Convolutional Neural Networks

Deep Learning

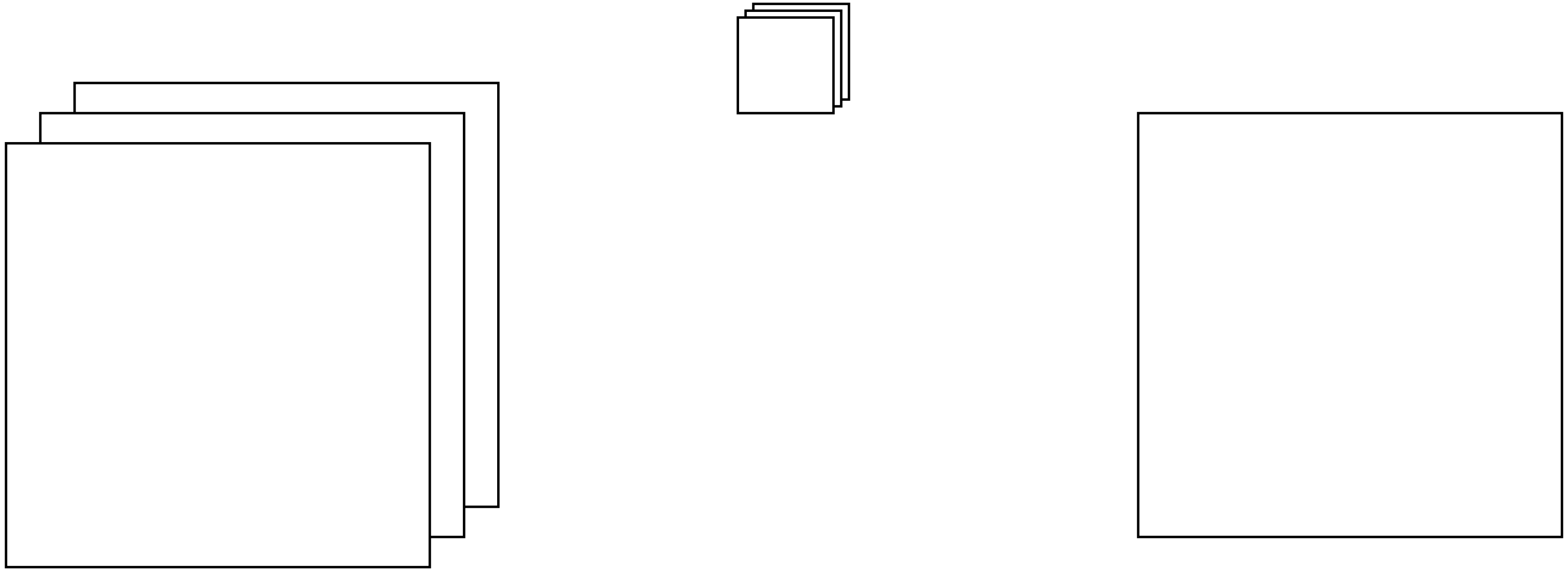
[Brad Quinton](), [Scott Chin]()

Learning Objectives

- Understand how a Convolutional Layer works
- Understand how we can build a Convolutional Neural Network using Convolutional and Fully-Connected Layers
- Understand how we can ”transform” a Convolutional Layer to a Fully-Connected Layer and vice-versa
- Understand what stride and padding are
- Understand how Pooling Layers work

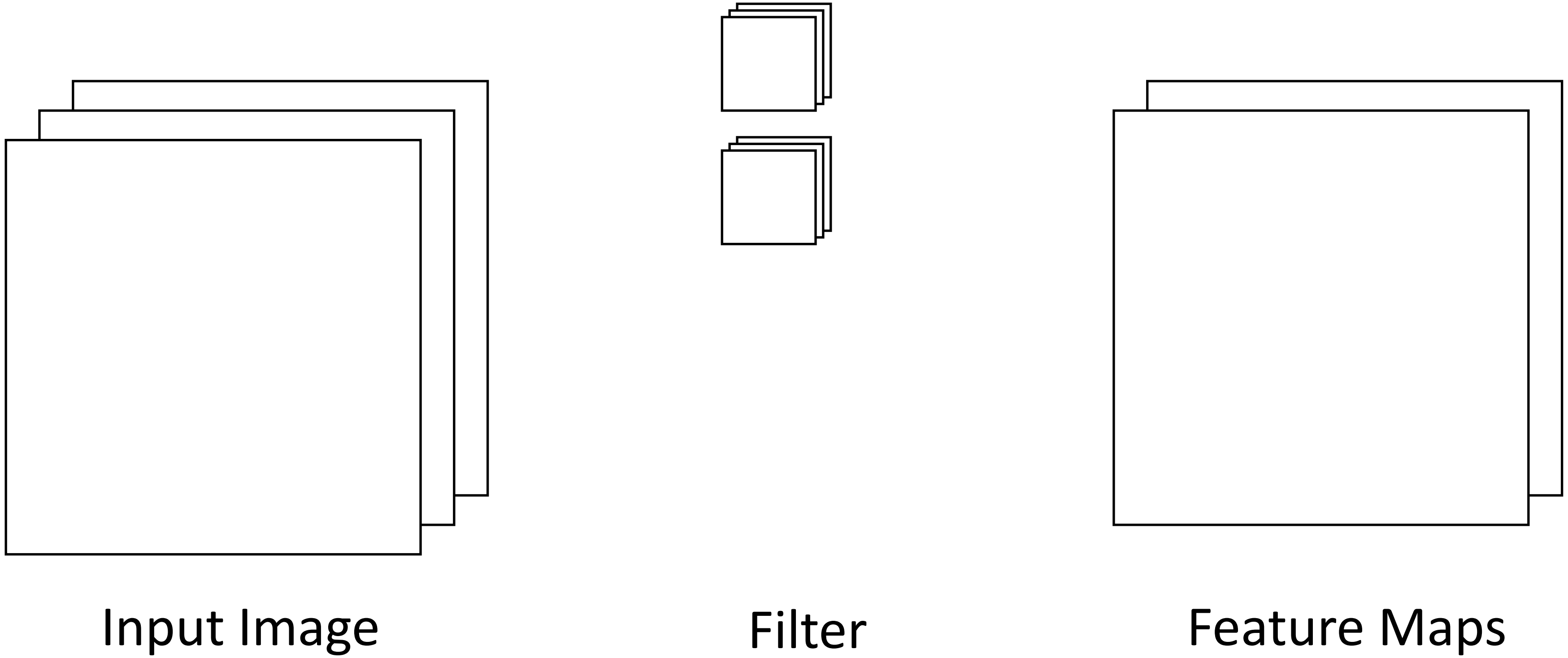
Convolutional Layer

So far, talking about one filter



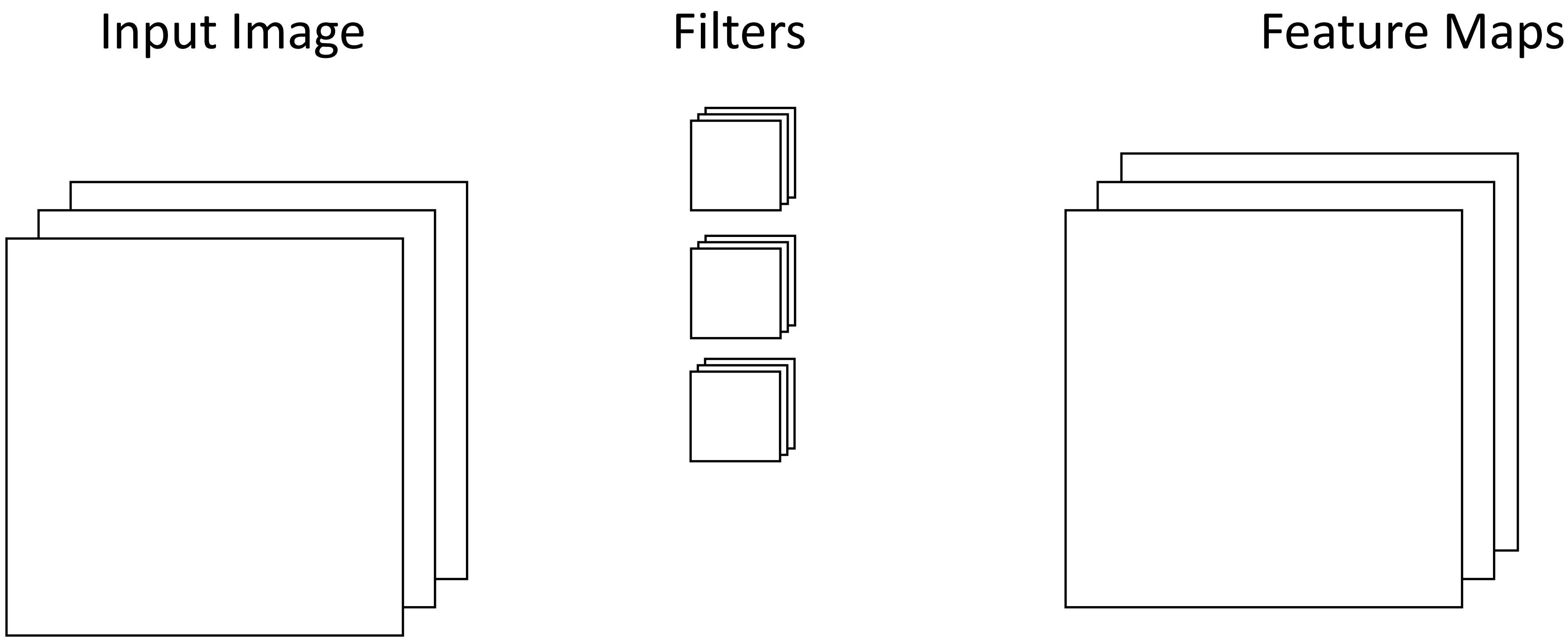
Input Image Filter Feature Map

Want to look for more than one feature

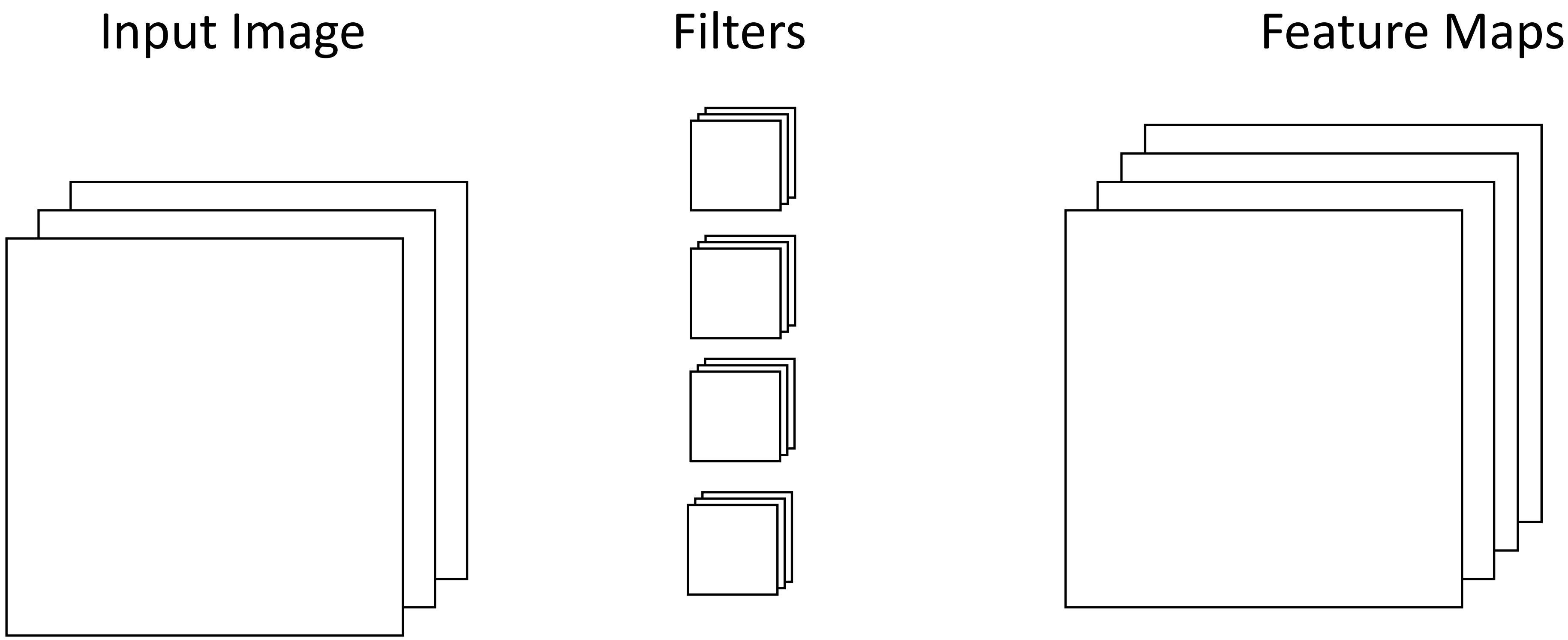


Brad Quinton, Scott Chin

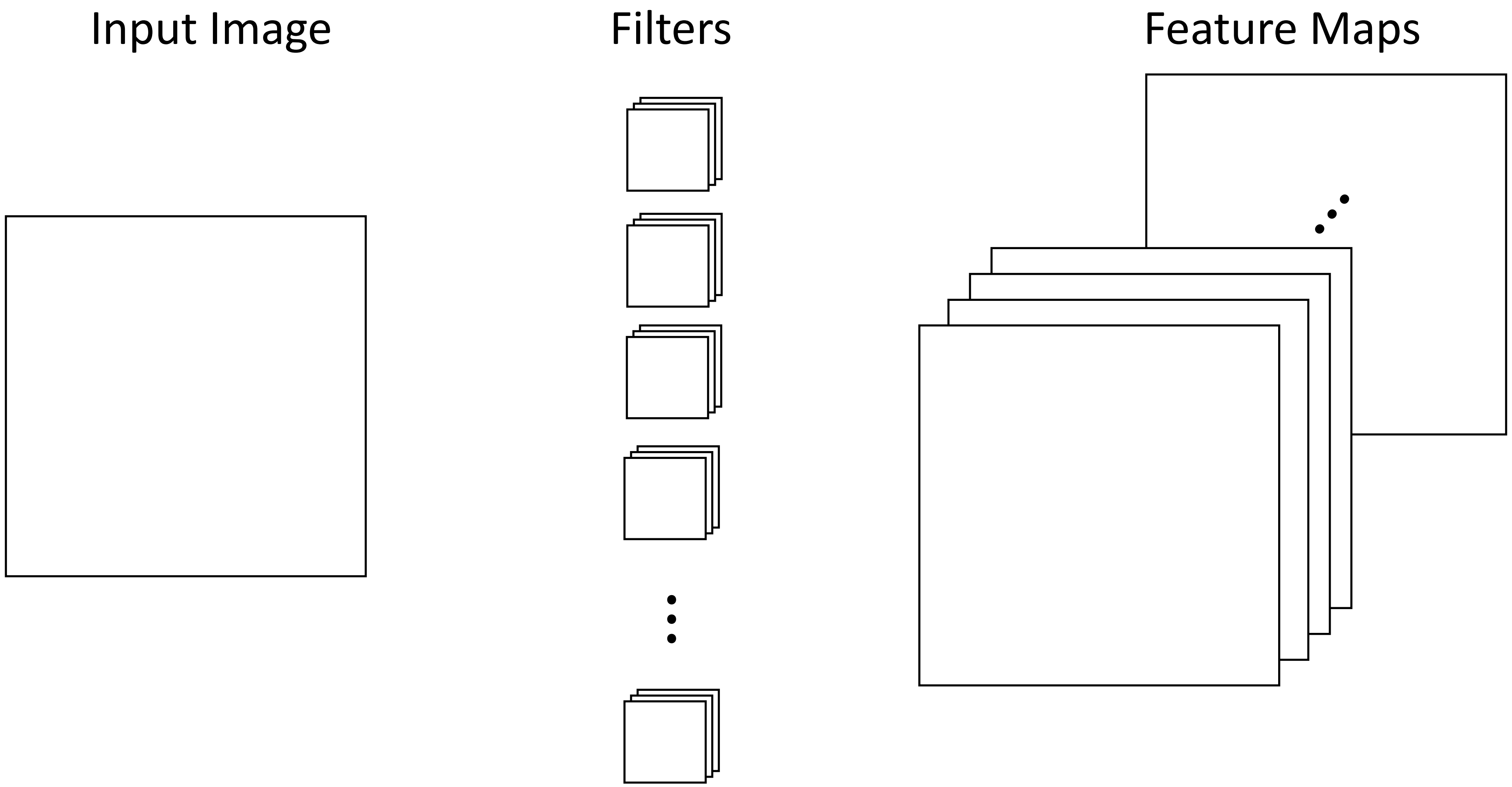
Working towards defining a Convolutional Layer



