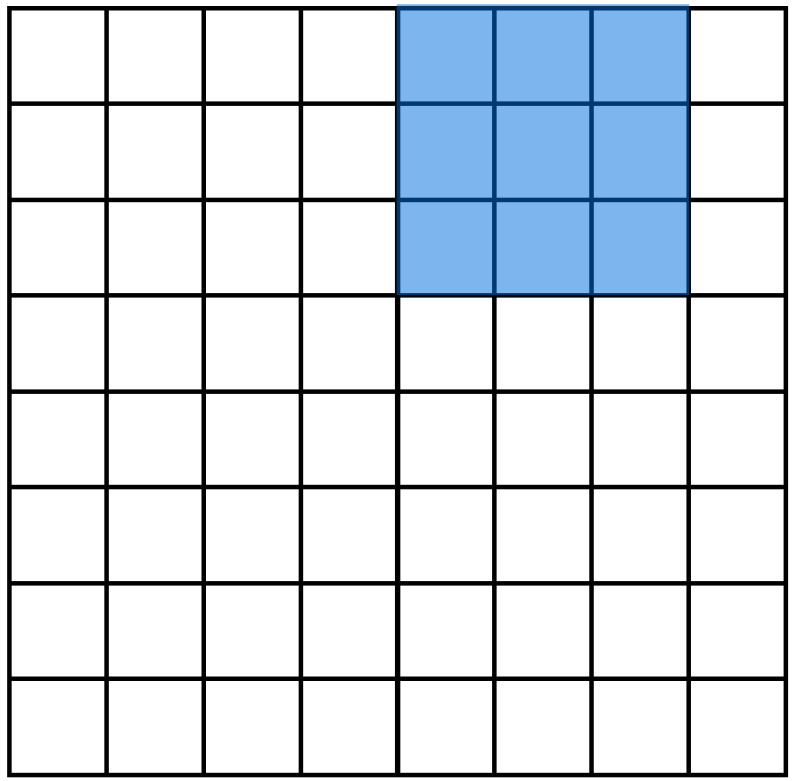
Input



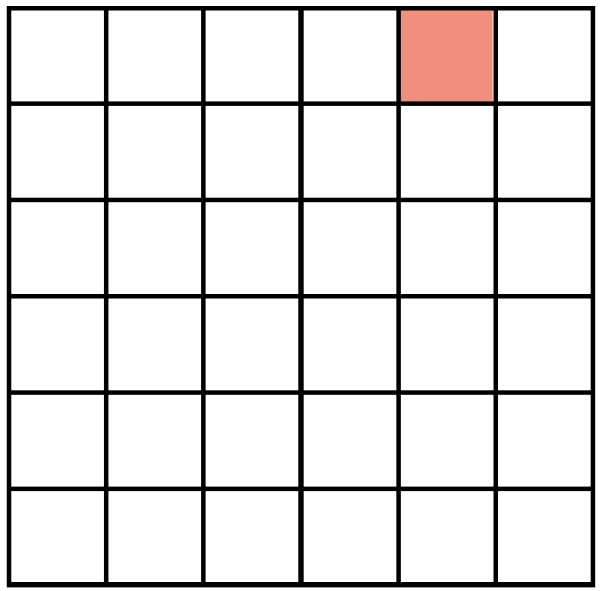
Output

Convolution: Stride

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



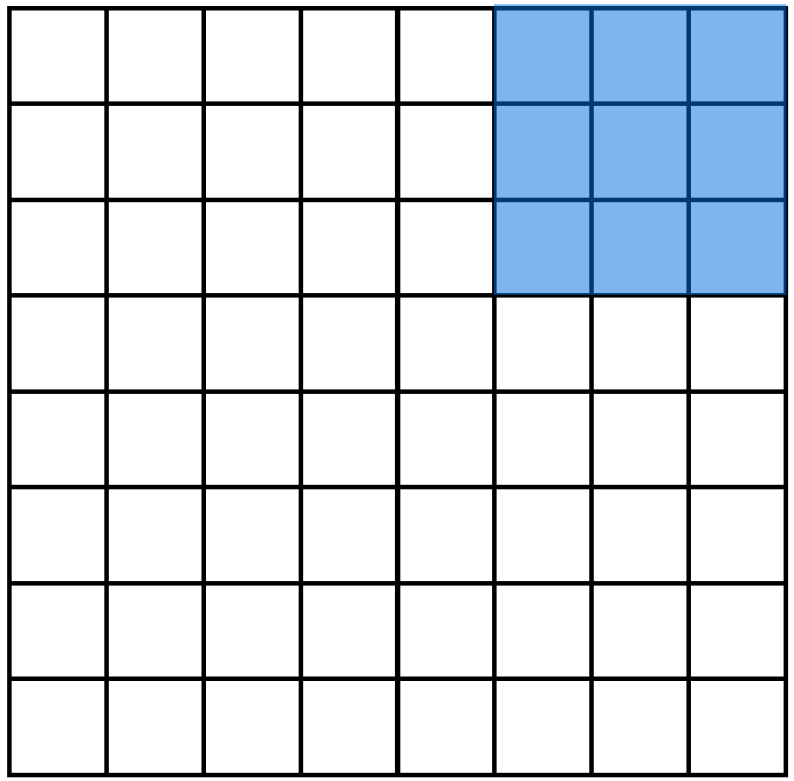
Input



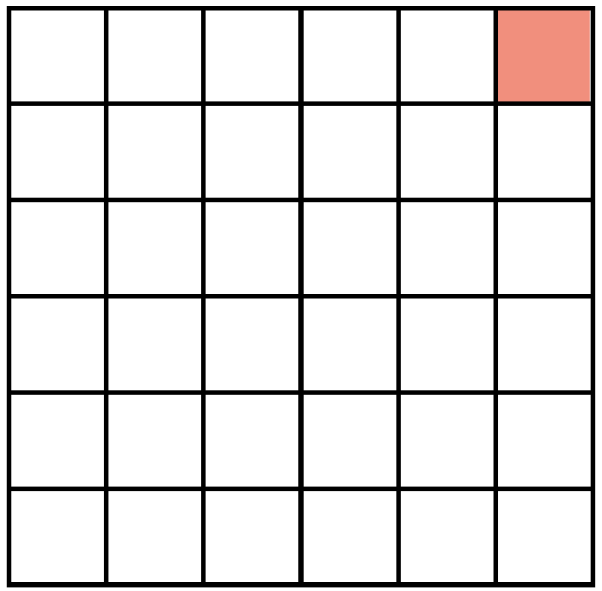
Output

Convolution: Stride

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



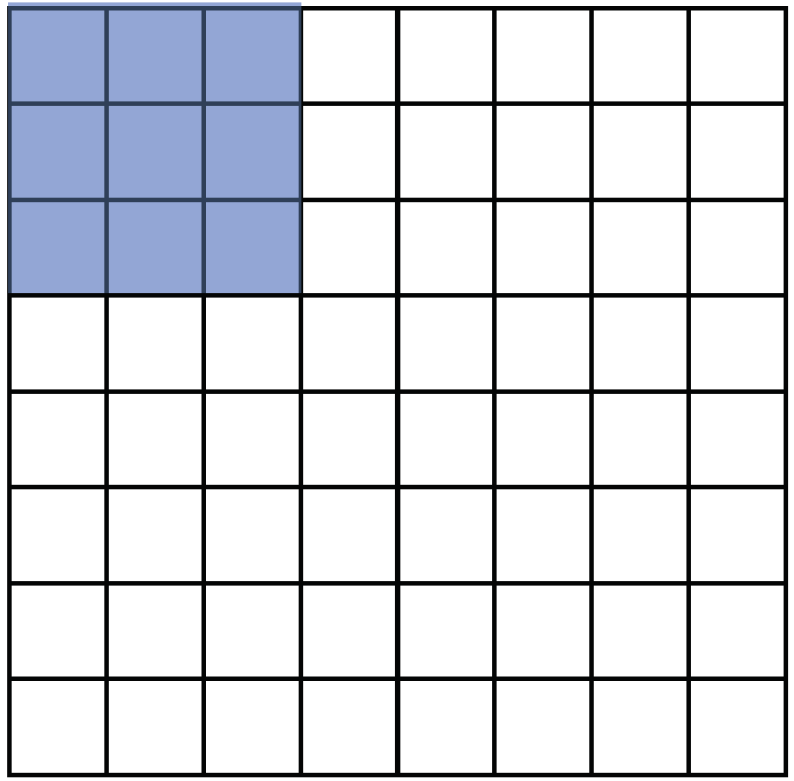
Input



Output

Convolution: Stride

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



Input

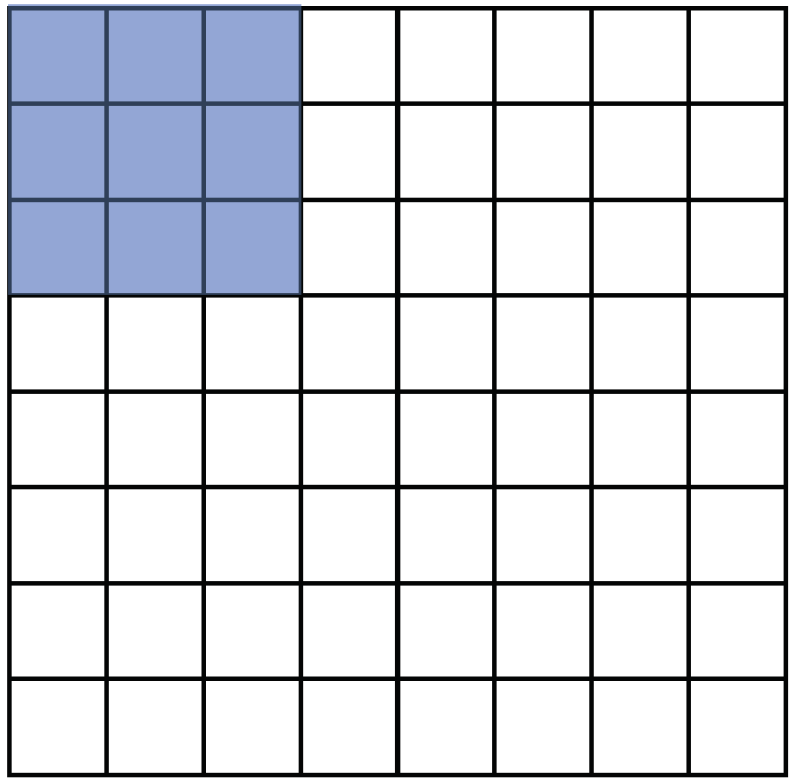
Recall that at each position, we are doing a **3D** sum:

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

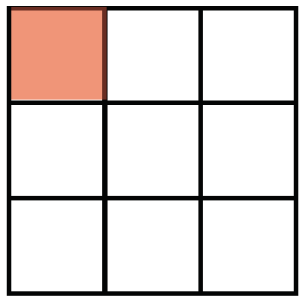
(channel, row, column)