Working towards defining a Convolutional Layer

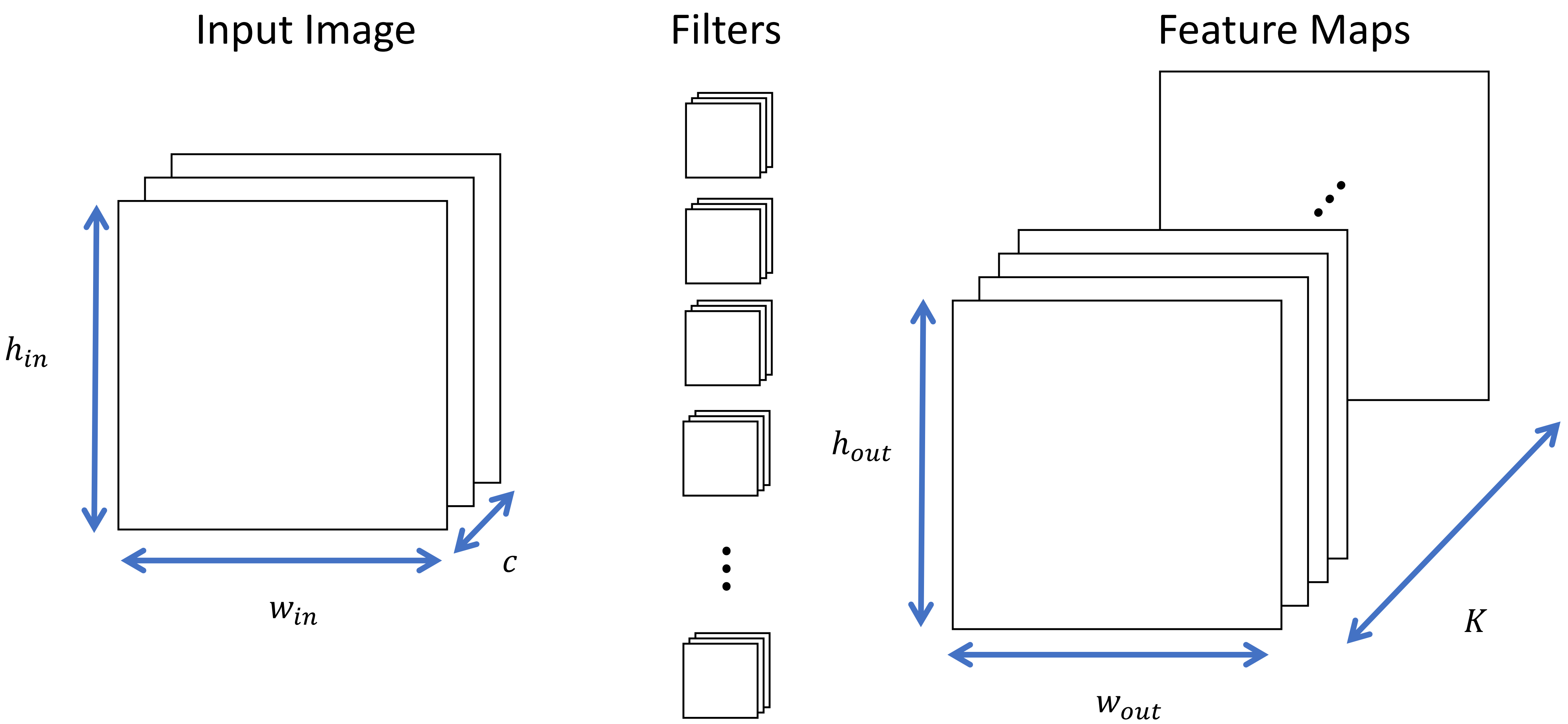


Working towards defining a Convolutional Layer



Brad Quinton, Scott Chin

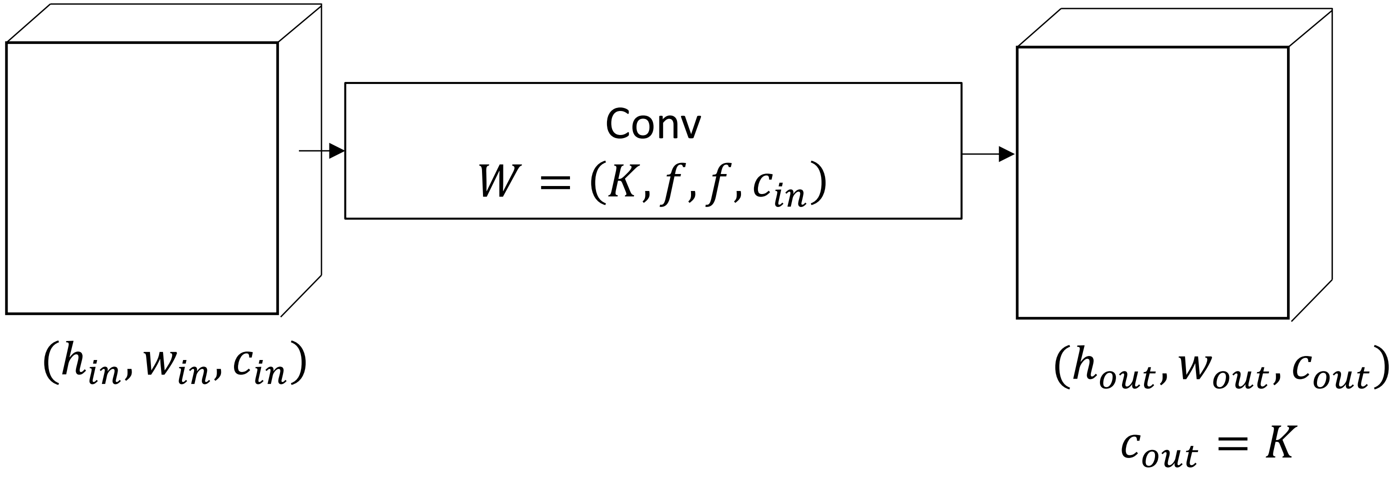
Working towards defining a Convolutional Layer



- K number of (f, f, c)
- Can think of each filter as a "neuron"

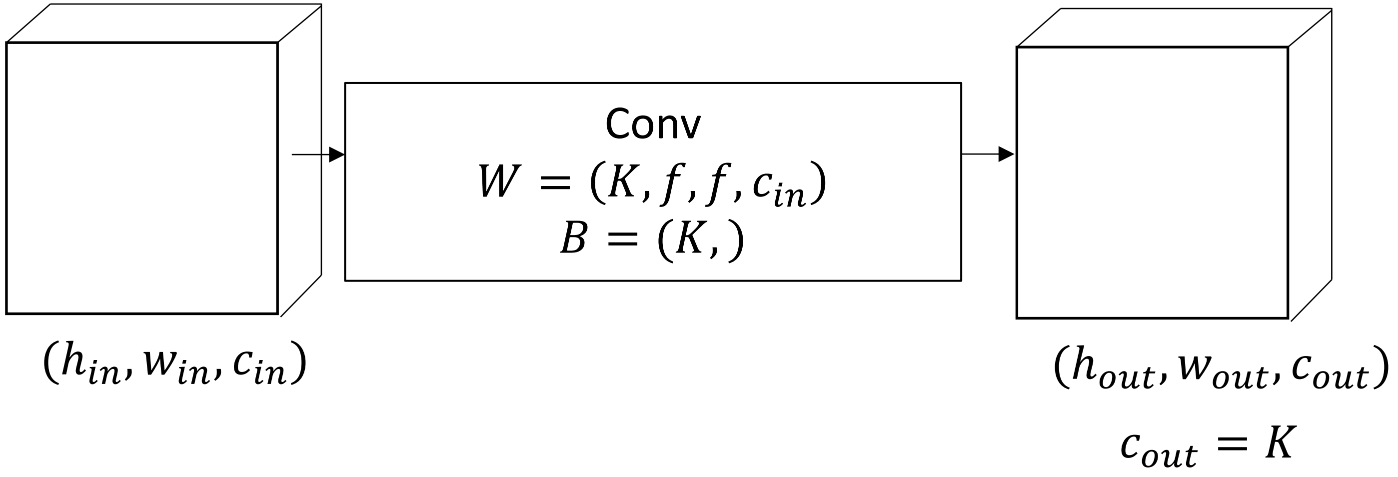
Brad Quinton, Scott Chin

Working towards defining a Convolutional Layer



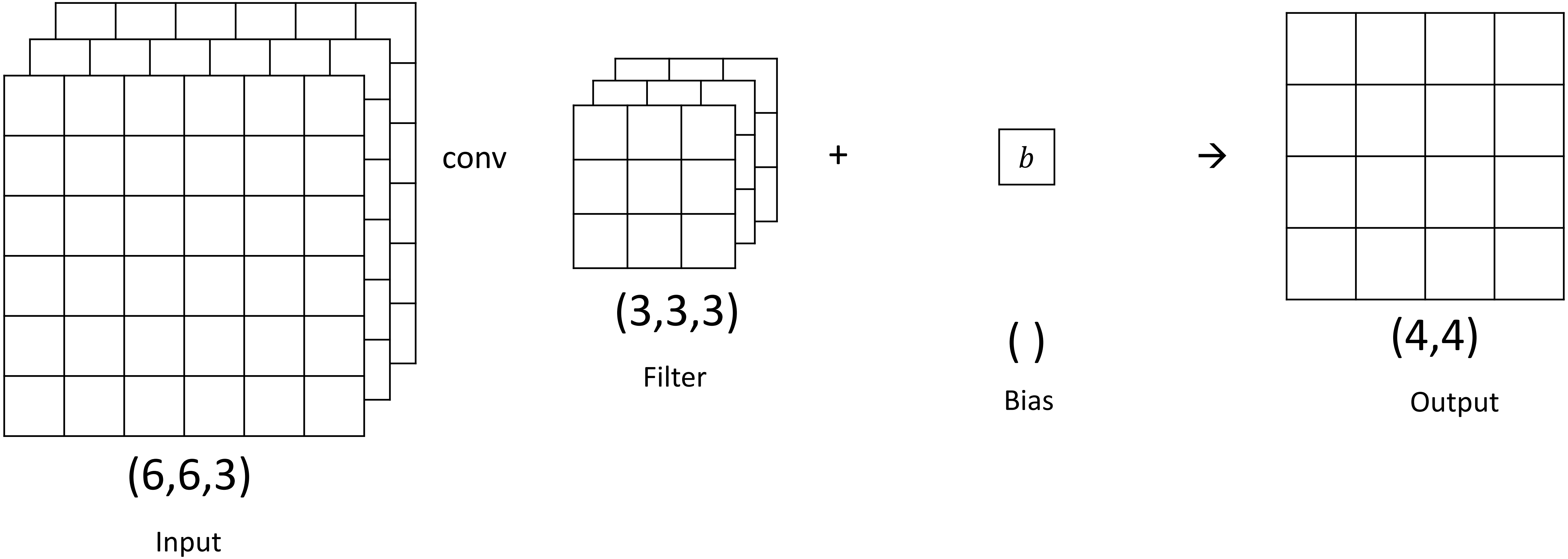
- Collection of filters can be represented as a single (K, f, f, c) weight tensor
- We don't quite have a convolutional layer yet.
- Still need bias parameters and activations

Working towards defining a Convolutional Layer



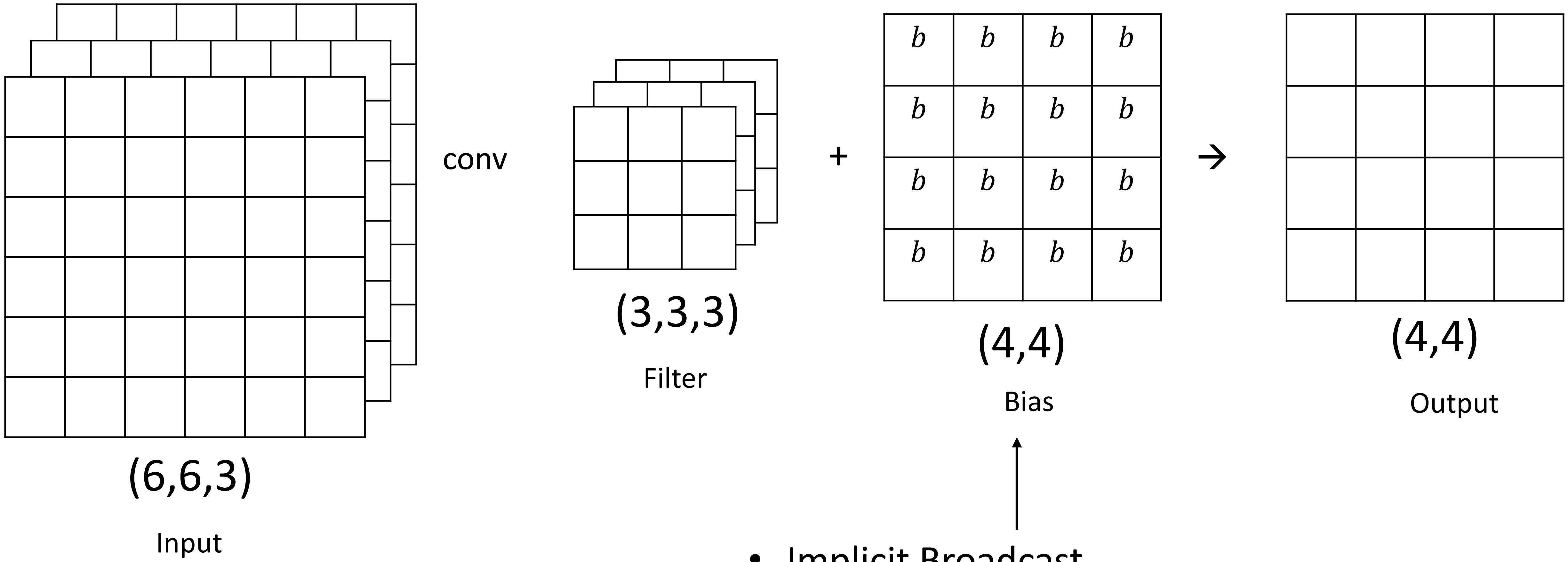
- If we think of each filter as a neuron, then we need one bias per filter