Convolution: Stride

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



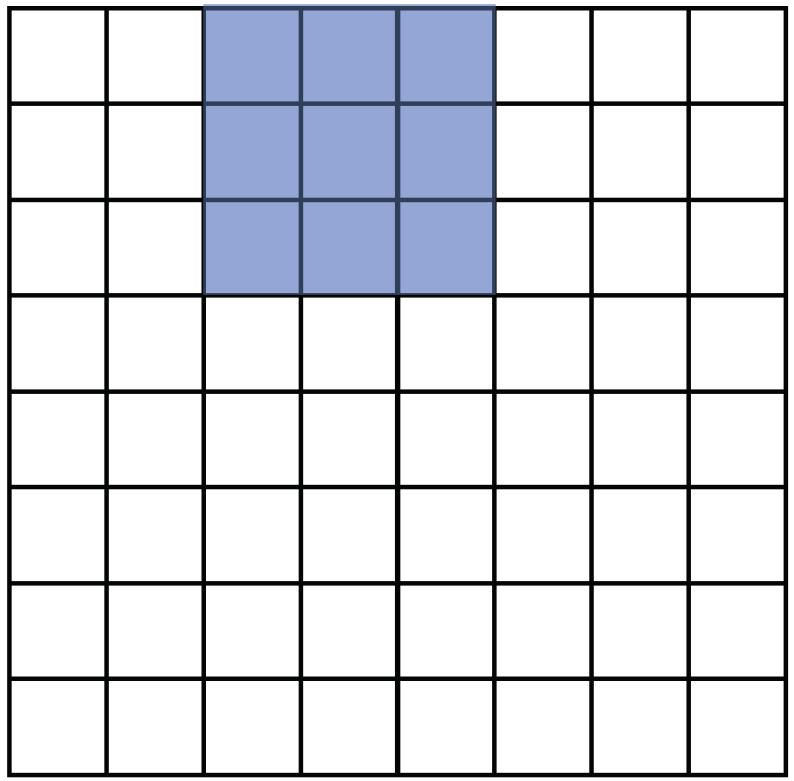
Input



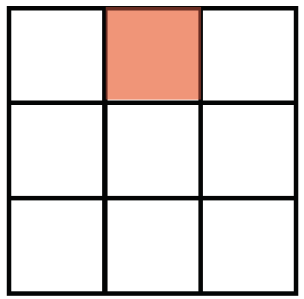
Output

Convolution: Stride

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



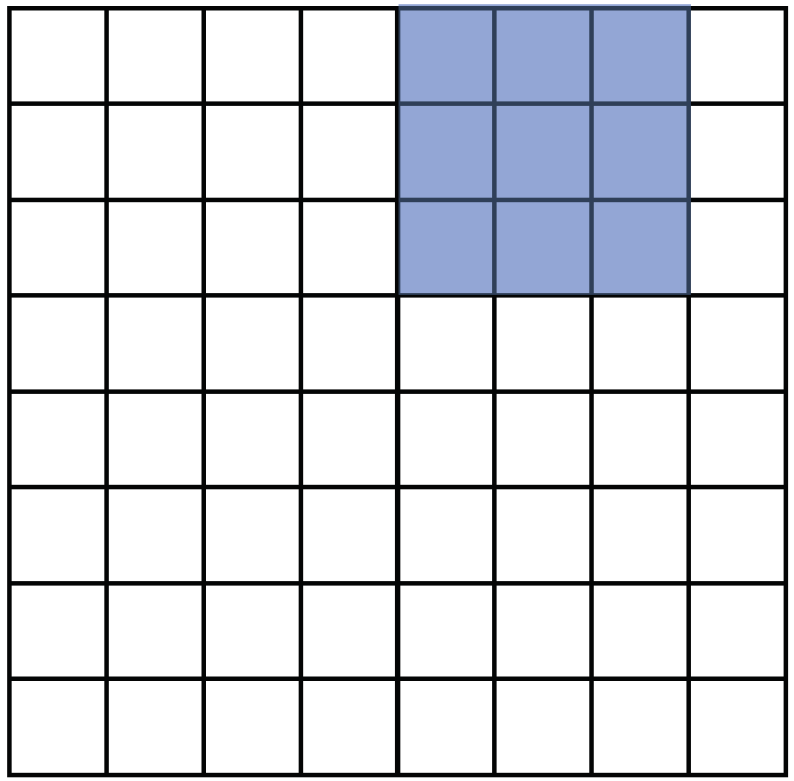
Input



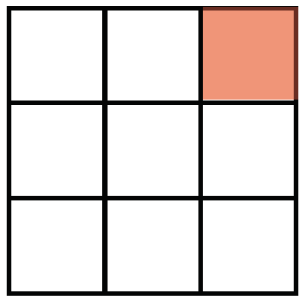
Output

Convolution: Stride

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



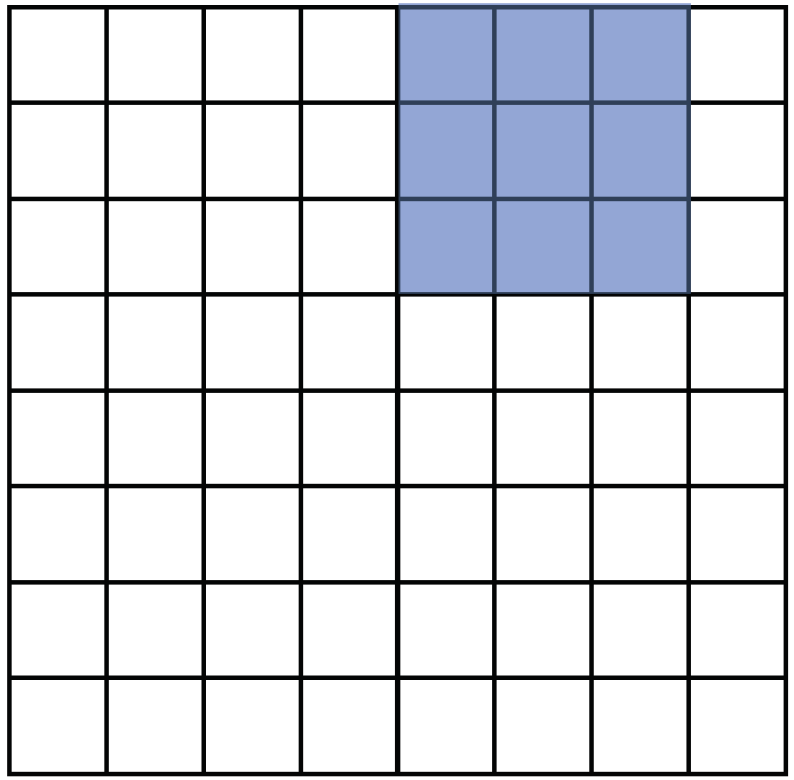
Input



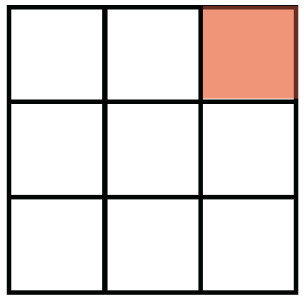
Output

Convolution: Stride

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



Input



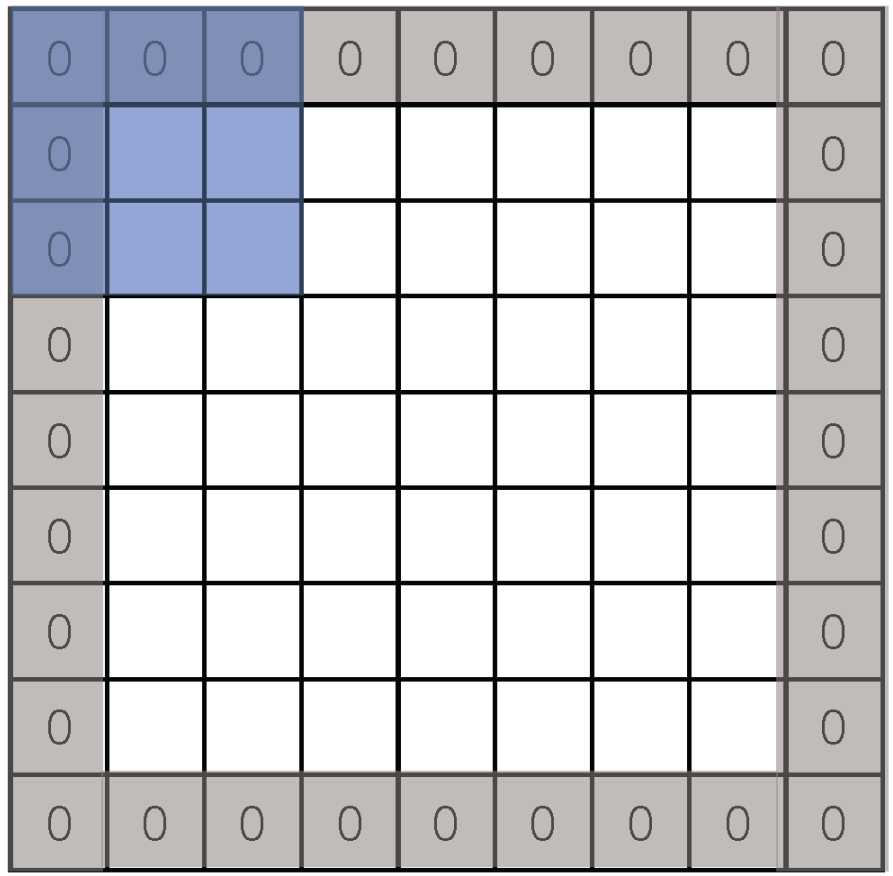
Output

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

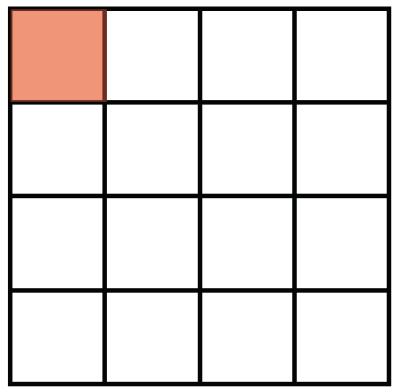
- The output is also half the size of the input