Computation for ONE filter



Brad Quinton, Scott Chin

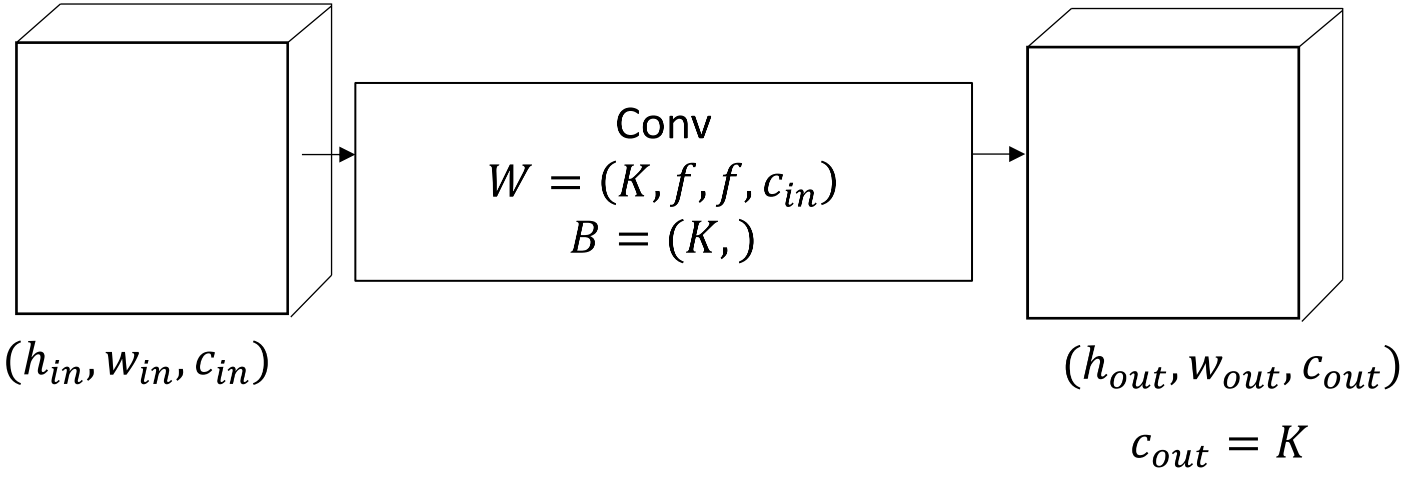
Computation for ONE filter



- Implicit Broadcast.
- There is still only **ONE** bias parameter

Brad Quinton, Scott Chin

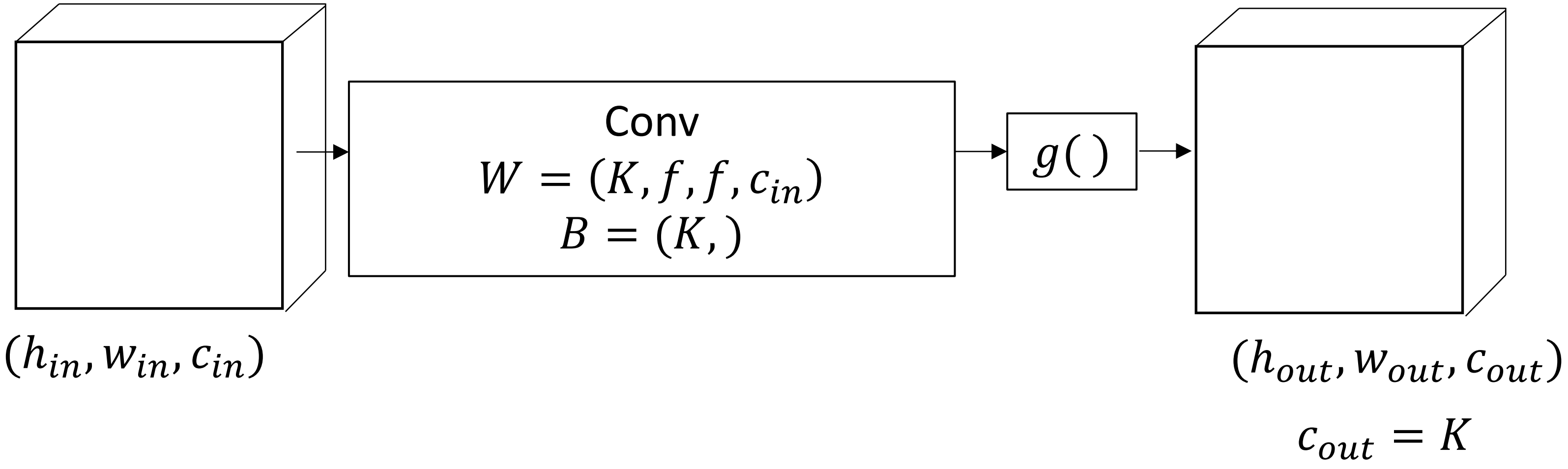
Working towards defining a Convolutional Layer



- So far we've defined the "linear" part of the layer. Similar to the z=WX+B part of the fully-connected layer
- Convolution is a linear operation
- Still need nonlinear activation

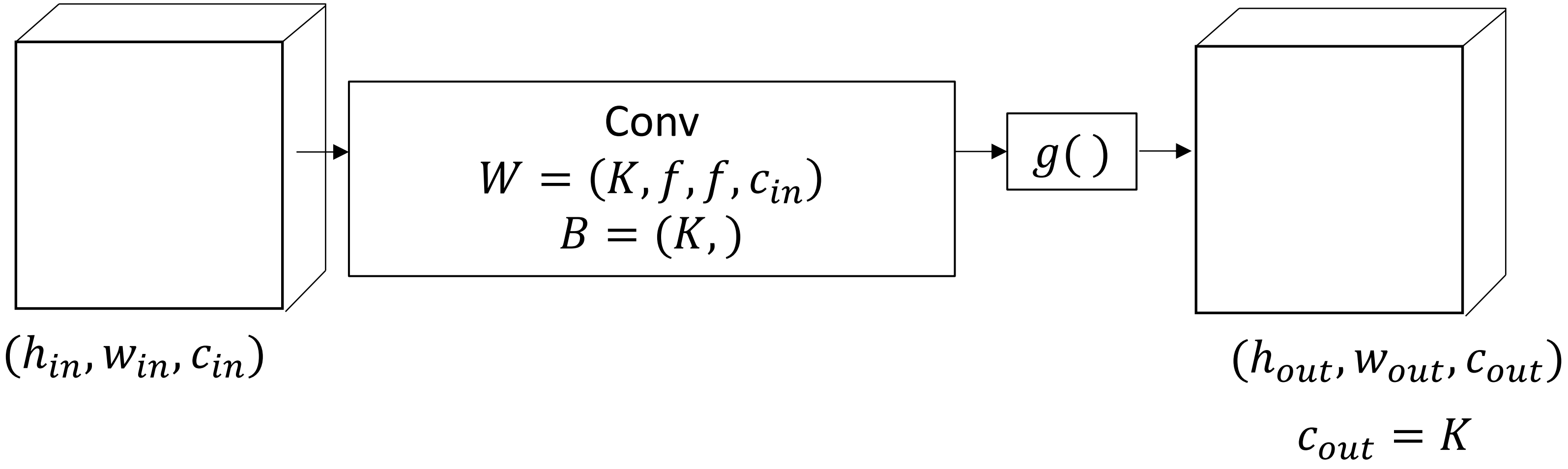
Brad Quinton, Scott Chin

This is a Convolutional Layer

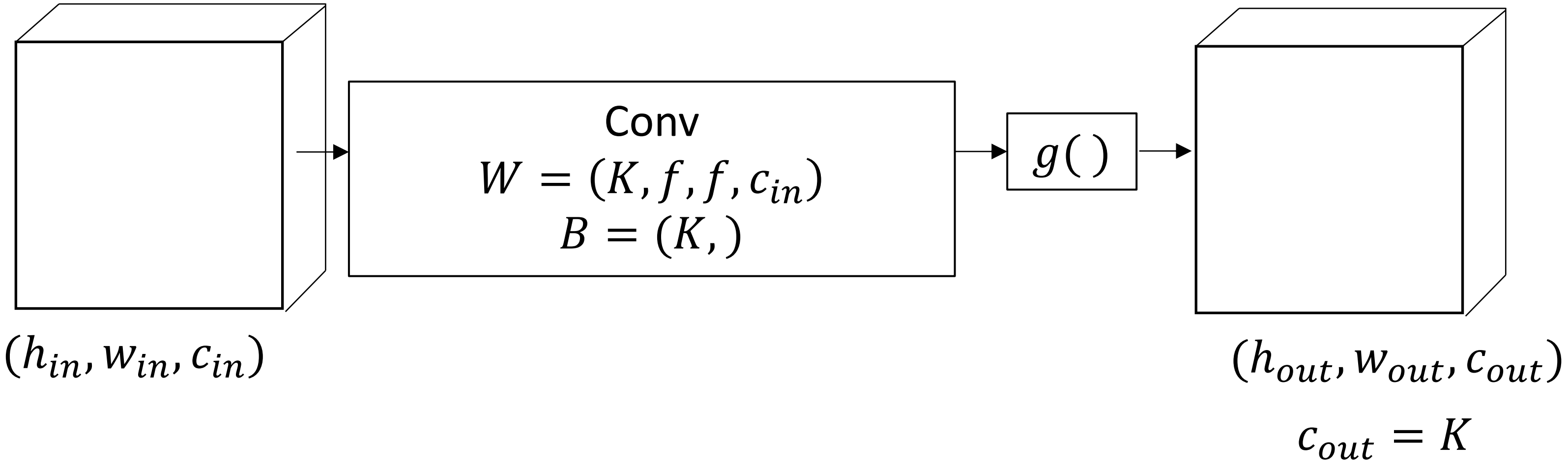


- Remember that activation is applied to each element separately

Number of parameters in a Conv Layer?



Number of parameters in a Conv Layer?

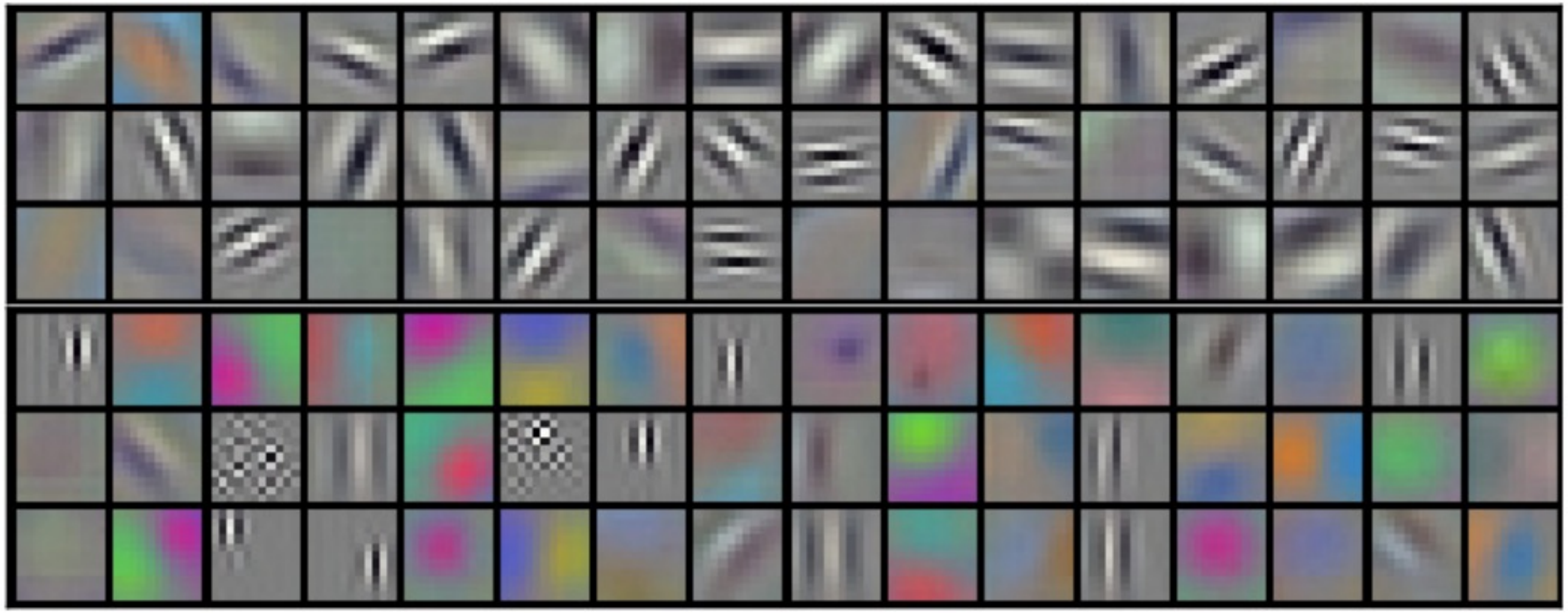


- Weight parameters: $K * f * f * c_{in}$
- Bias parameters: K
- Total: $K(f * f * c_{in} + 1)$

Number of parameters doesn't change if input size changes

Brad Quinton, Scott Chin

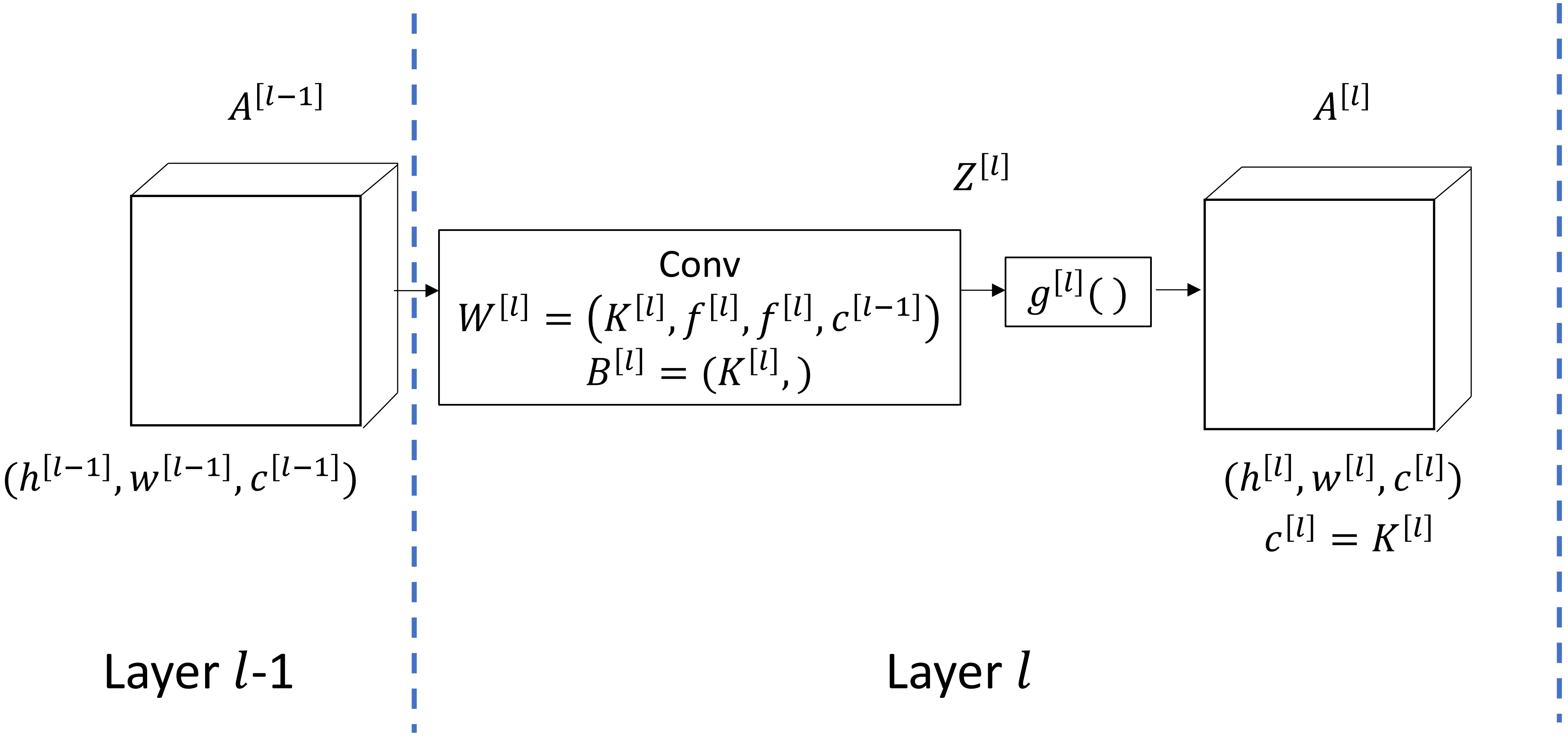
First Layer Filters of AlexNet (11x11)



“ImageNet Classification with Deep Convolutional Neural Networks”, Krizhevsky, Sutskever, Hinton, 2012,
https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

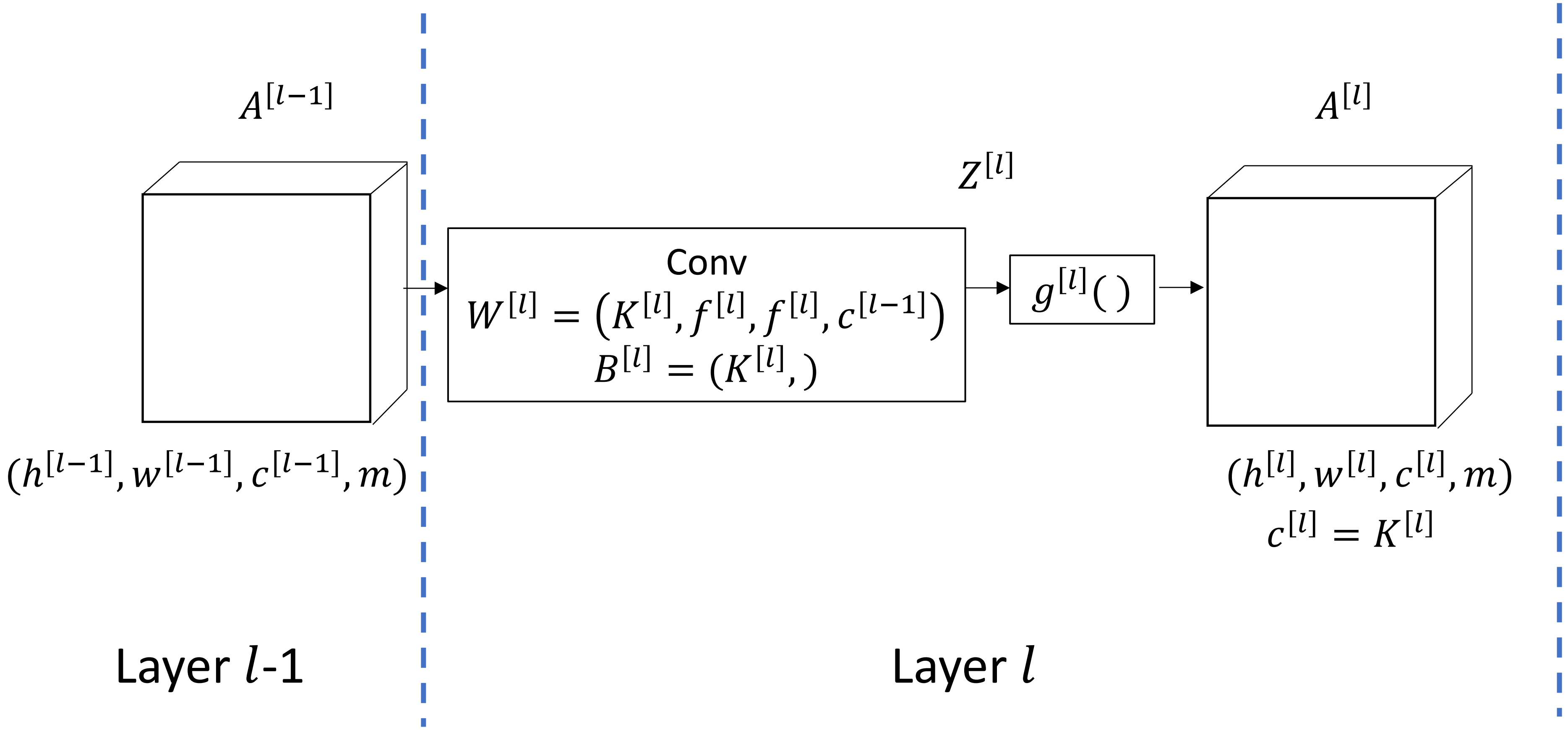
Stacking Convolution Layers

Generalize



- $[l]$ denotes layer l

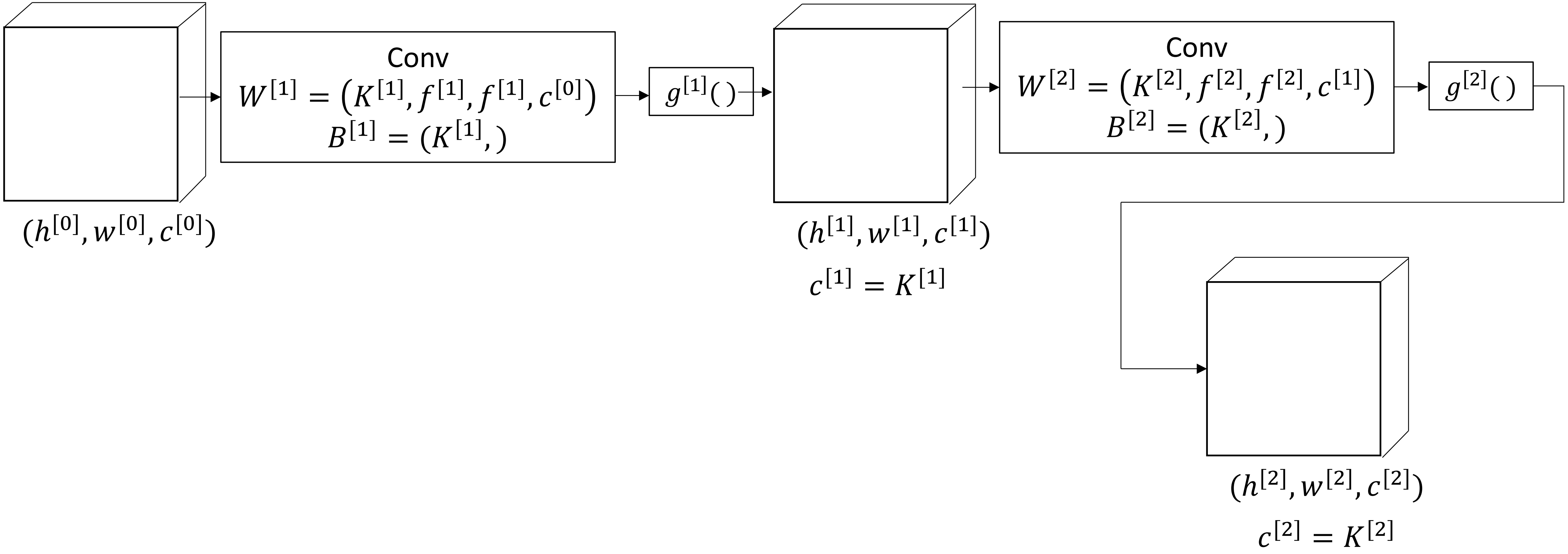
Vectorizing for multiple samples (Next Lecture)