Convolution: Padding

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



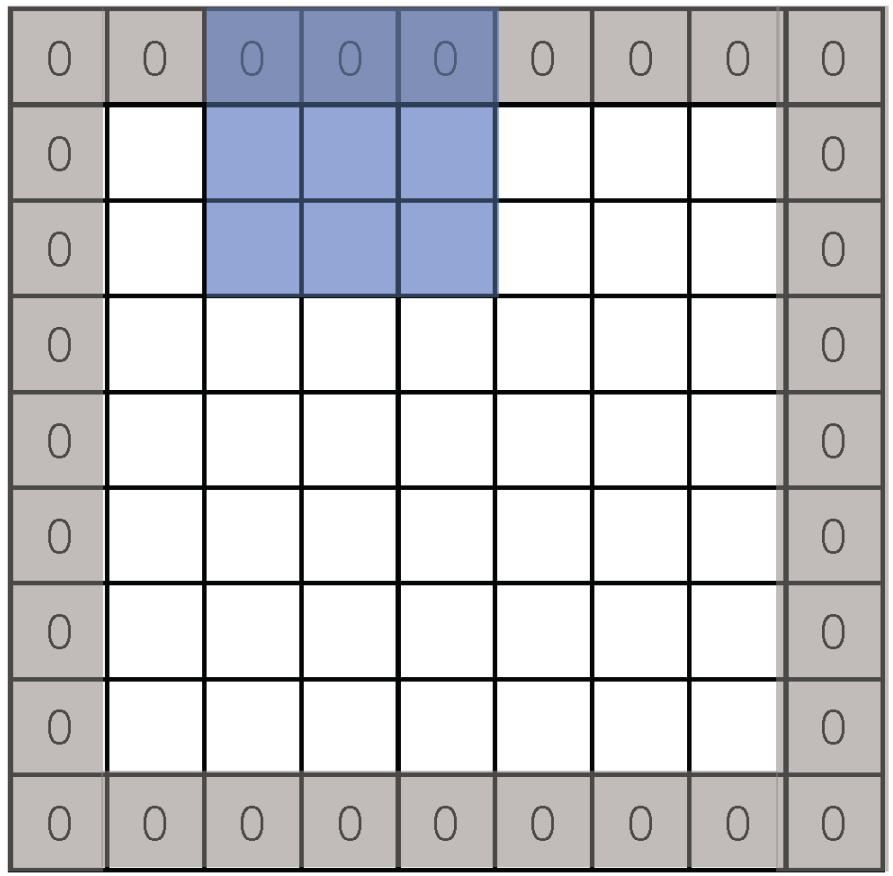
Input



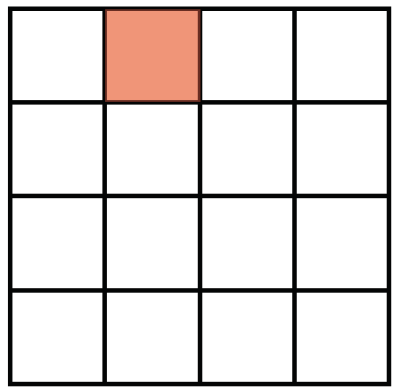
Output

Convolution: Padding

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



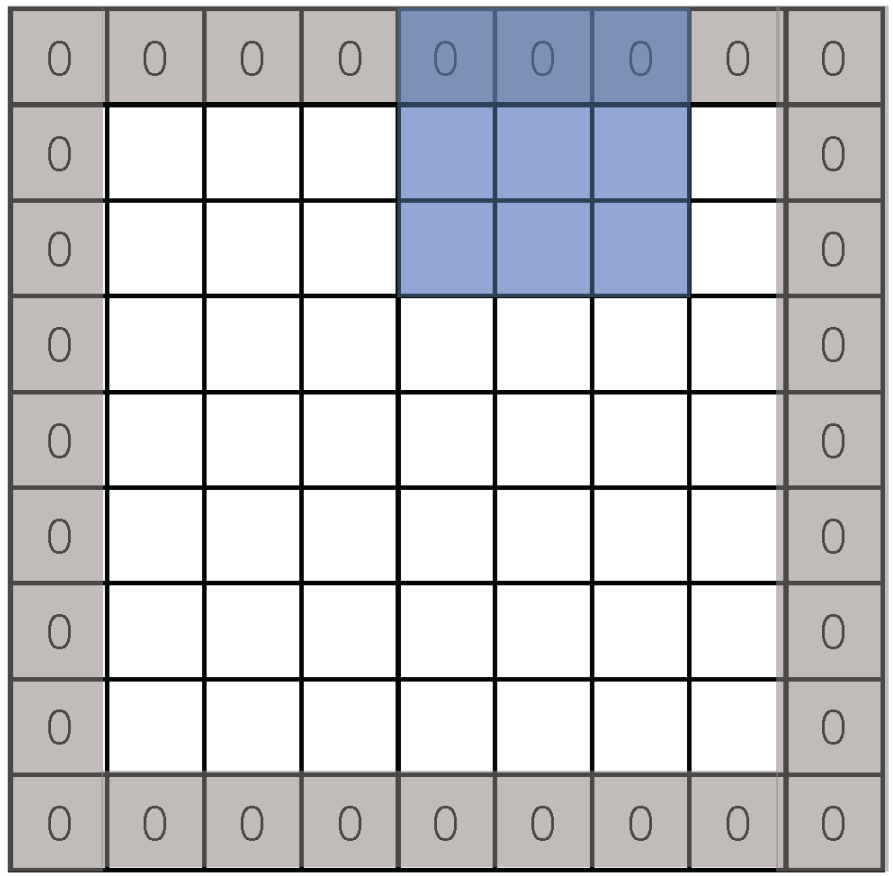
Input



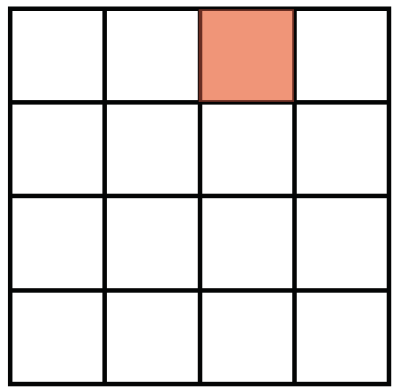
Output

Convolution: Padding

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



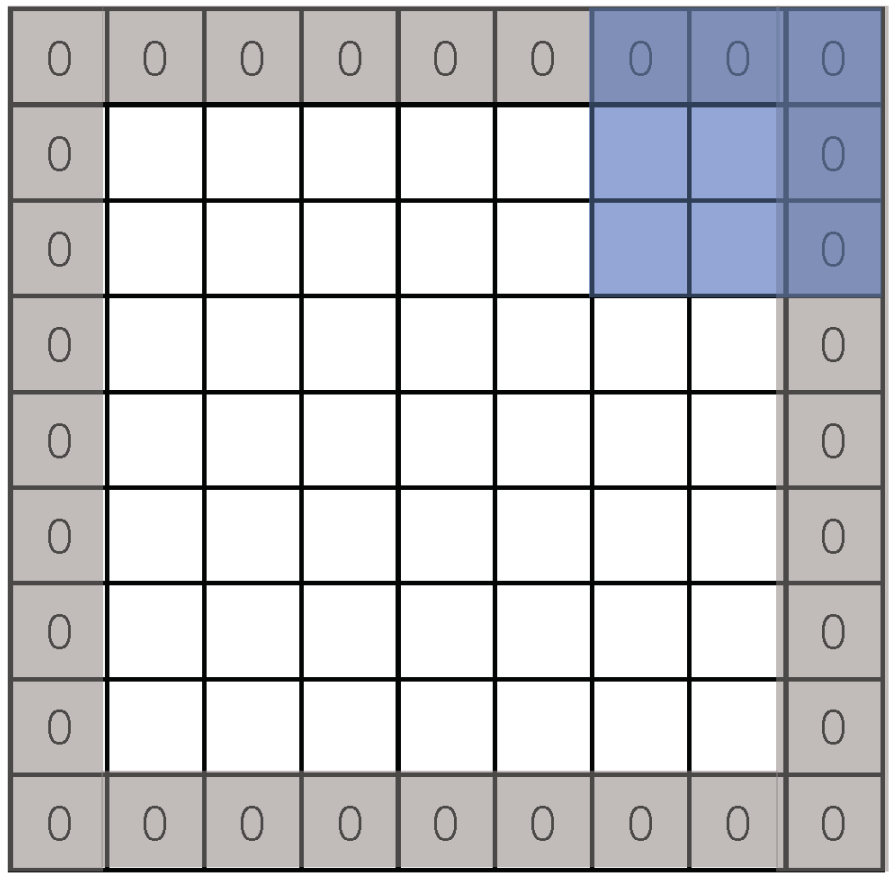
Input



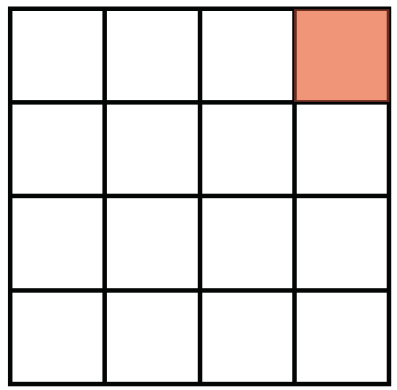
Output

Convolution: Padding

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



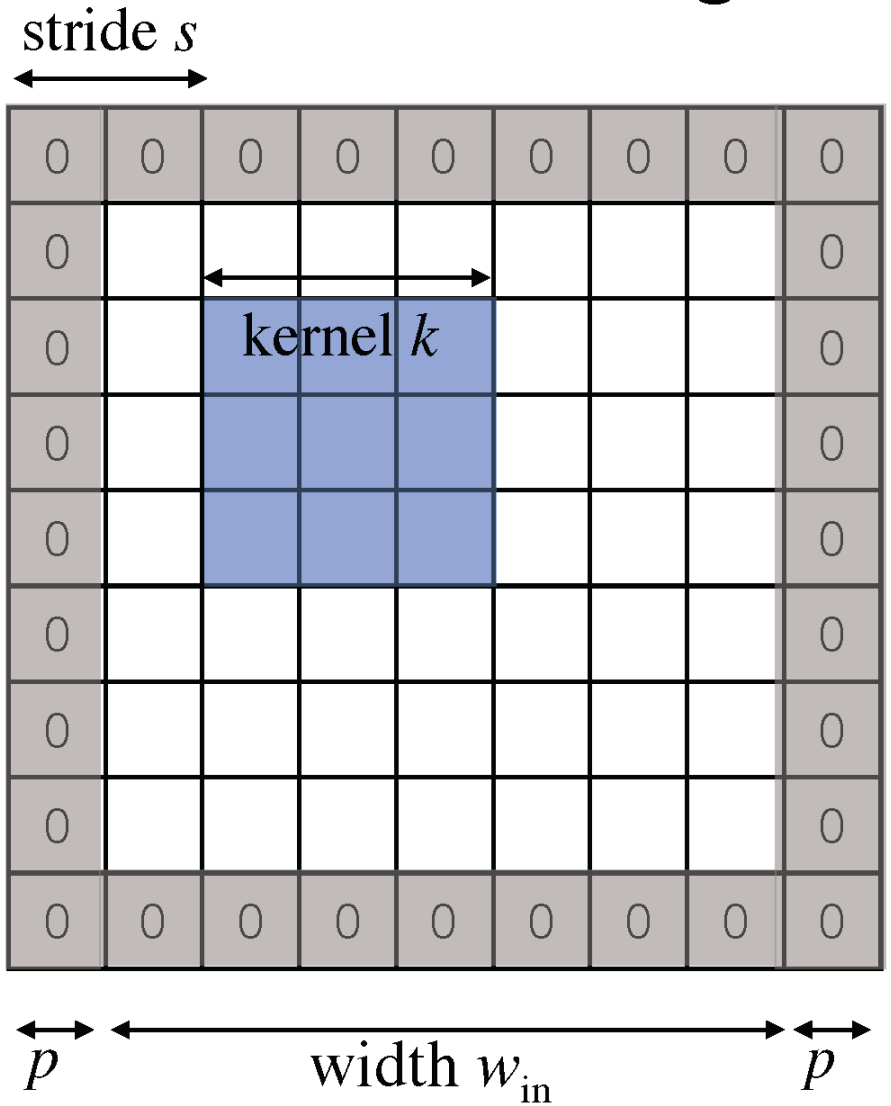
Input



Output

Convolution:

How big is the output?



We can choose different parameters for the convolution:

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

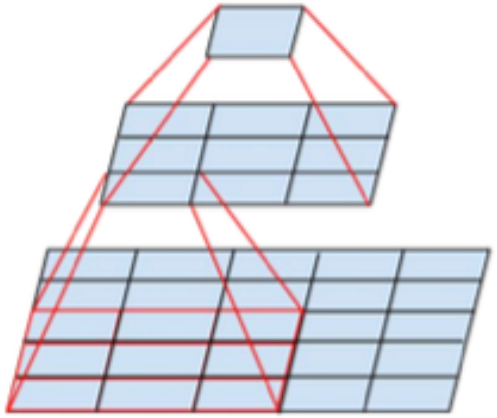
Such that w_{out} is an int.

Stacking conv layers

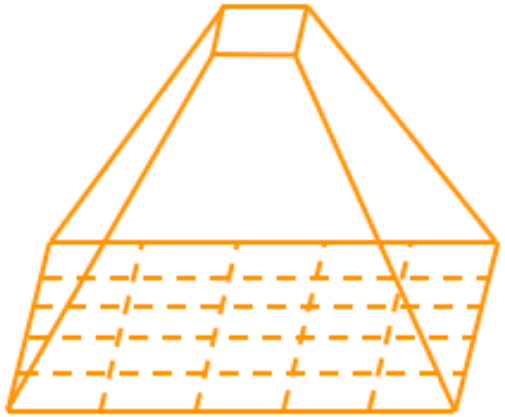
- What is better? One layer of 5x5 kernel or 2 layers of 3x3 kernel?

Stacking conv layers

- What is better? One layer of 5x5 kernel or 2 layers of 3x3 kernel?
- Their receptive fields are the same!



two successive
3x3 convolutions



5x5 convolution

Stacking conv layers

- What is better? One layer of 5x5 kernel or 2 layers of 3x3 kernel?
- Their receptive fields are the same!
- Less weights to learn! assuming C channels input and output- the single 5x5 CONV layer would contain $C \times (5 \times 5 \times C) = 25C^2$ parameters, while the three 3x3 CONV layers would only contain $2 \times (C \times (3 \times 3 \times C)) = 18C^2$.

Stacking conv layers

- What is better? One layer of 5x5 kernel or 2 layers of 3x3 kernel?
- Their receptive fields are the same!
- Less weights to learn! assuming C channels input and output- the single 5x5 CONV layer would contain $C \times (5 \times 5 \times C) = 25C^2$ parameters, while the three 3x3 CONV layers would only contain $2 \times (C \times (3 \times 3 \times C)) = 18C^2$.
- non-linearities between the layers make their features more expressive.

Stacking conv layers

- What is better? One layer of 5x5 kernel or 2 layers of 3x3 kernel?
- Their receptive fields are the same!
- Less weights to learn! assuming C channels input and output- the single 5x5 CONV layer would contain $C \times (5 \times 5 \times C) = 25C^2$ parameters, while the three 3x3 CONV layers would only contain $2 \times (C \times (3 \times 3 \times C)) = 18C^2$.
- non-linearities between the layers make their features more expressive.

- 2 layers of 3x3 kernel is the way to go!
- Adding strides (and padding where needed) can make the receptive fields even larger with fewer weights to learn.

contents

- Chain rule
- **ConvNets**
  - Convolution layer
  - **Pooling layer**
- Overfitting
- Architectures
  - Alexnet (dropout)
  - VGG
  - ResNet (batch norm)

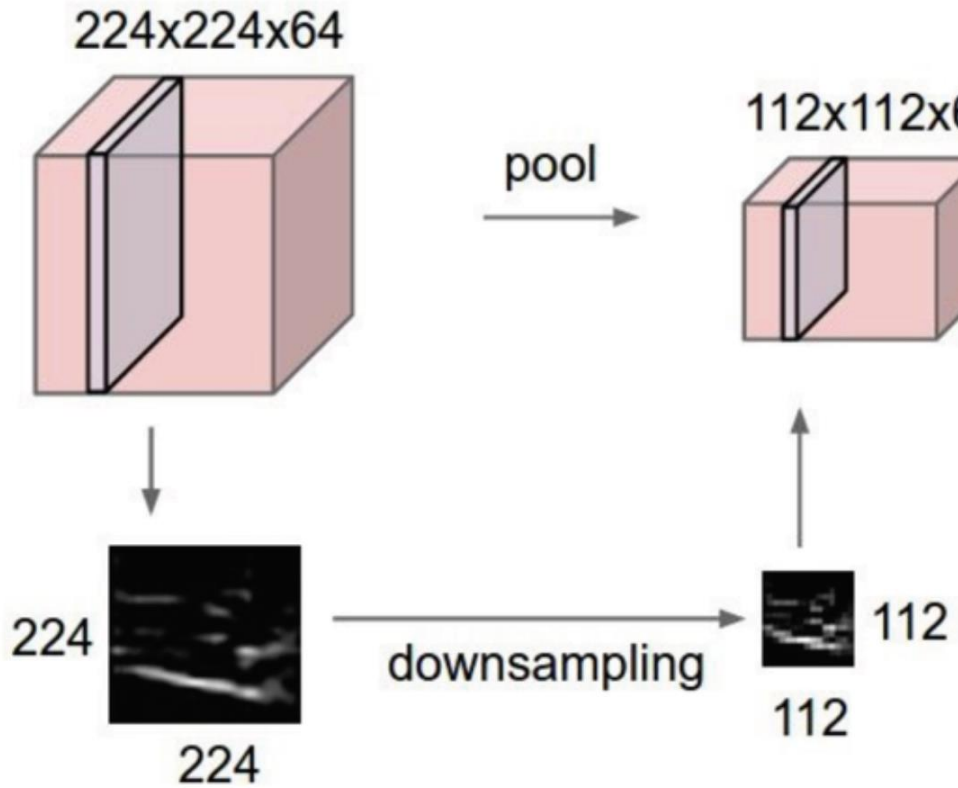
Pooling

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

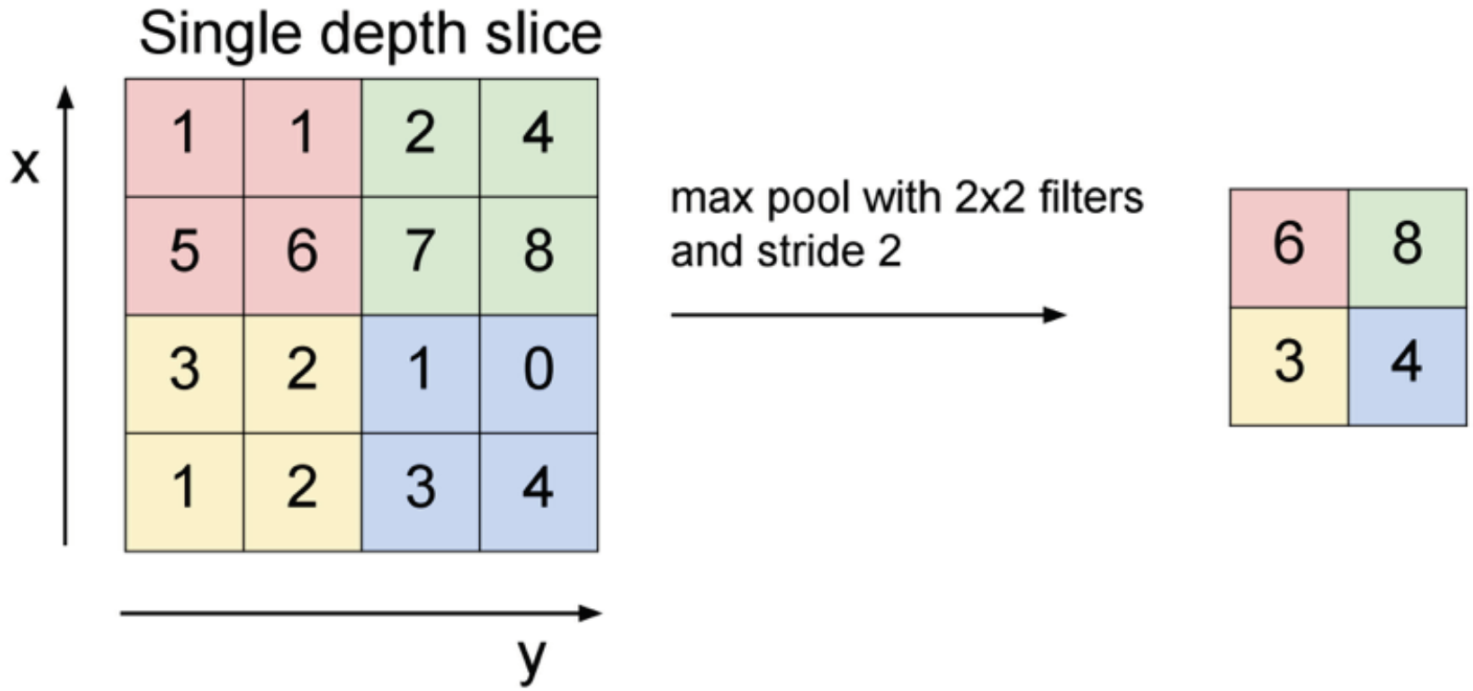
- Creates a smaller representation while retaining the most important information
- The “max” operation is the most common