- $[l]$ denotes layer l
- m denotes batch-size

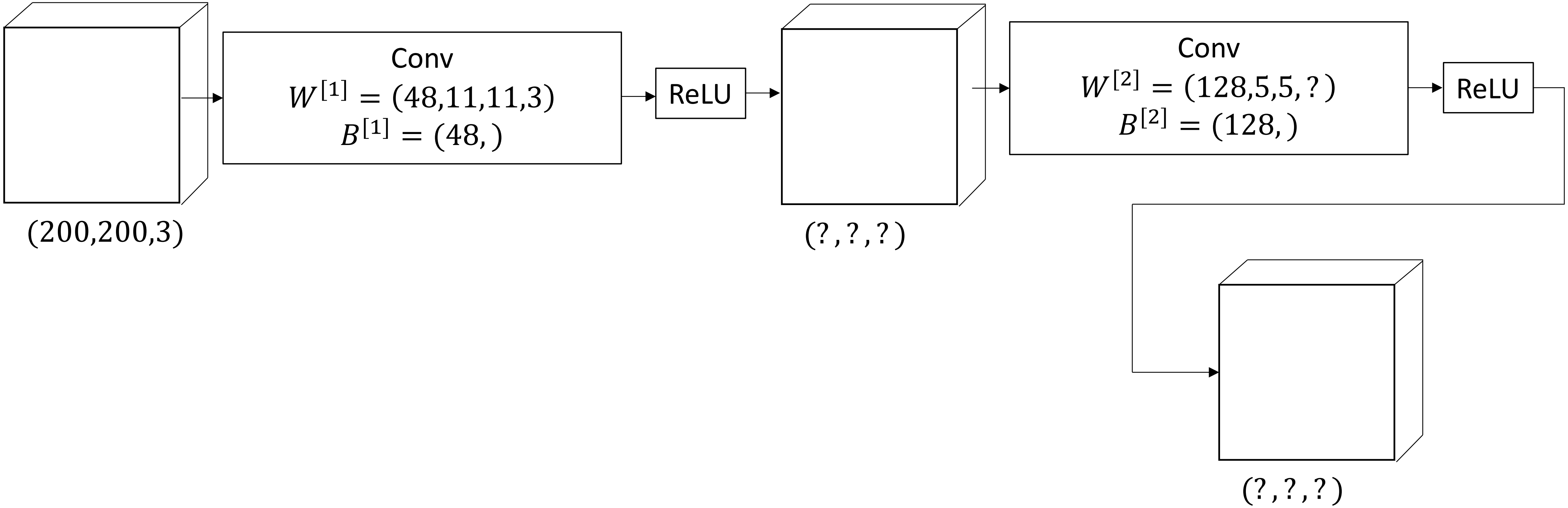
Brad Quinton, Scott Chin

Two Convolutional Layers Stacked

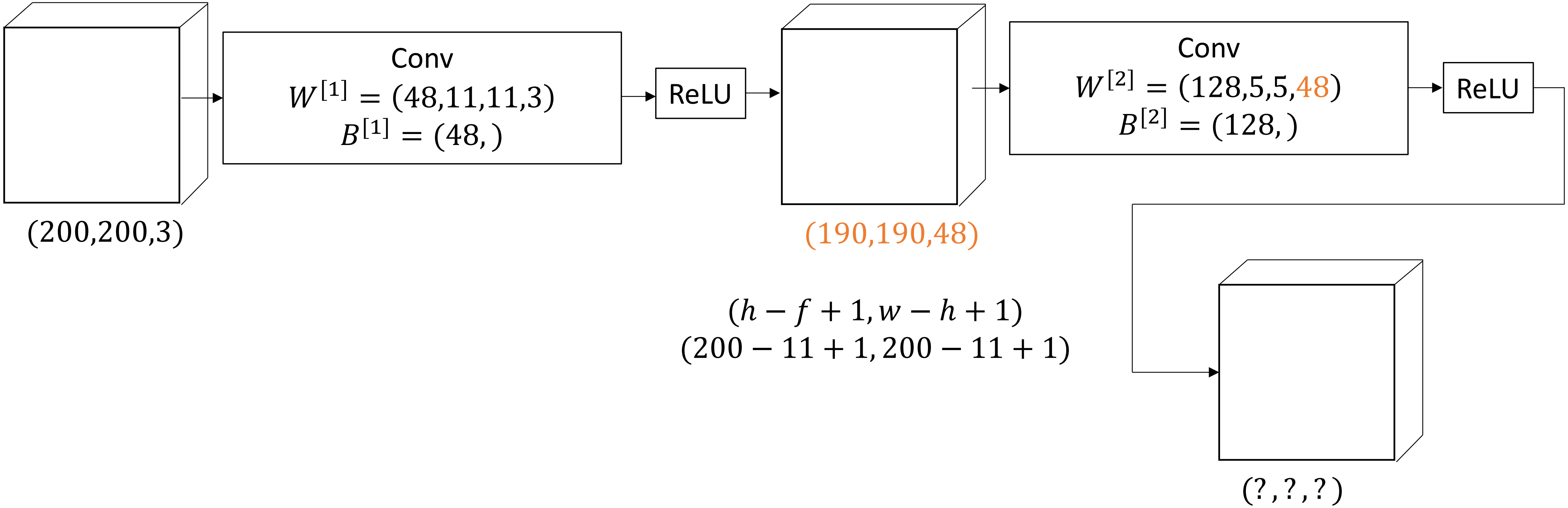


Brad Quinton, Scott Chin

Now with some real numbers

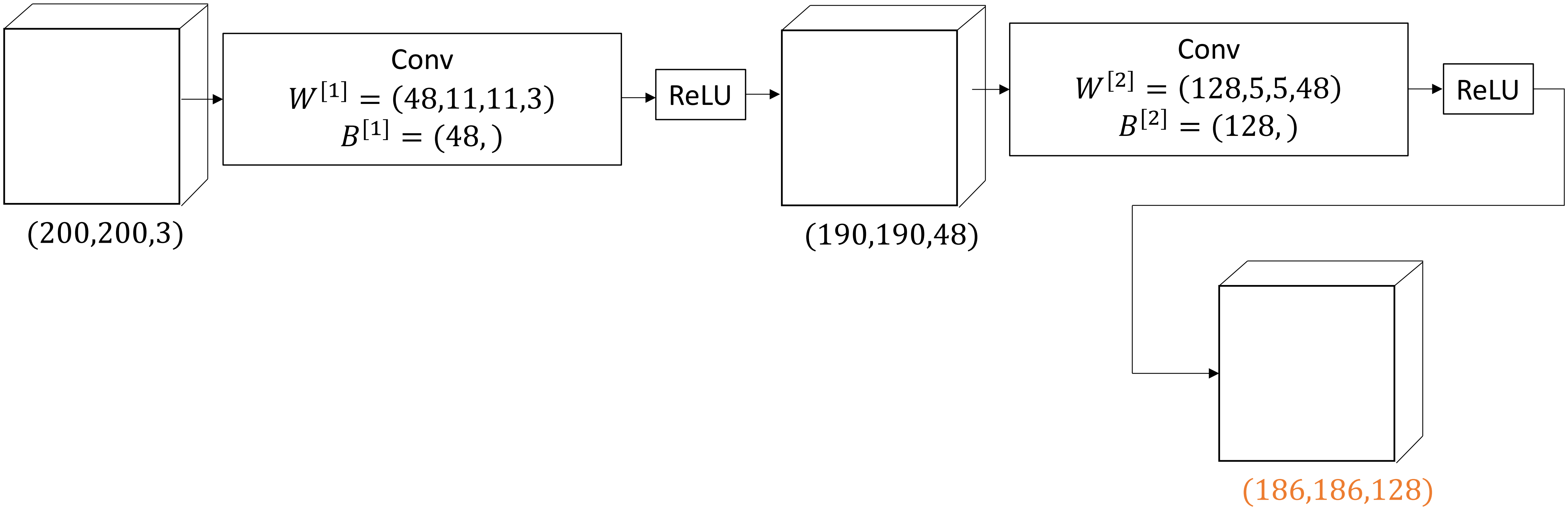


Simple CNN Example with numbers



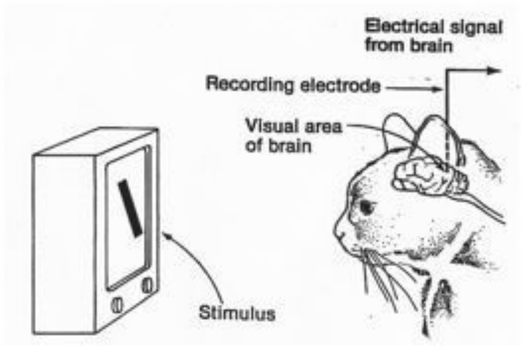
Brad Quinton, Scott Chin

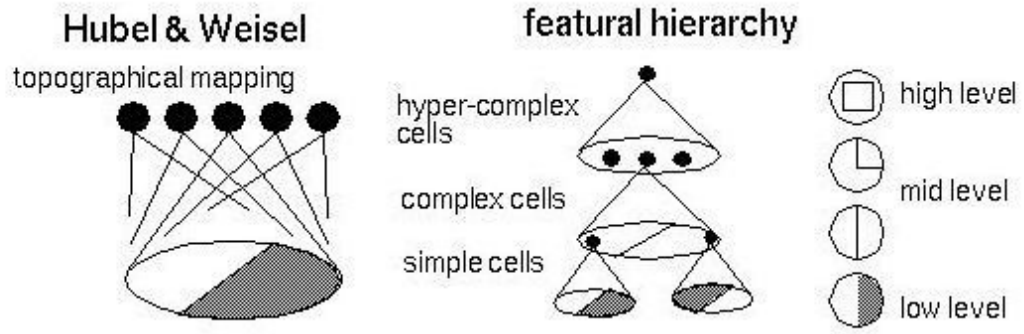
Simple CNN Example with numbers



$$(h - f + 1, w - h + 1)$$
$$(190 - 5 + 1, 190 - 5 + 1)$$

Interpretation of Stacking Convolutions

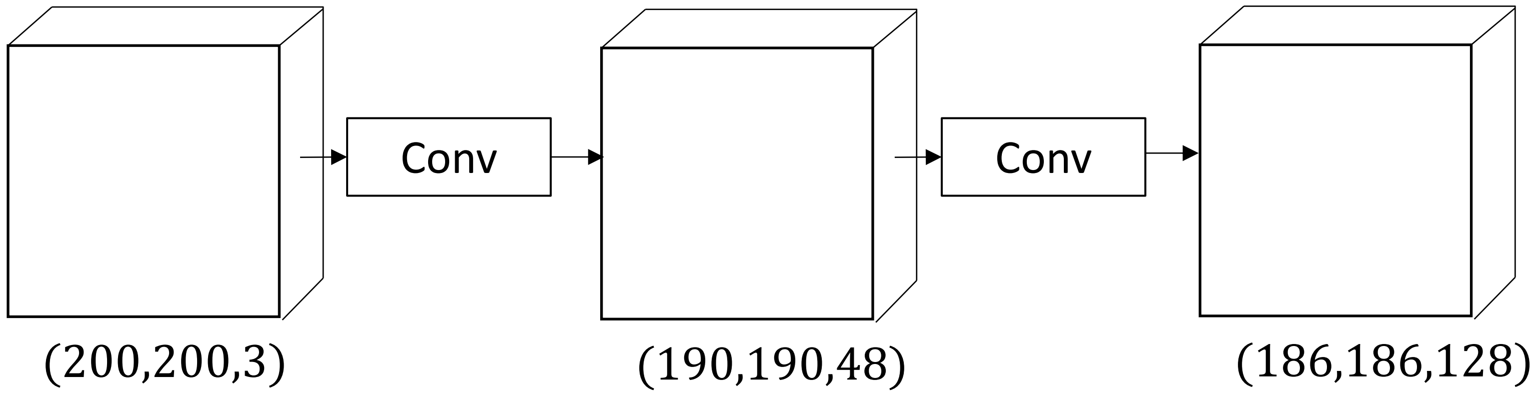




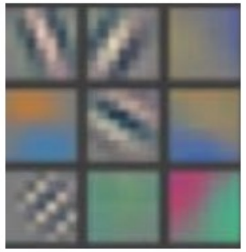
Brad Quinton, Scott Chin

Why do filters look across all channels?

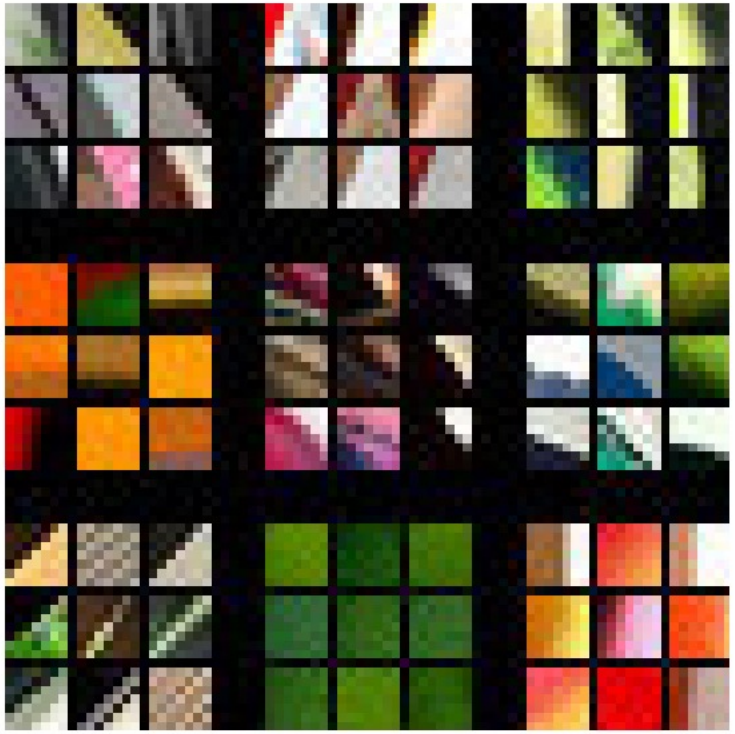
- Beyond the first layer, each channel of a volume is the activation map of a lower level feature
- In order to build filters that look for **compositions** of lower level features, must look at multiple activation maps.