Pooling

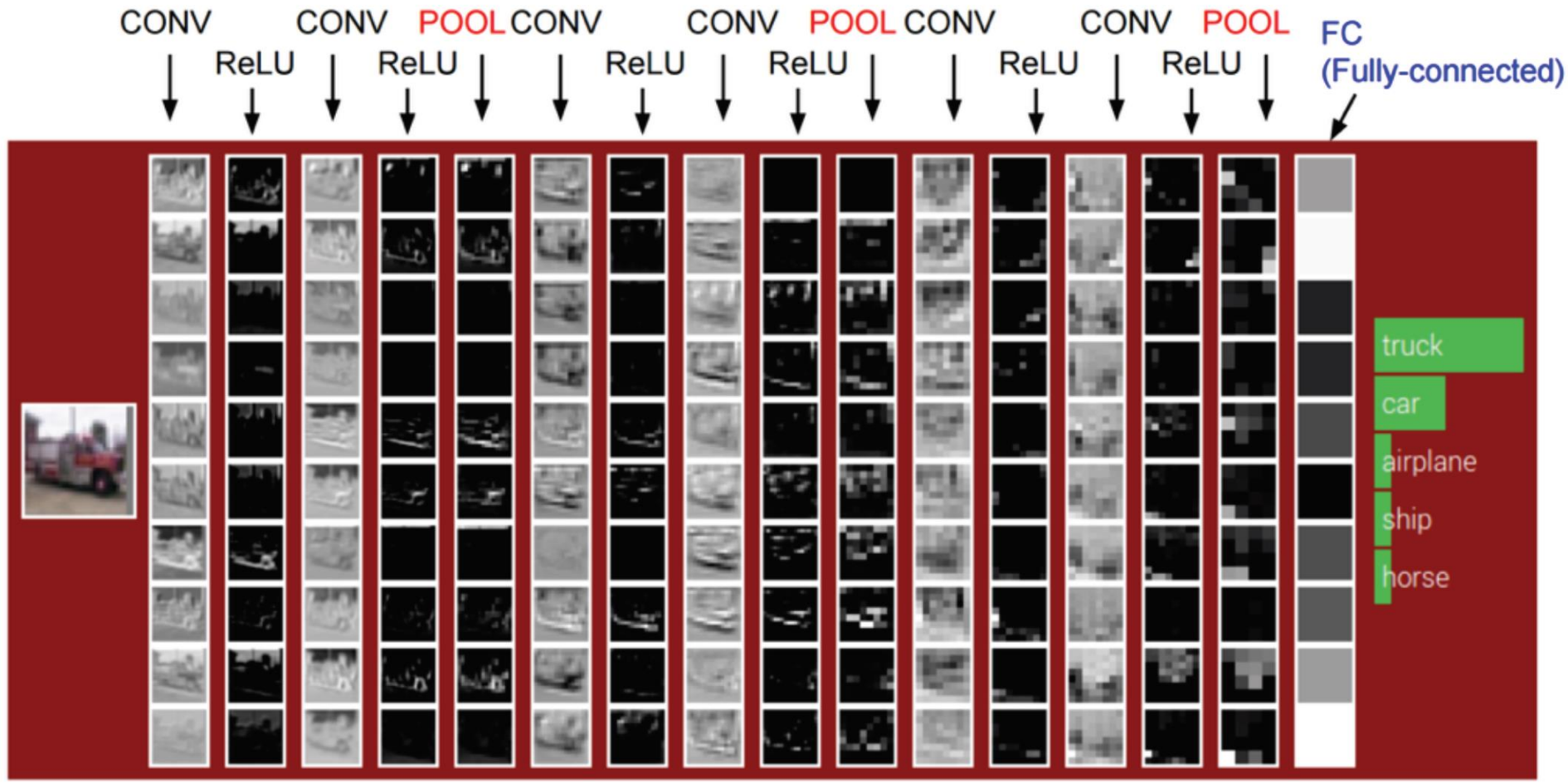
- makes the representations smaller and more manageable
- operates over each activation map independently:



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

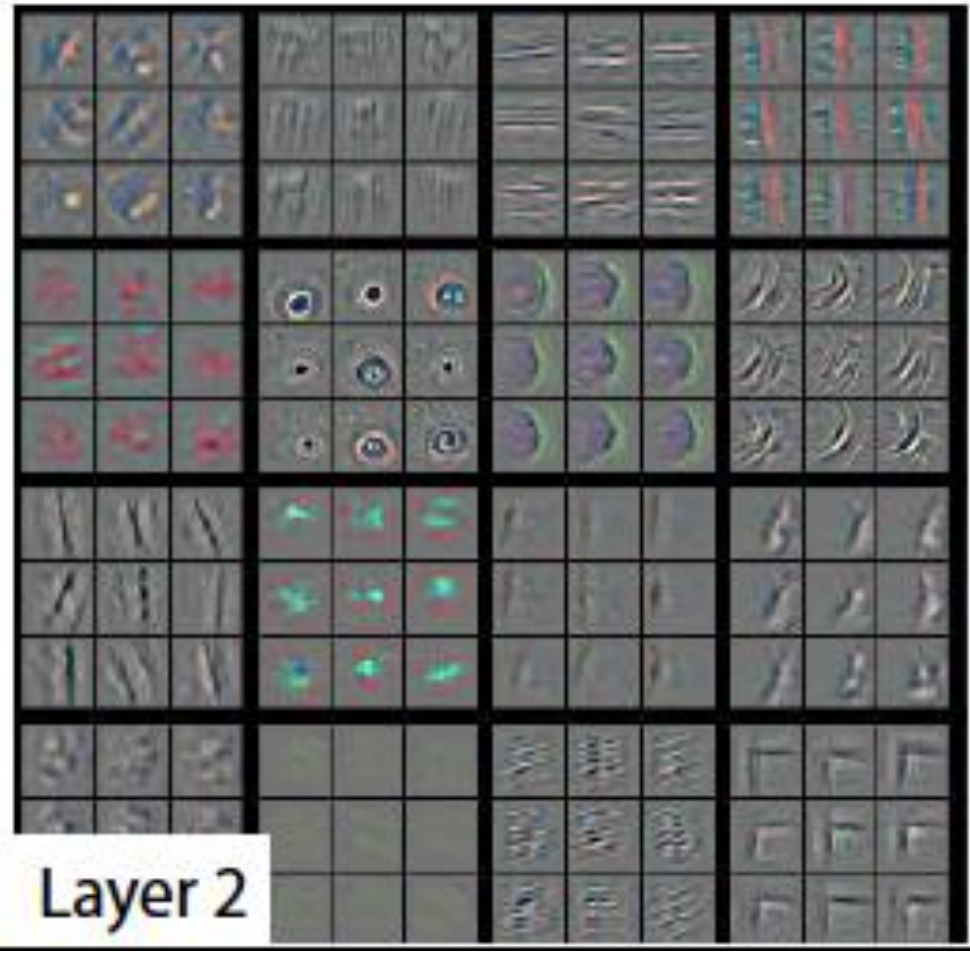
Figure: Andrej Karpathy

Layer depth

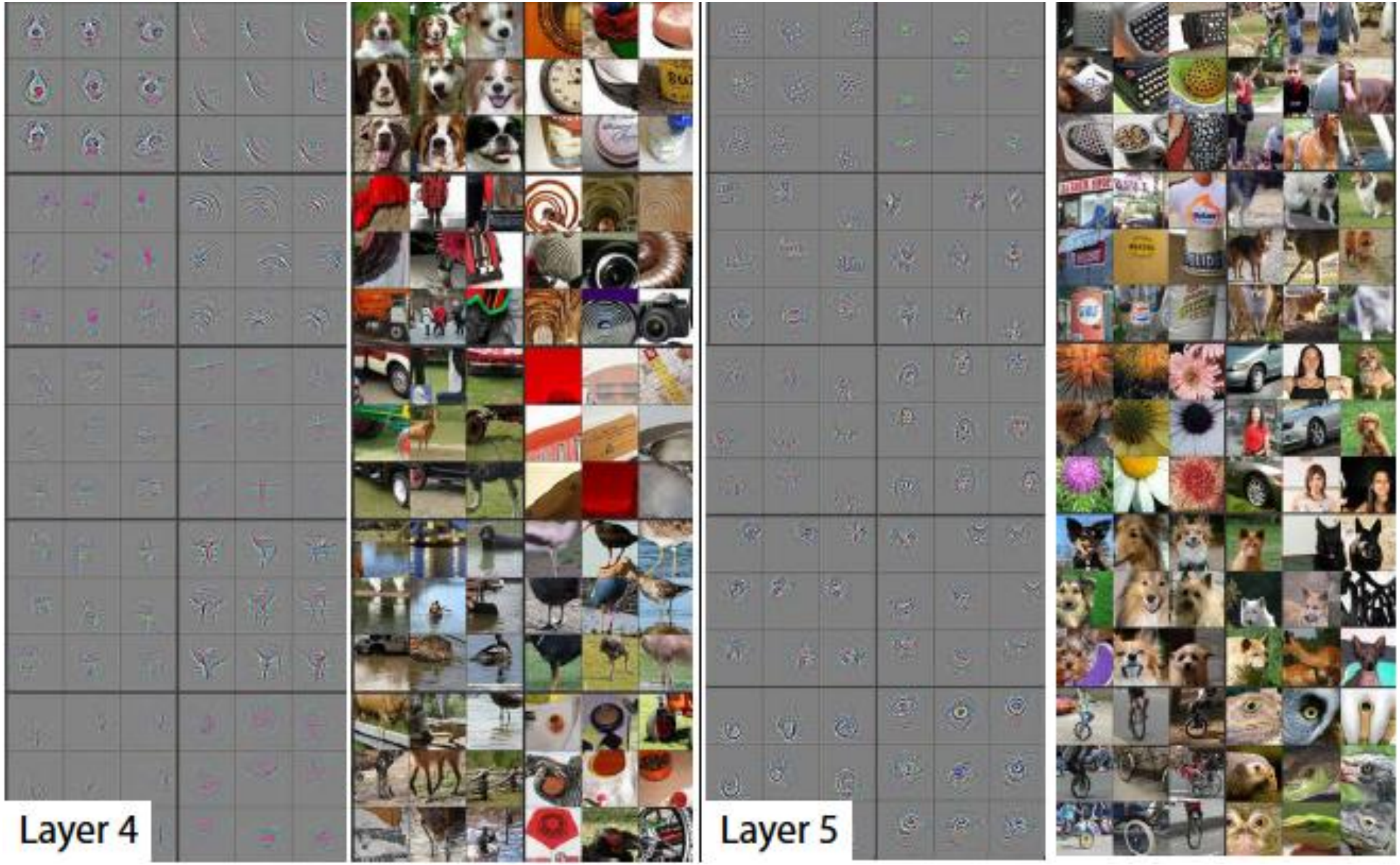
- First layers detect simple patterns (colors, edges, etc.)
- Deeper layers detects more complex patterns which is built upon the output of the first layers.

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

Layer 1



Layer 2


Layer 4
Layer 5

- Conv colab

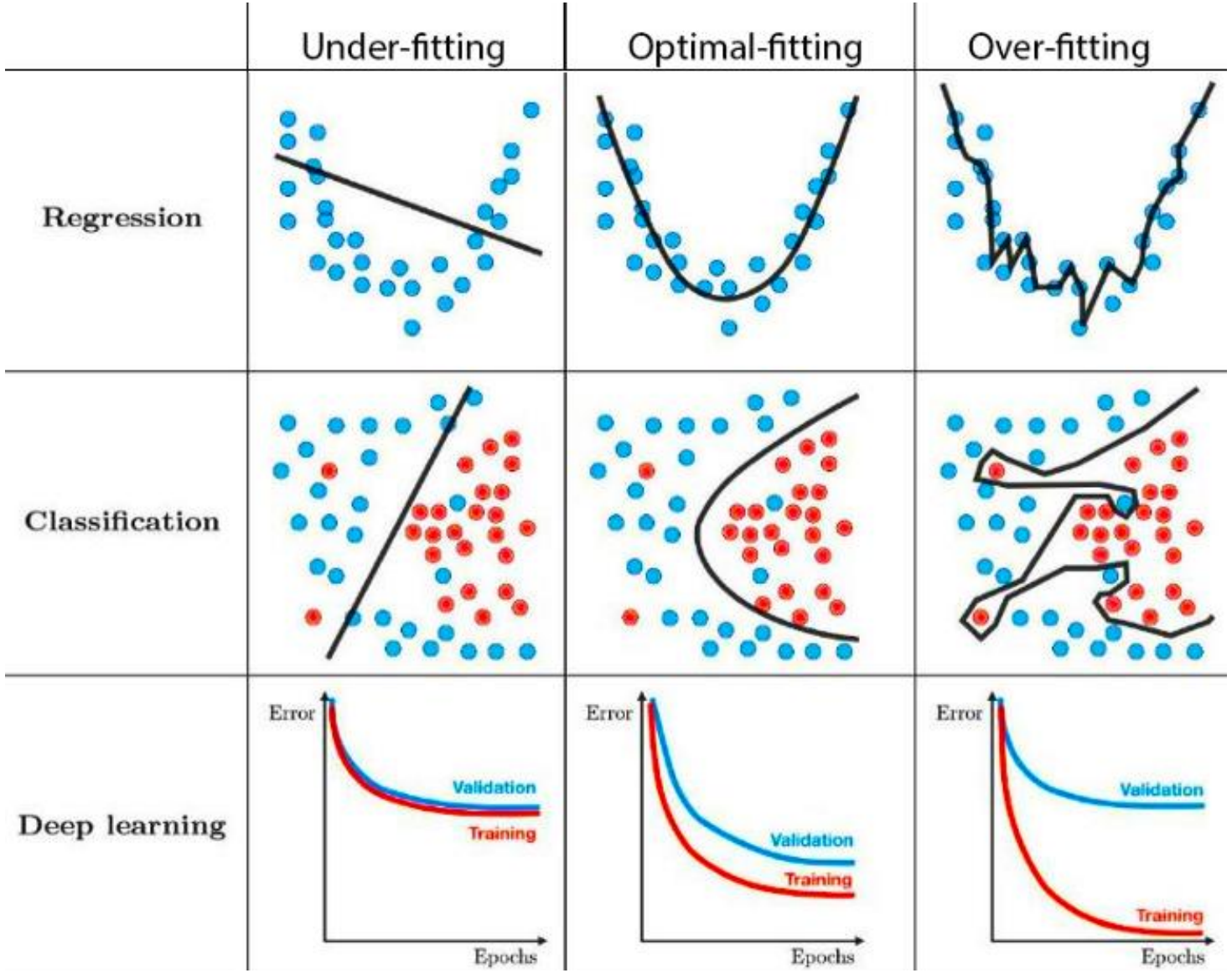
contents

- Chain rule
- ConvNets
  - Convolution layer
  - Pooling layer
- **Overfitting**
- Architectures
  - Alexnet (dropout)
  - VGG
  - ResNet (batch norm)

Overfitting

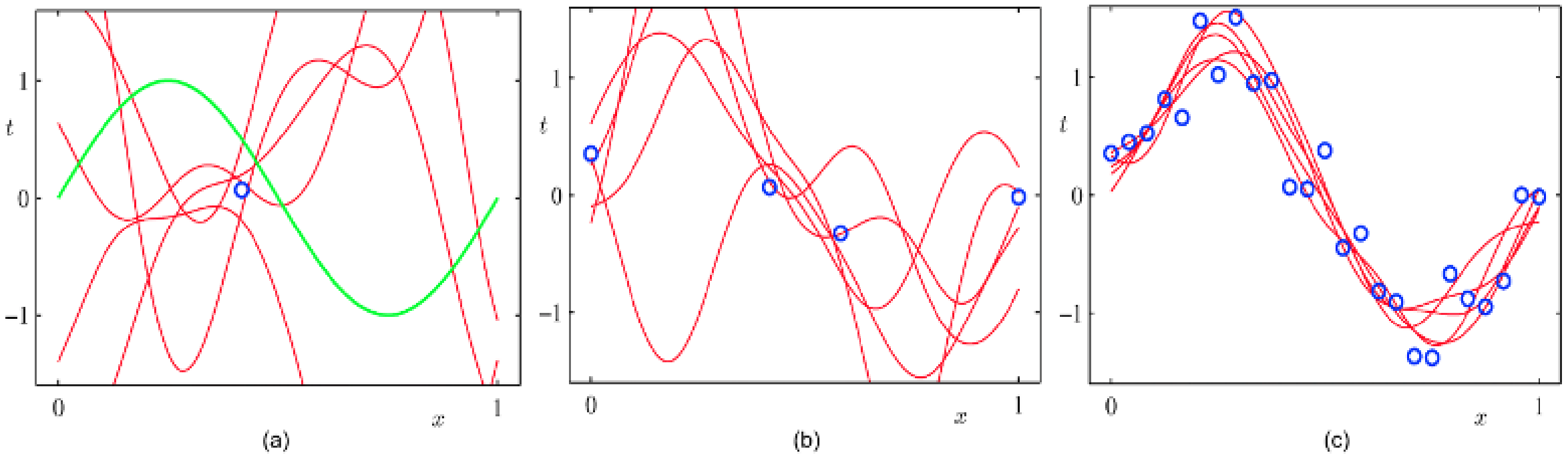
- Overfitting: analysis that corresponds **too closely or exactly to a particular set of data and** may therefore **fail to fit additional data** or predict future observations reliably.

- Underfitting: insufficiently modeling the relationship of the data points.



Overfitting

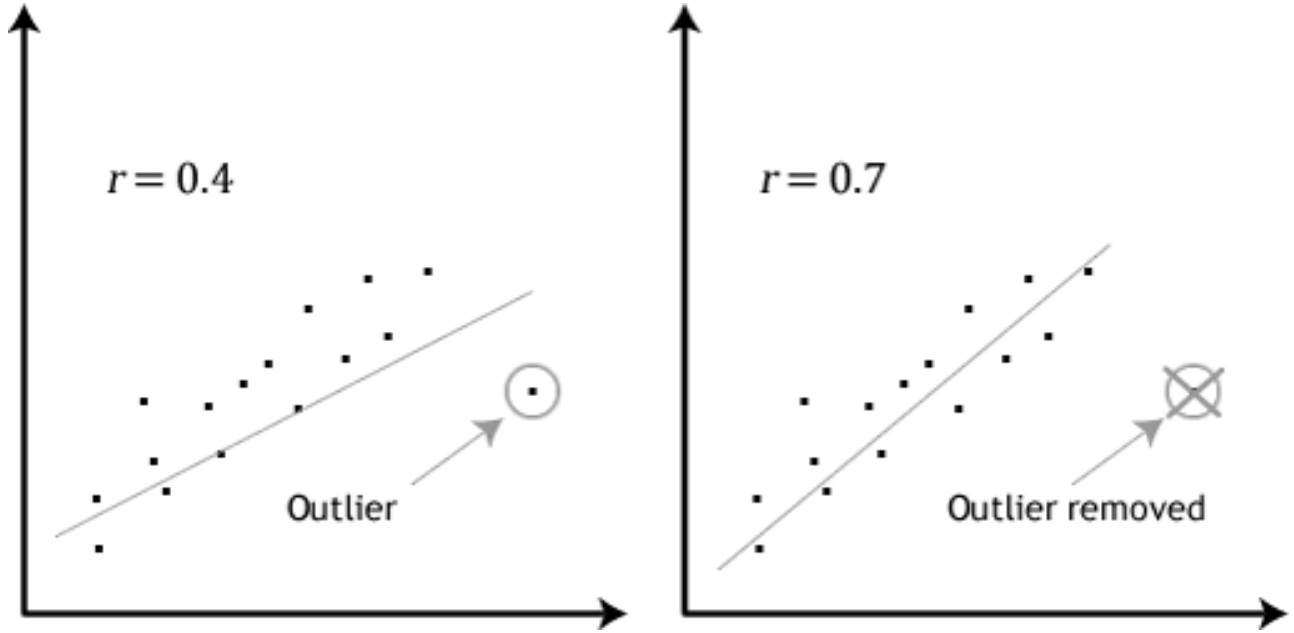
- Overfitting is one of the biggest problems in NN.
- **It is said that for a net to not overfit we need X10 data-samples as weights.**
- What to do when you don't have millions of examples?



Example of trying to fit small dataset to a model with a lot of DOFs.

Remove outliers

- Clean the data so that that its variance would be relatively small.
- With a small dataset- even one outlier datapoint can be problematic.