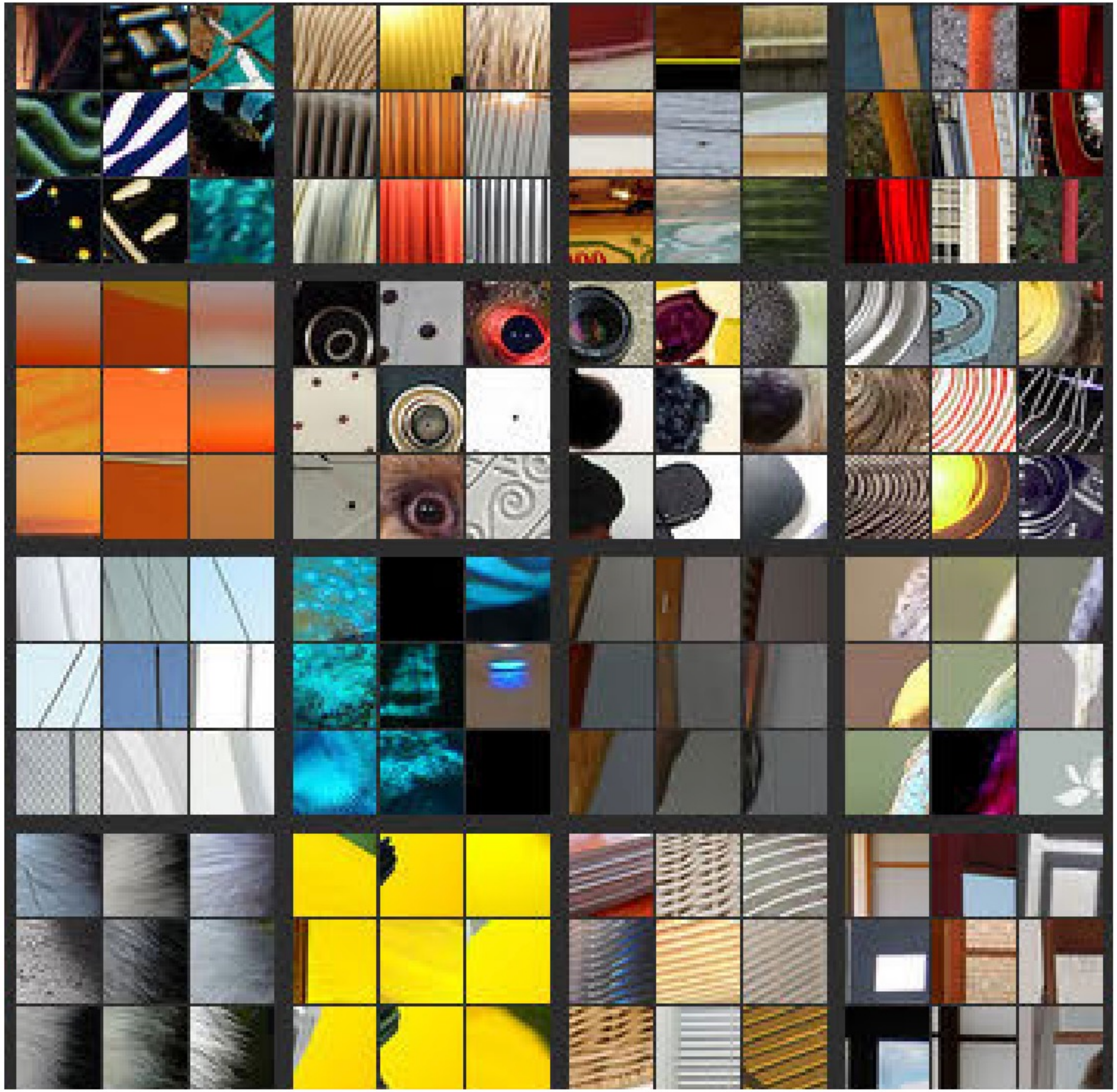
Brad Quinton, Scott Chin

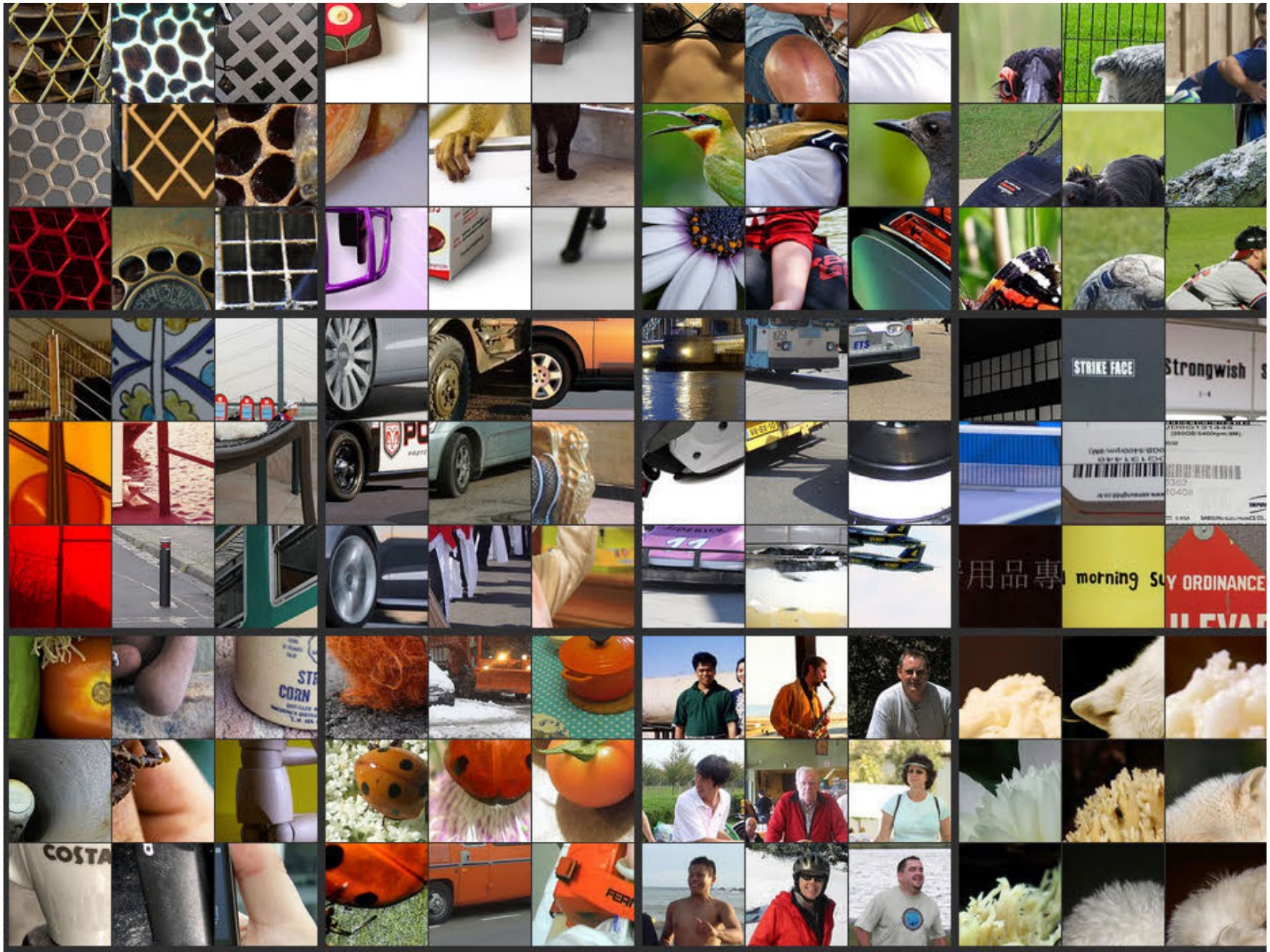


Layer 1

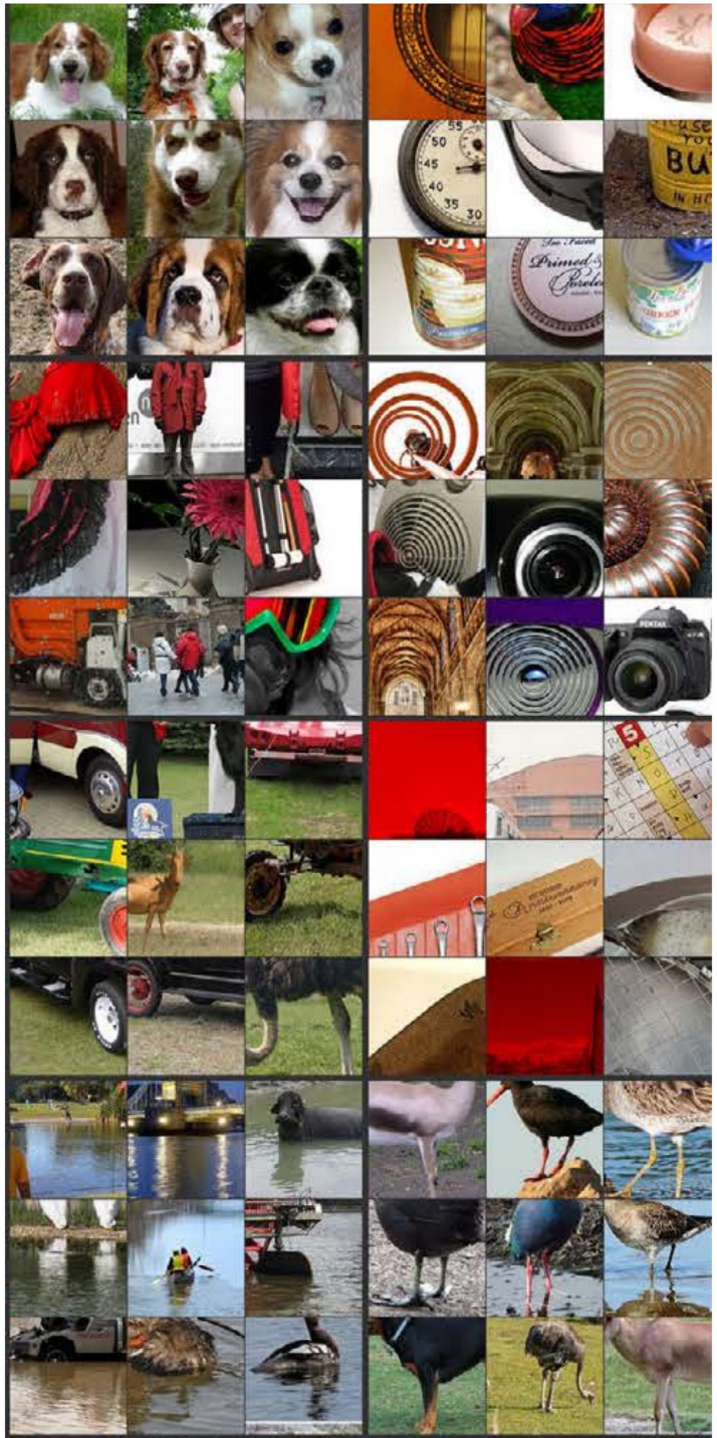


Layer 2





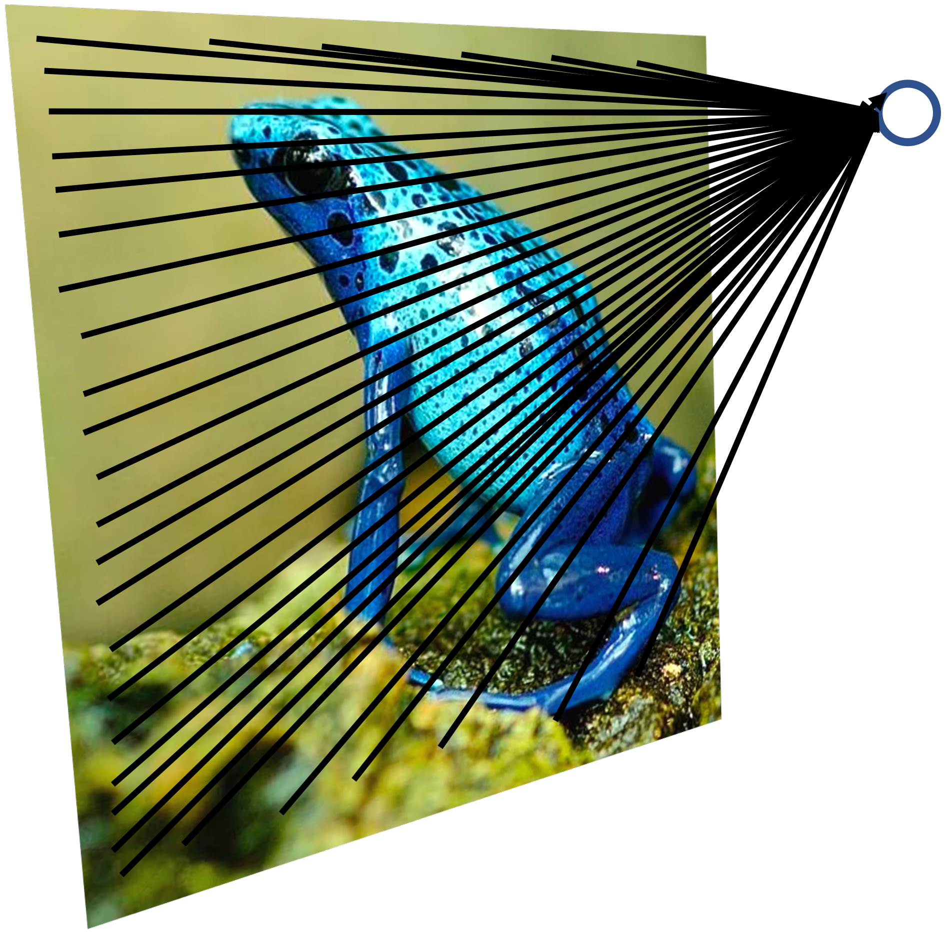
ad Quinton, Scott Chin




BARBER SHOP
Reliant
LINCOLN
WELDERS
#94759
GULF
DOUGHNUTS

Relationship between Convolutional and Fully-Connected Layer

Recall Motivation for Convolutional Layer

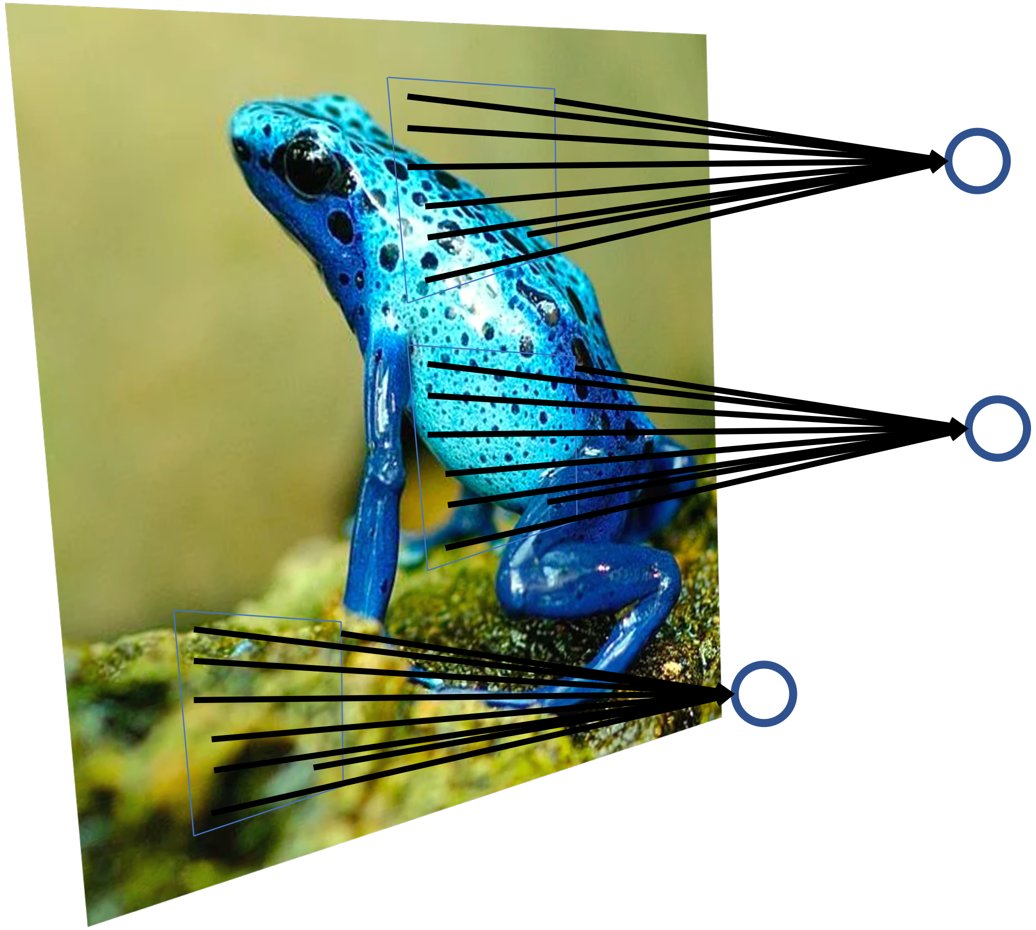


One fully connected neuron shown

- We said that Fully-Connected neural networks leads to a lot of parameters
- And doesn't make use of locality of visual features spatial correlation
- Not making use of locality of visual features spatial correlation

Brad Quinton, Scott Chin

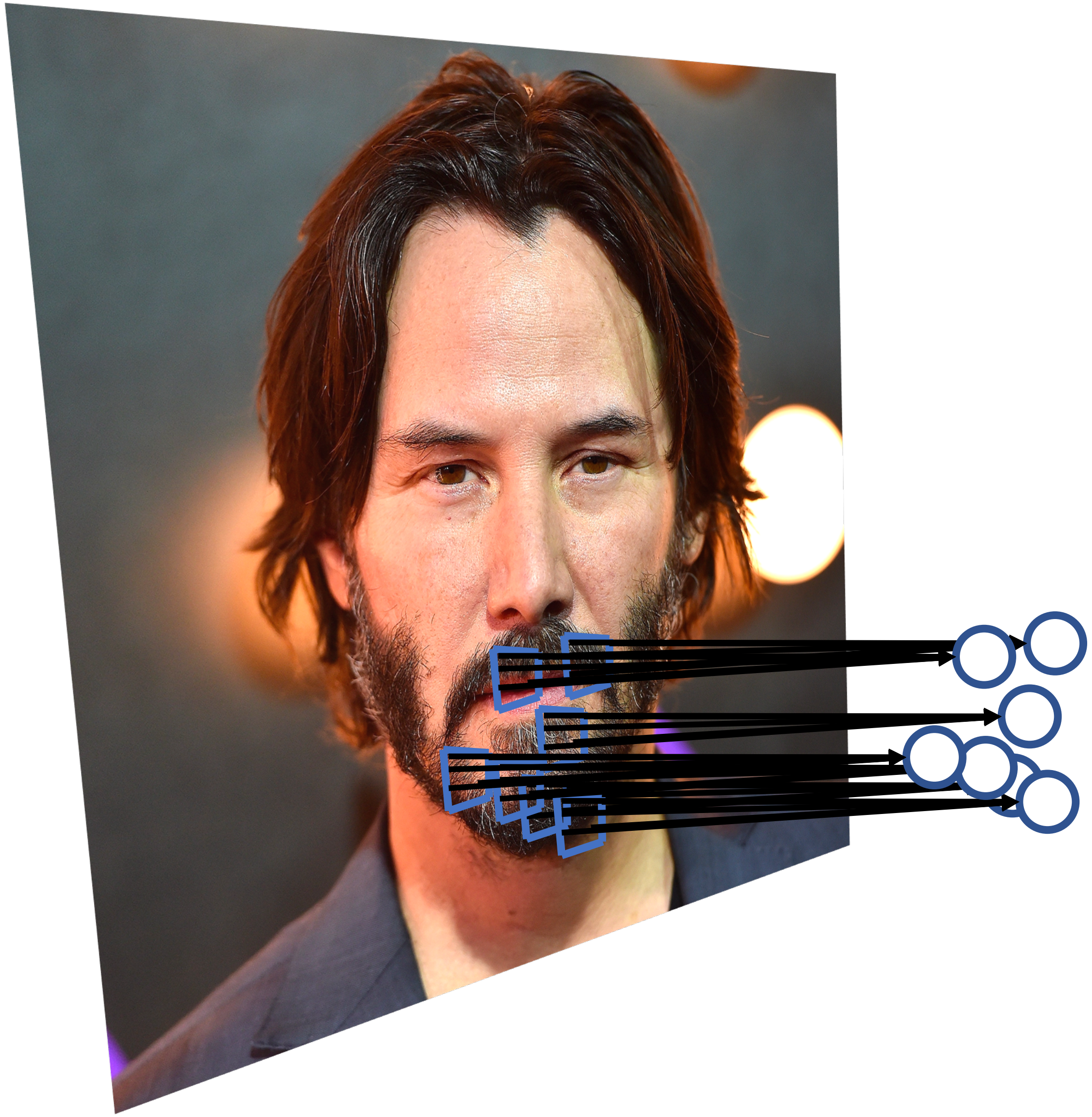
Recall Motivation for Convolutional Layer



- We said that Fully-Connected neural networks leads to a lot of parameters
- And doesn’t make use of locality of visual features spatial correlation
- We then said that maybe we could solve these problems using some sort of well thought-out sparsely-connected organization

Brad Quinton, Scott Chin

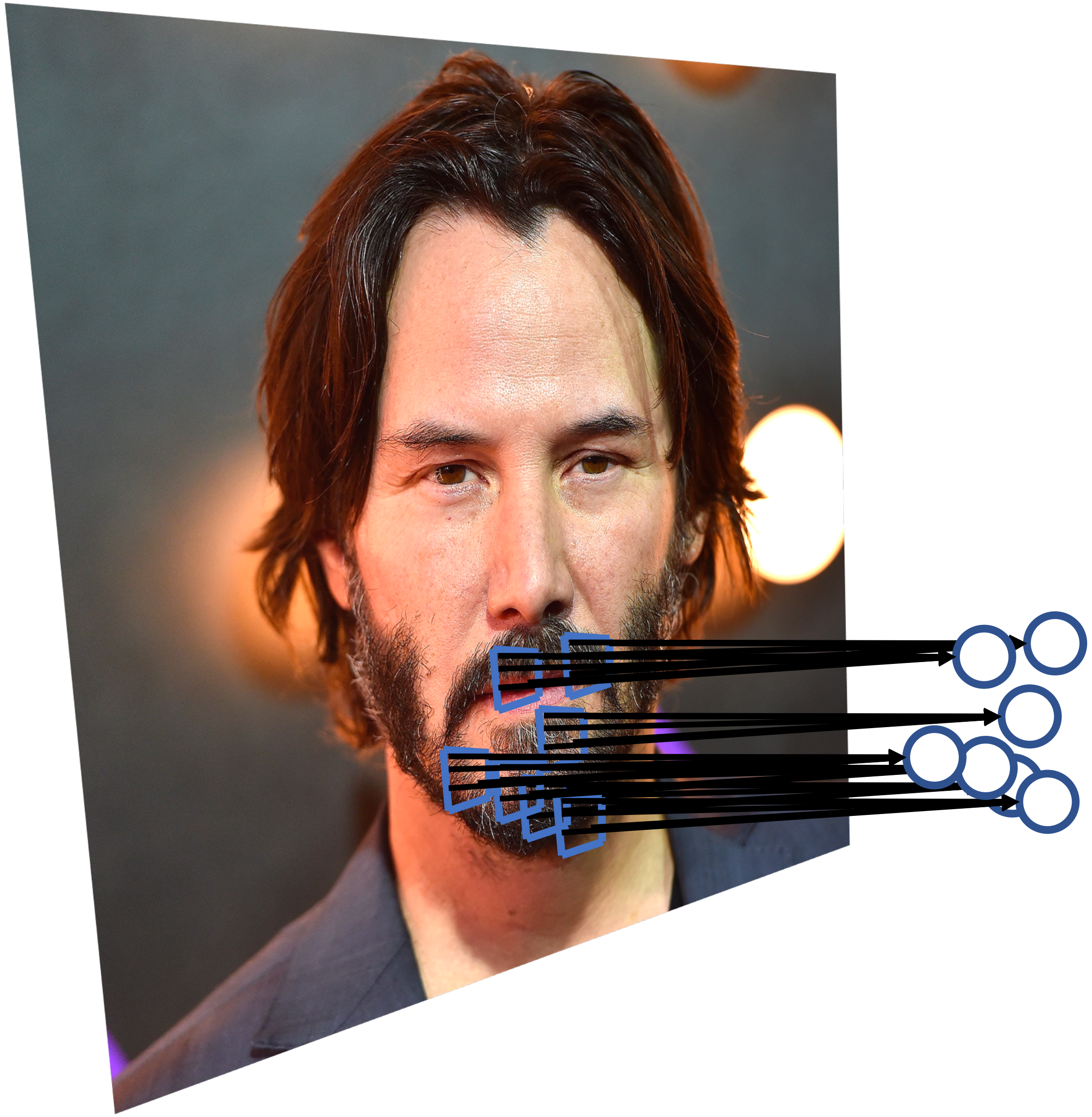
Recall Motivation for Conv Layer



- We said that even for only localized connections, we would have redundant neurons
- And that maybe we could solve this with parameter sharing