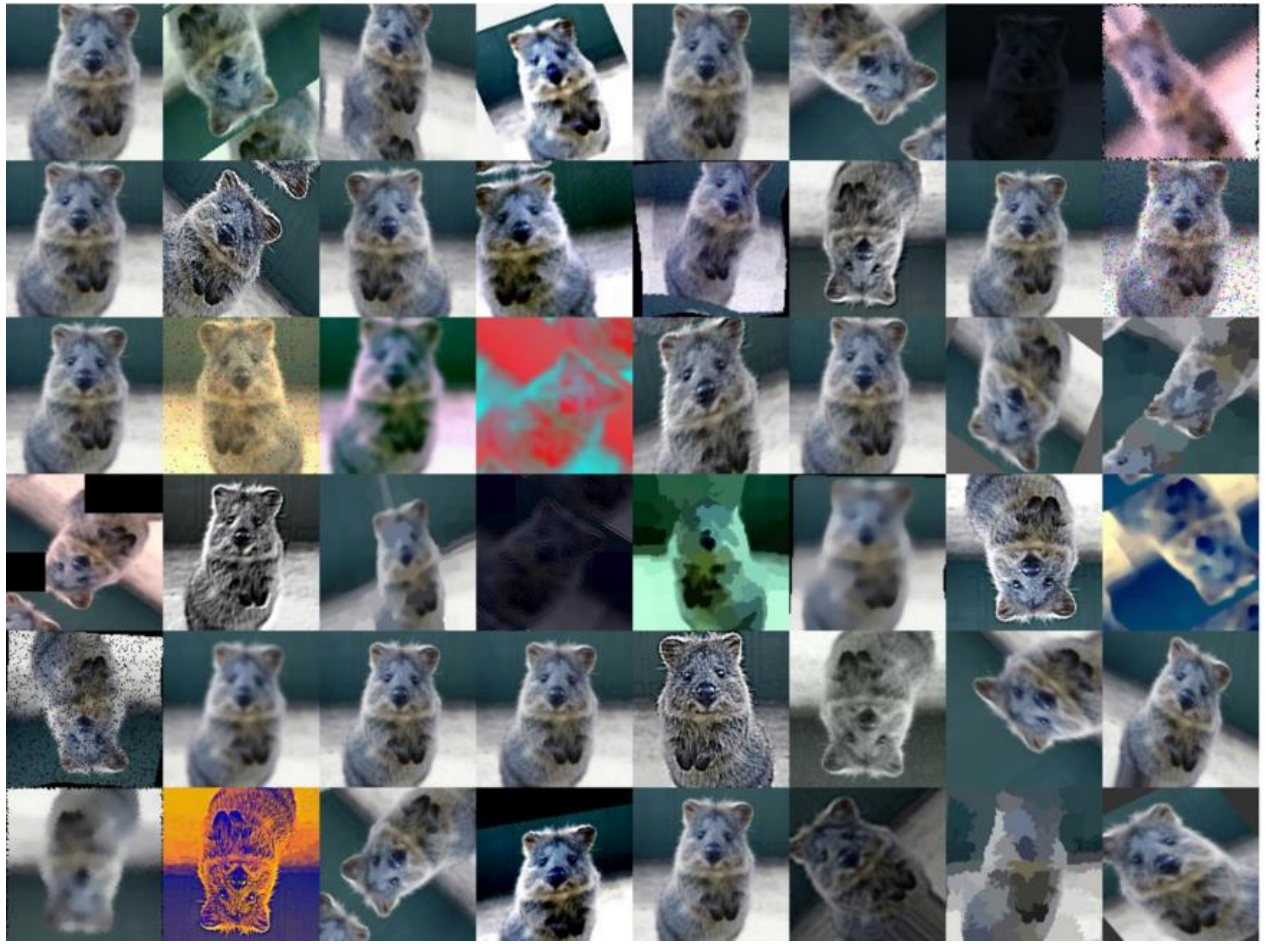


bat

Also a bat – but not what we looked for

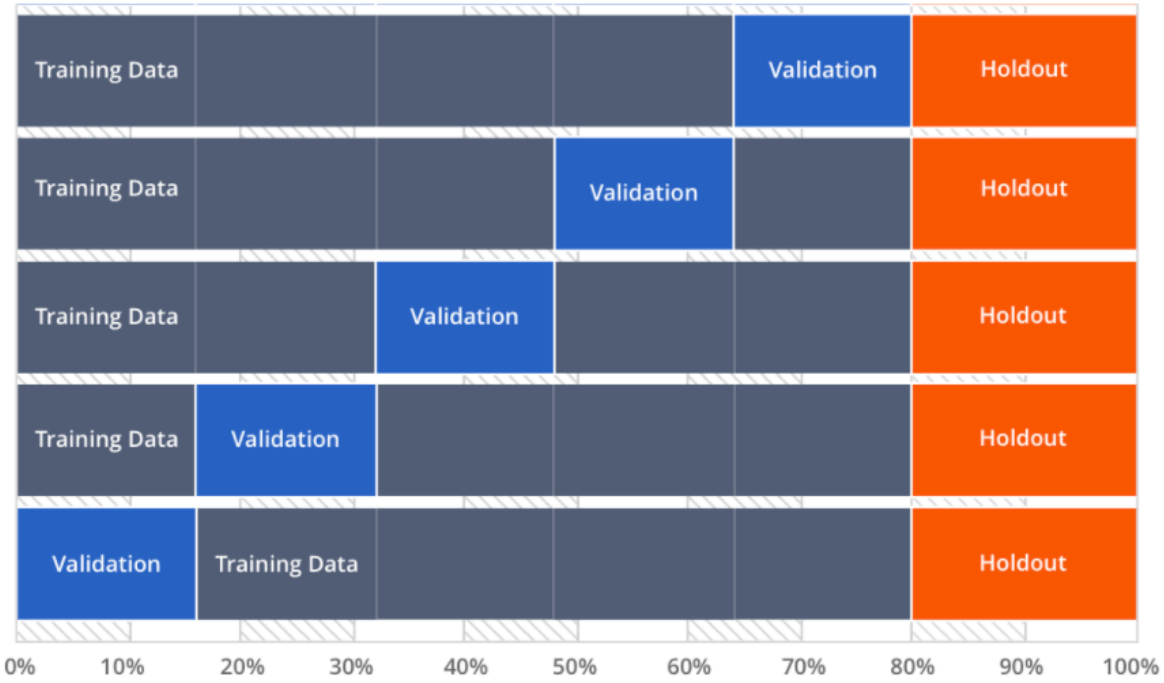
Augmentations

- Technique used to increase the amount of data by adding slightly modified copies of already existing data
- Examples: rotation, translation, scale, shear, brightness, contrast, noise, crop, warping etc…



Cross validation

- Partitioning a sample of data into complementary subsets and validating the analysis on the other subset. Validation results are averaged over the rounds to give an estimate of the model's predictive performance.
- The final step is to train the model on the entire dataset and by doing you get more training data because you don’t need a validation set.
- Downside: training time is longer.

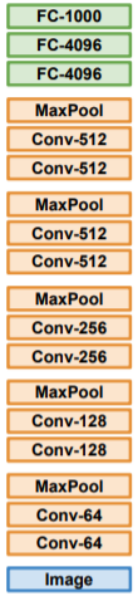


Transfer learning

- Take a pretrained model on a larger similar dataset and train again a subset of its layers to your new dataset.
- The idea is that the features learned in the first layers are still relevant to your data, and all you need is to train the more complex layers near the output to learn your new data.

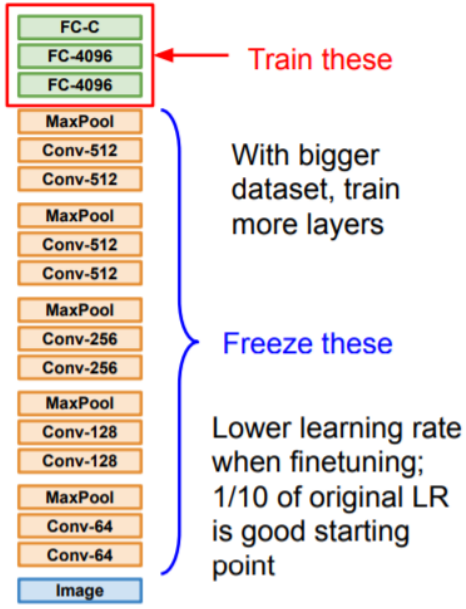
Transfer learning

1. Train on Imagenet



2. Small Dataset (C classes)

3. Bigger dataset



	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

Overfitting- the upside

- Overfitting isn’t always bad- there is a known trick to check the correctness of your NN: **take a small batch of data and try intentionally to overfit it!**
- This will show you if you have any underlying problems in your net architecture before you start train.

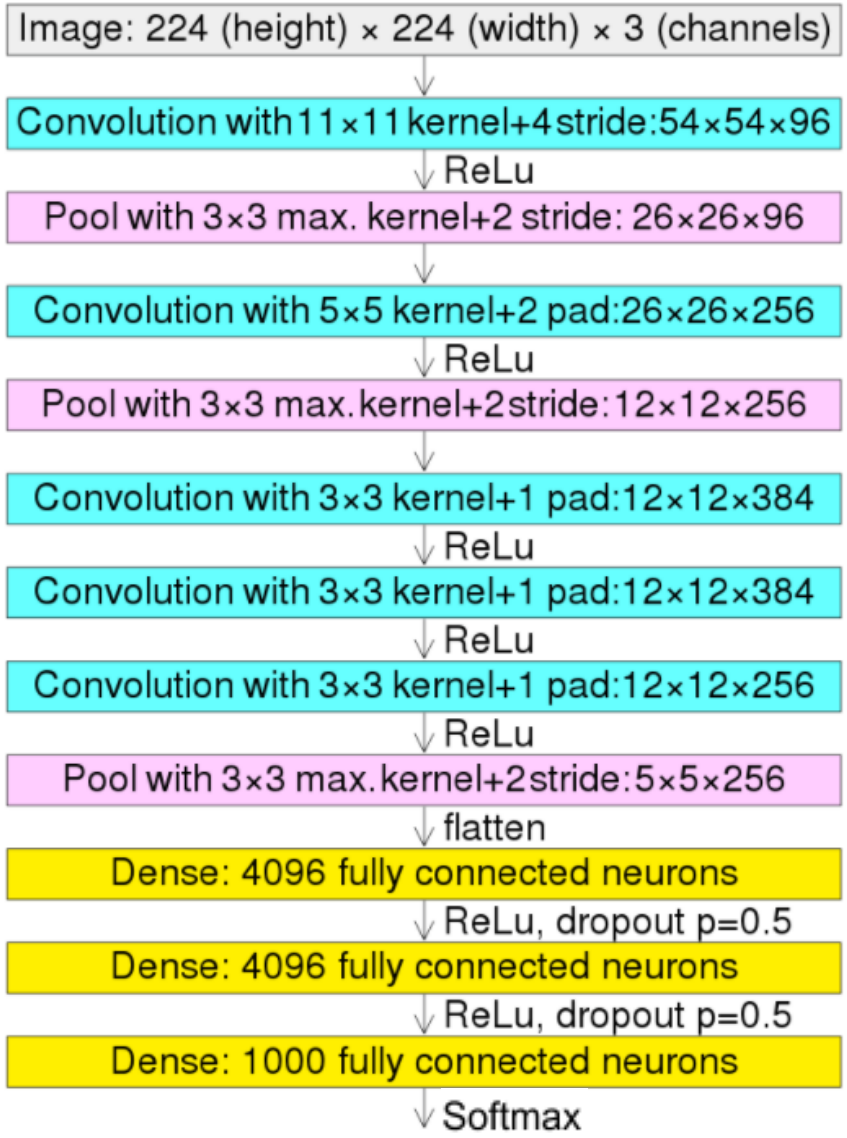
contents

- Chain rule
- ConvNets
  - Convolution layer
  - Pooling layer
- Overfitting
- **Architectures**
  - Alexnet (dropout)
  - VGG
  - ResNet (batch norm)

Alexnet

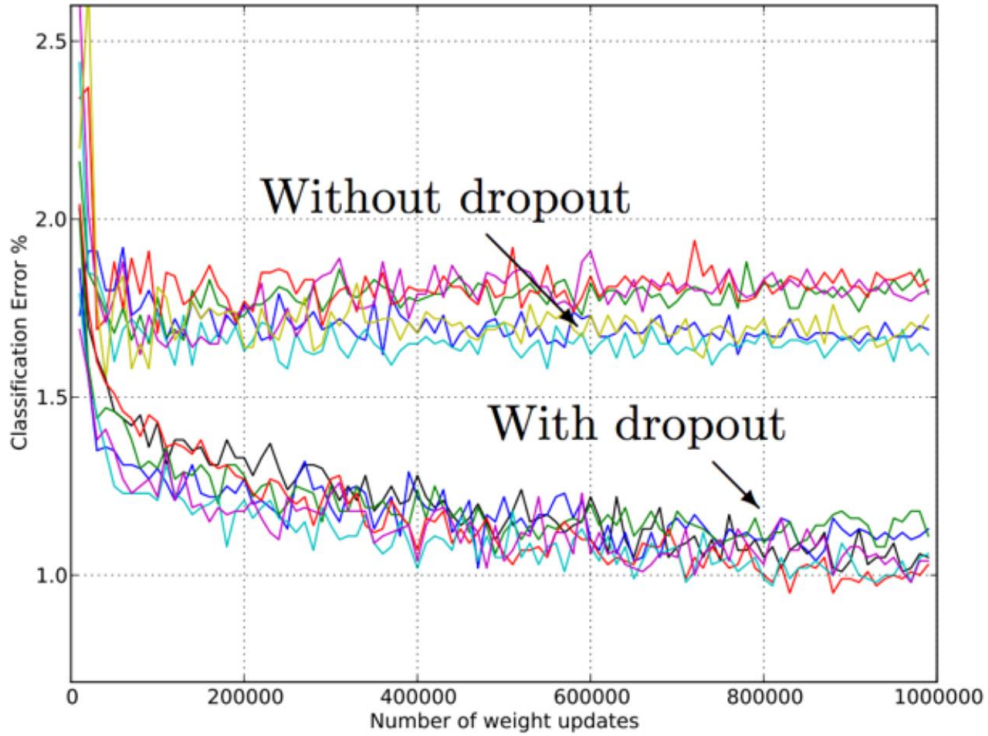
- The winner of the 2012 IMAGENET competition (classification % of top 5 out of 1000 labels).

- 224X224X3 input.
- 8 learnable layers.
- 61M parameters.



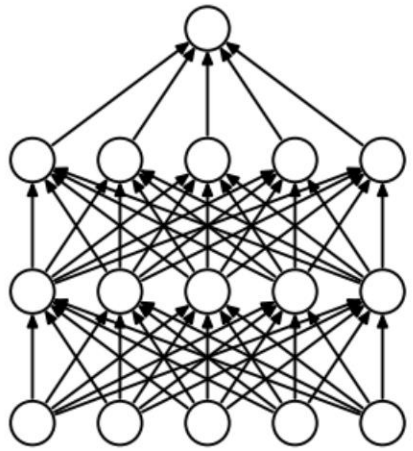
Alexnet

- Interesting facts:
  - The first winner of IMAGENET who used NN- started the NN trend we are feeling now.
  - Not really used today.
  - Used dropout as a technique to avoid overfitting.

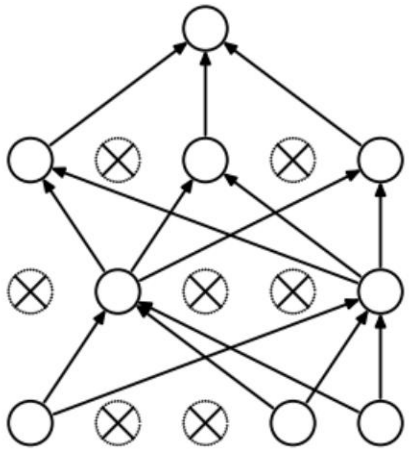


dropout

- Neuron is dropped from the network with a probability of 0.5. When a neuron is dropped, it does not contribute to either forward or backward propagation.
- Every input goes through a different network architecture, as a result, the learnt weight parameters are more robust.
- During testing, there is no dropout and the whole network is used, but output is scaled by a factor of 0.5 to account for the missed neurons while training.
- Dropout increases the number of iterations needed to converge by a factor of 2.



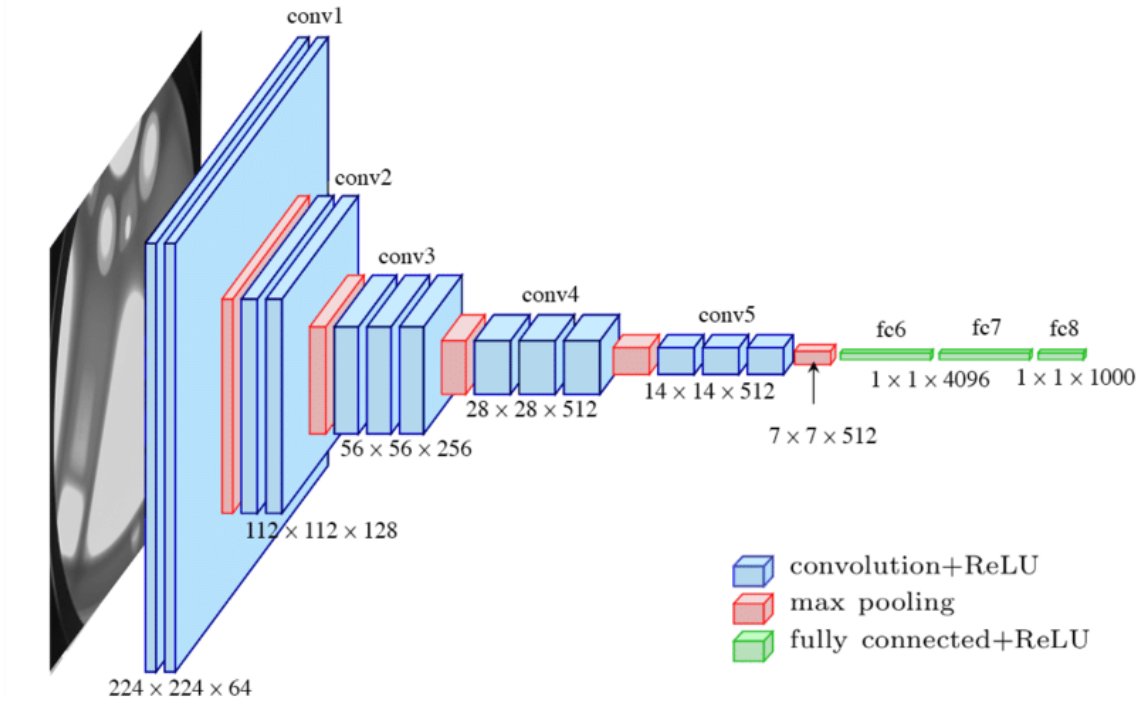
(a) Standard Neural Net



(b) After applying dropout.

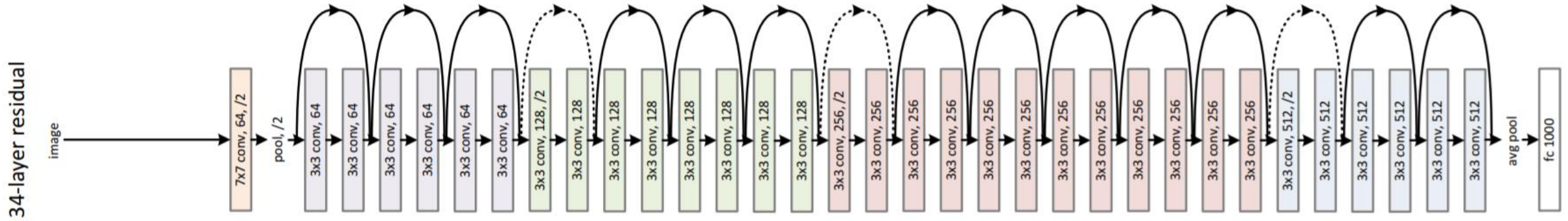
VGG16

- Winner of IMAGENET 2014 classification challenge.
- 16 layers (there are also other variations with 11,13 and 19 layers).
- 138M parameters.
- Used small receptive fields with stacked conv layers (shown before)- compared to 11X11 and 5X5 in AlexNet, her there is only 3X3 layers.
- Still in use today!



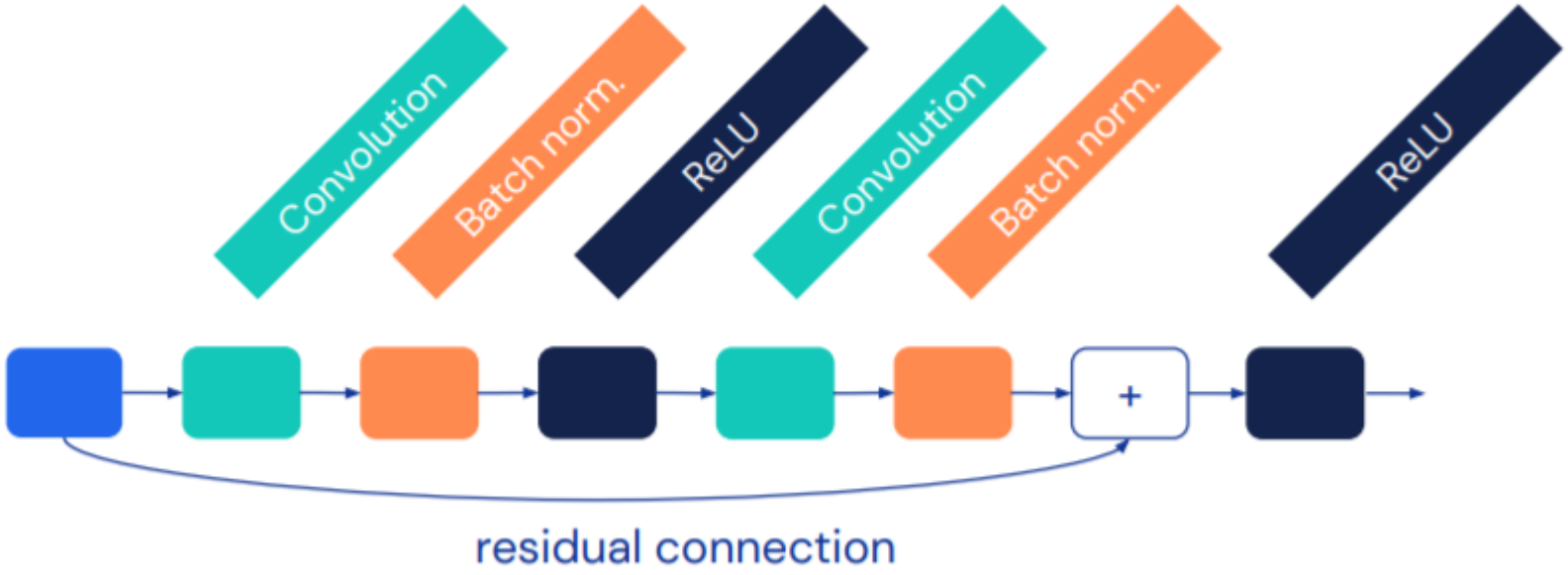
ResNet

- Winner of IMAGENET 2015 classification challenge.
- 152 layers (there are also other variations with 18,34,50 and 101 layers).
- 60M parameters (<0.5X VGG16!!!).
- Still in use today!



ResNet

- Interesting blocks:
  - Batch norm (for vanishing gradient problem).
  - Residual block (for train accuracy degradation) – out of scope.



Vanishing gradient