Brad Quinton, Scott Chin

Recall Motivation for Conv Layer



- We said that even for only localized connections, we would have redundant neurons
- And that maybe we could solve this with parameter sharing

Note: For all these problems, given a fully-connected network with enough capacity, and a large enough training data set, we could get around these problems. But you'd need a massive network and a, likely, infeasibly large data set. CNNs are much much more efficient

Nothing magical about Convolutional Layers

- Convolution layer basically let us achieve sparse connectivity between layers while also taking advantage of spatial structure of image data to allow parameter sharing!
- Although it employs a lot of inspiration and intuition from our understanding of human vision, it is mathematically just a sparsely connected version of the fully connected layer with parameter sharing.
- Let's illustrate with an example how we transform a fully-connected layer to a convolutional layer

Example for one filter:

$$\underset{3\text{x}3}{\begin{array}{|c|c|c|}\hline x_1 & x_2 & x_3 \\ \hline x_4 & x_5 & x_6 \\ \hline x_7 & x_8 & x_9 \\ \hline \end{array}} \quad \text{conv} \quad \underset{2\text{x}2}{\begin{array}{|c|c|}\hline w_1 & w_2 \\ \hline w_3 & w_4 \\ \hline \end{array}} \quad + \quad \begin{array}{|c|}\hline b_1 \\ \hline \end{array} \quad = \quad \underset{2\text{x}2}{\begin{array}{|c|c|}\hline z_1 & z_2 \\ \hline z_3 & z_4 \\ \hline \end{array}}$$

Example for one filter:

$$\begin{array}{|c|c|c|}\hline x_1 & x_2 & x_3 \\ \hline x_4 & x_5 & x_6 \\ \hline x_7 & x_8 & x_9 \\ \hline \end{array} \quad \text{conv} \quad \begin{array}{|c|c|}\hline w_1 & w_2 \\ \hline w_3 & w_4 \\ \hline \end{array} \quad + \quad \begin{array}{|c|}\hline b_1 \\ \hline \end{array} \quad = \quad \begin{array}{|c|c|}\hline z_1 & z_2 \\ \hline z_3 & z_4 \\ \hline \end{array}$$

3x3 2x2 2x2

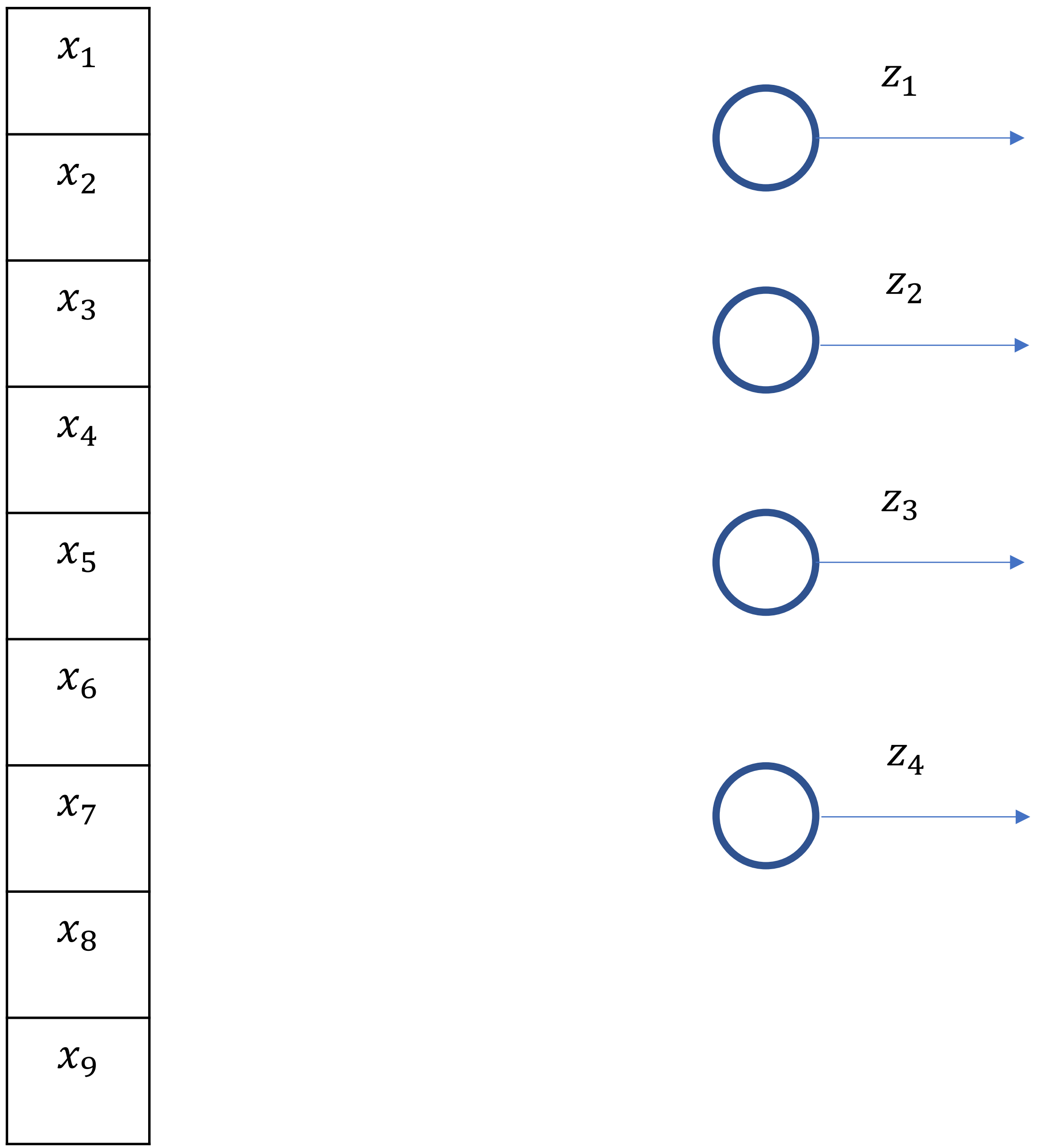
$$z_1 = w_1 x_1 + w_2 x_2 + w_3 x_4 + w_4 x_5 + b_1$$

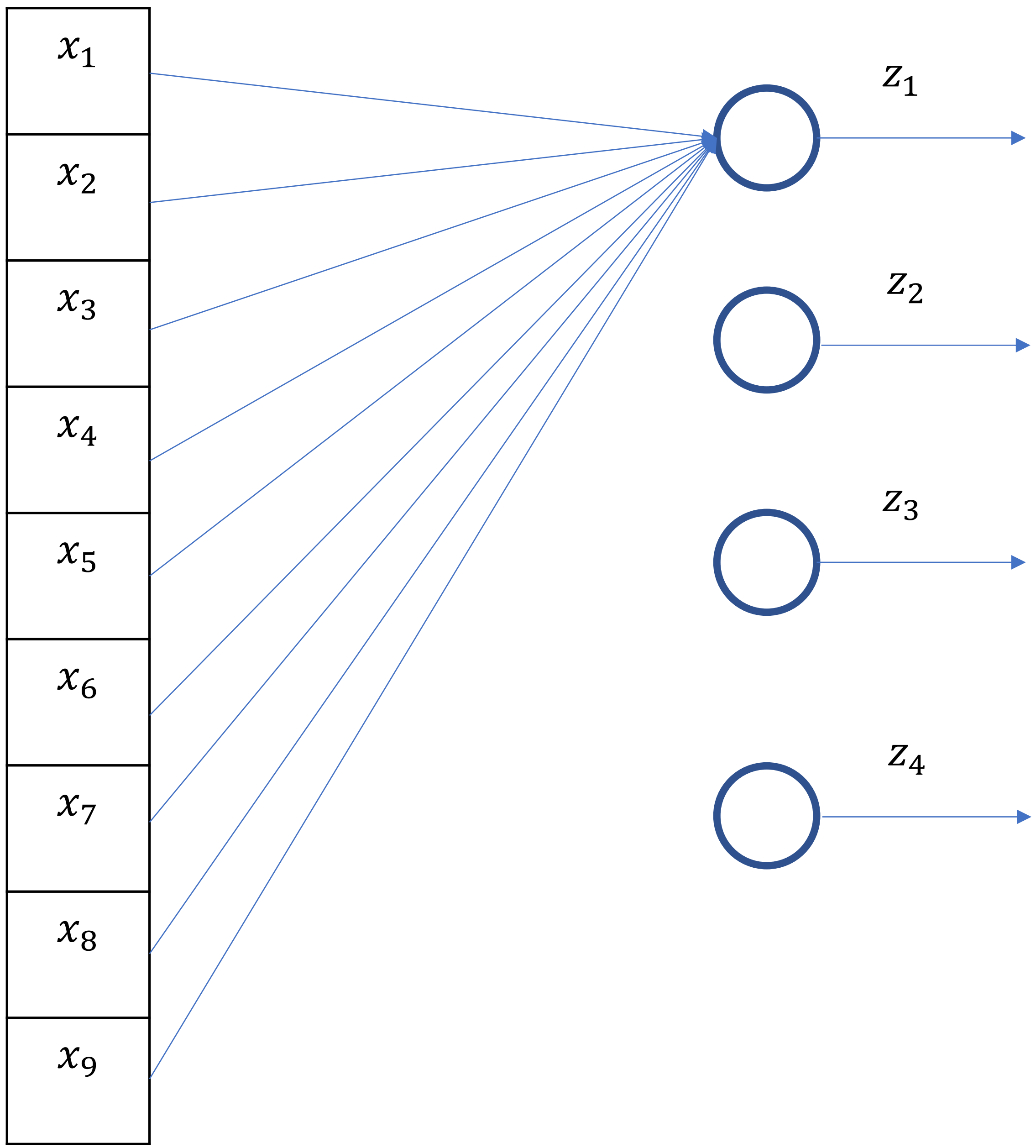
$$z_2 = w_1 x_2 + w_2 x_3 + w_3 x_5 + w_4 x_6 + b_1$$

$$z_3 = w_1 x_4 + w_2 x_5 + w_3 x_7 + w_4 x_8 + b_1$$

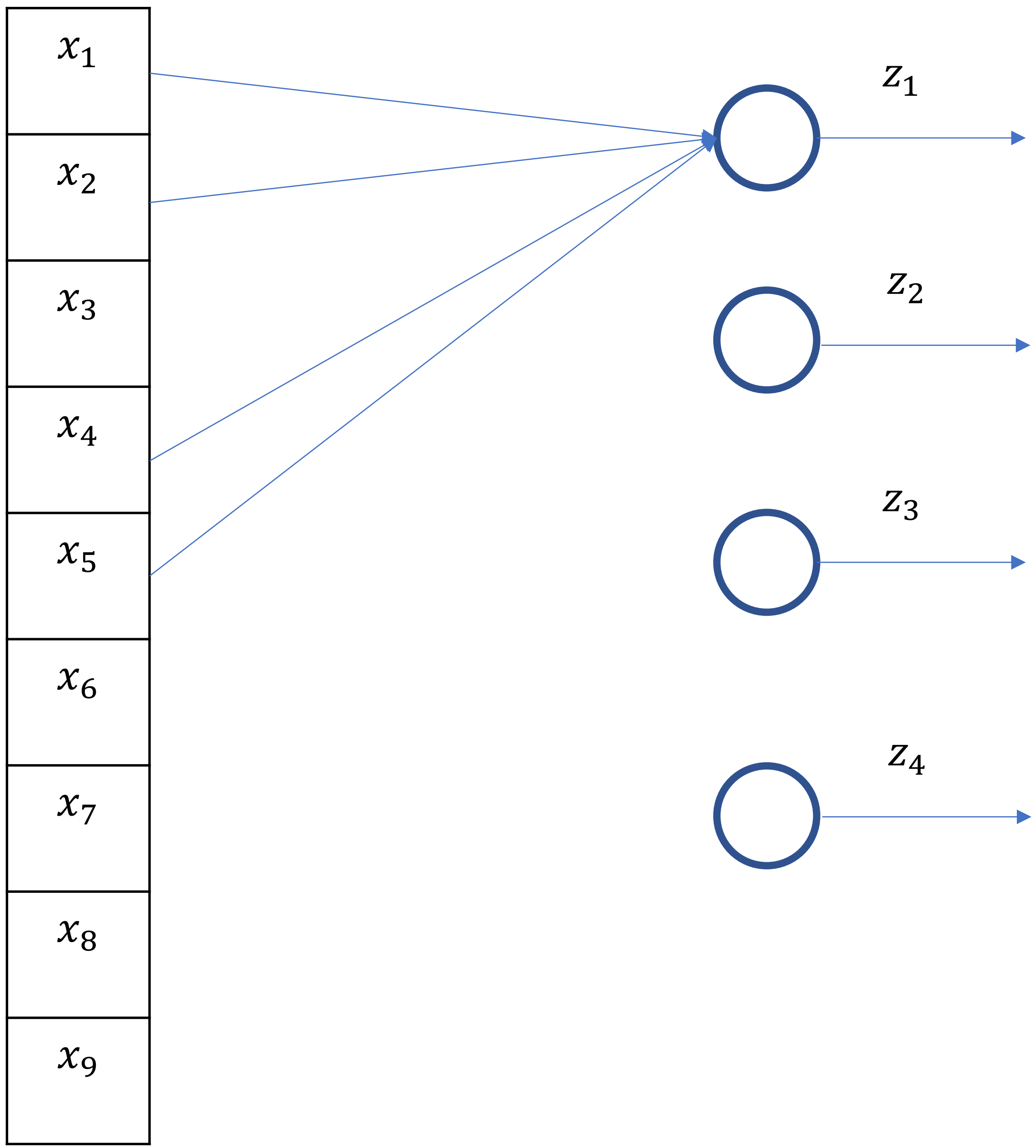
$$z_4 = w_1 x_5 + w_2 x_6 + w_3 x_8 + w_4 x_9 + b_1$$

- Consider a Fully Connected Layer with 4 units





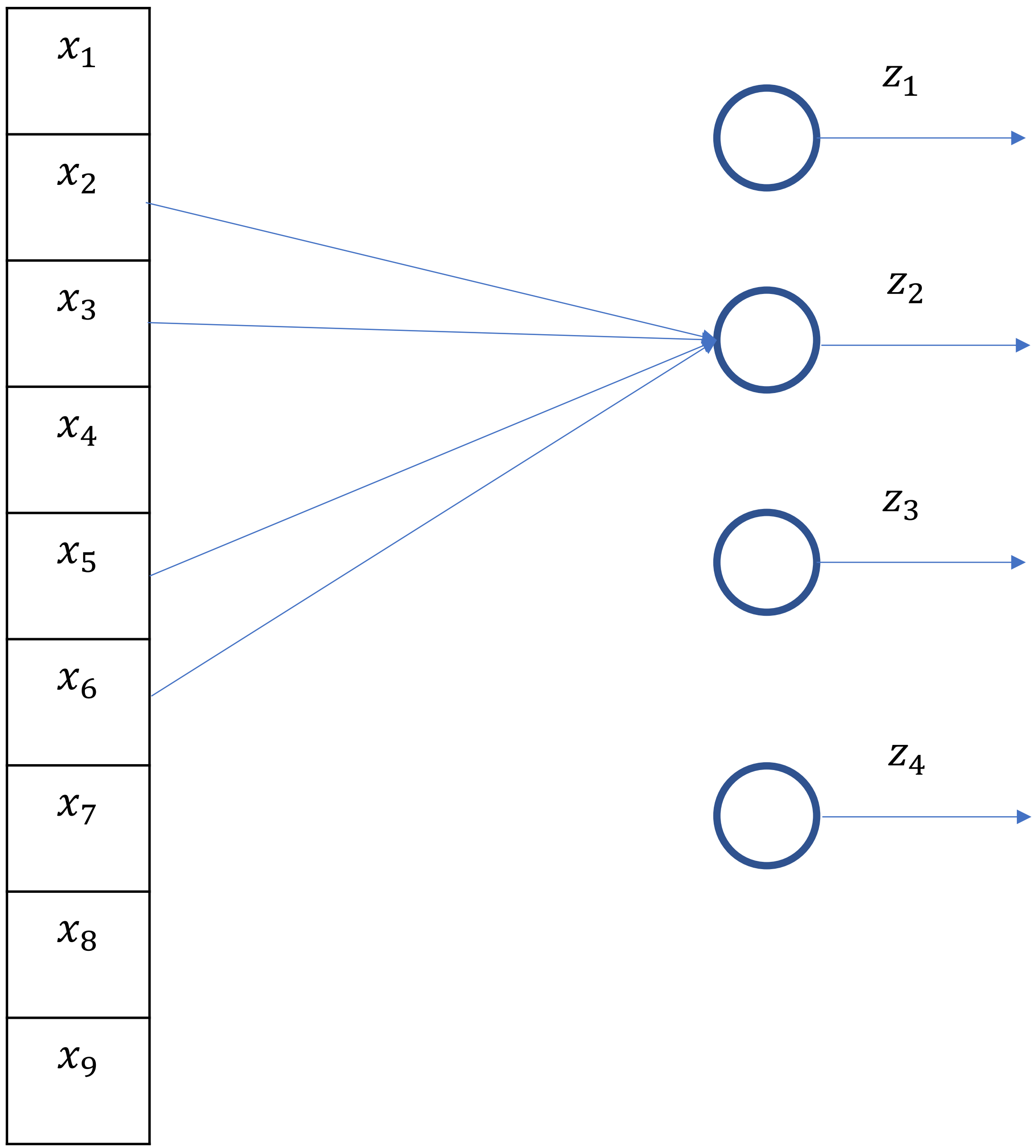
- Consider a Fully Connected Layer with 4 units
- Each unit would have a "connection" and weight for each input to the layer



- Consider a Fully Connected Layer with 4 units
- Each unit would have a “connection” and weight for each input to the layer

Sparsely Connected

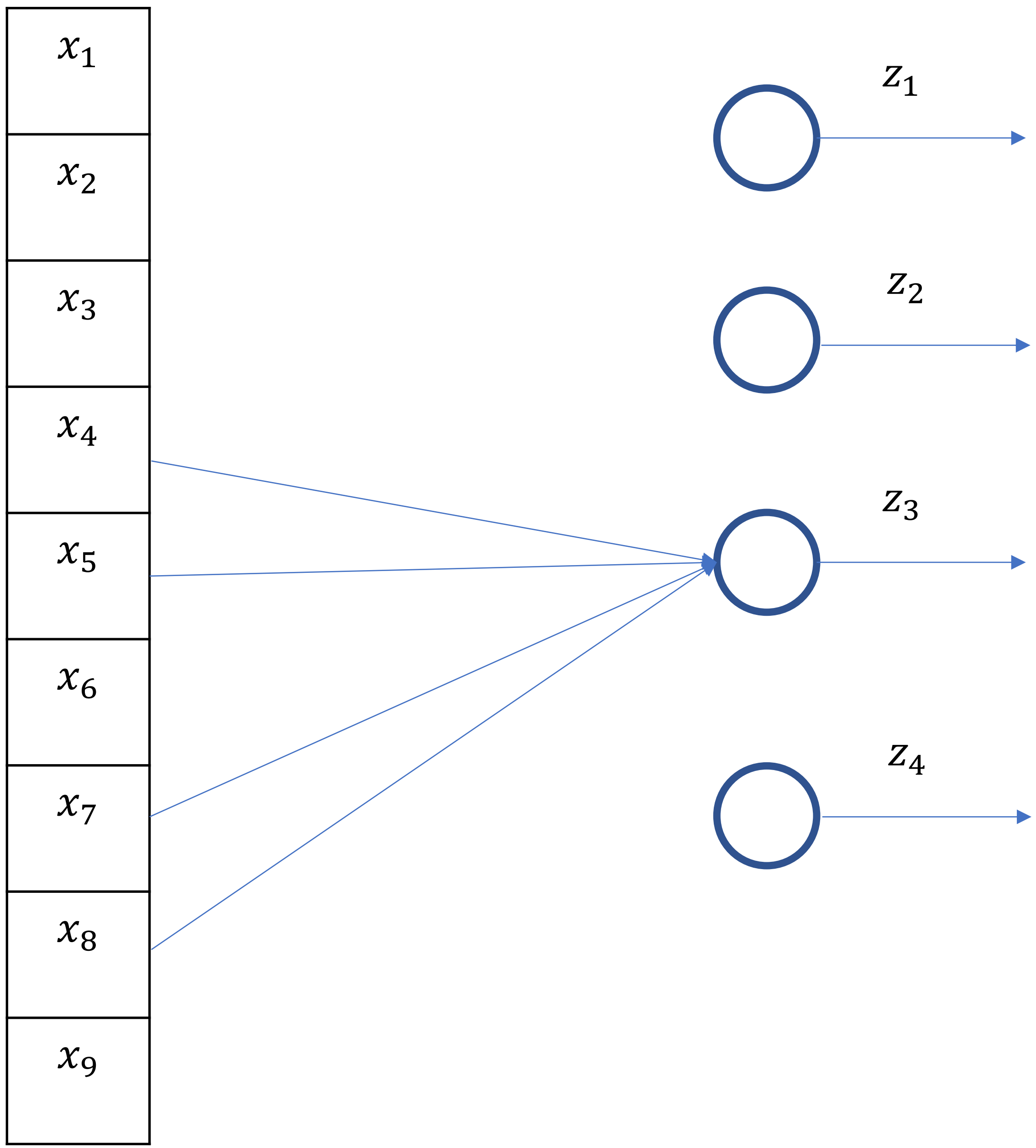
- Now let’s get rid of some of these connections (in this case, instead of 9 connections, only 4)



- Consider a Fully Connected Layer with 4 units
- Each unit would have a "connection" and weight for each input to the layer

Sparsely Connected

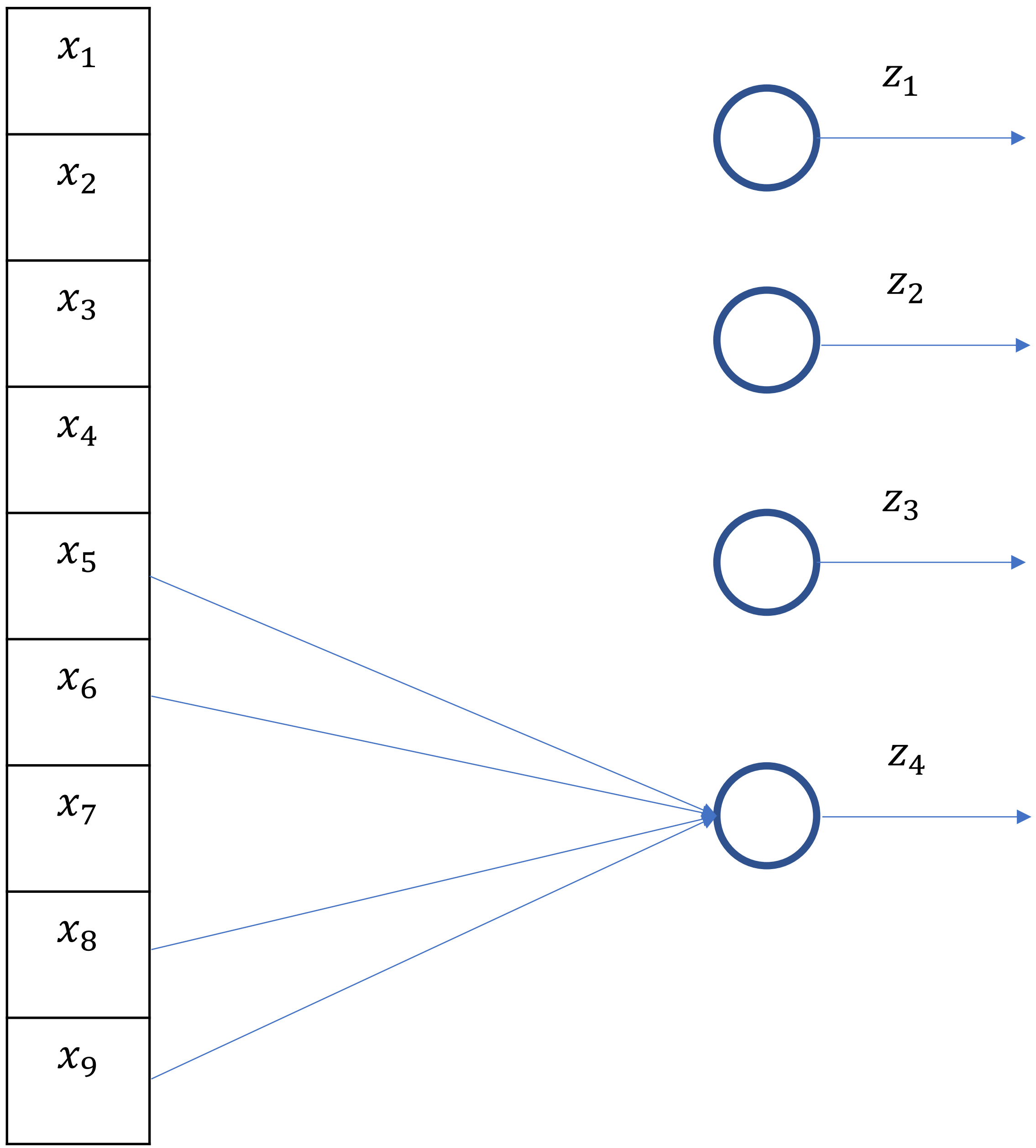
- Now let's get rid of some of these connections (in this case, instead of 9 connections, only 4)
- Each neuron is connected to a different subset of the inputs



- Consider a Fully Connected Layer with 4 units
- Each unit would have a “connection” and weight for each input to the layer

Sparsely Connected

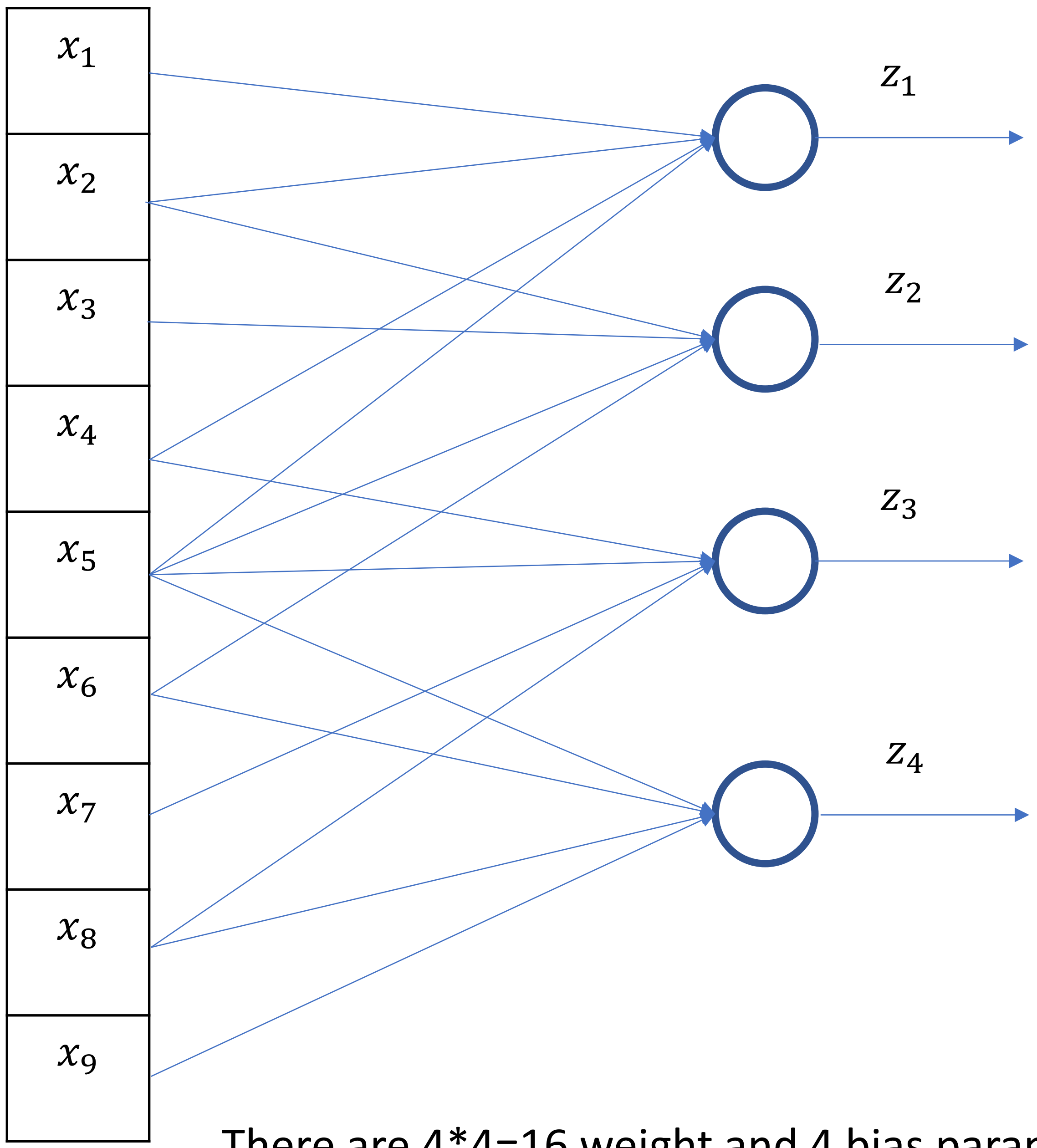
- Now let’s get rid of some of these connections (in this case, instead of 9 connections, only 4)
- Each neuron is connected to a different subset of the inputs



- Consider a Fully Connected Layer with 4 units
- Each unit would have a "connection" and weight for each input to the layer

Sparsely Connected

- Now let's get rid of some of these connections (in this case, instead of 9 connections, only 4)
- Each neuron is connected to a different subset of the inputs

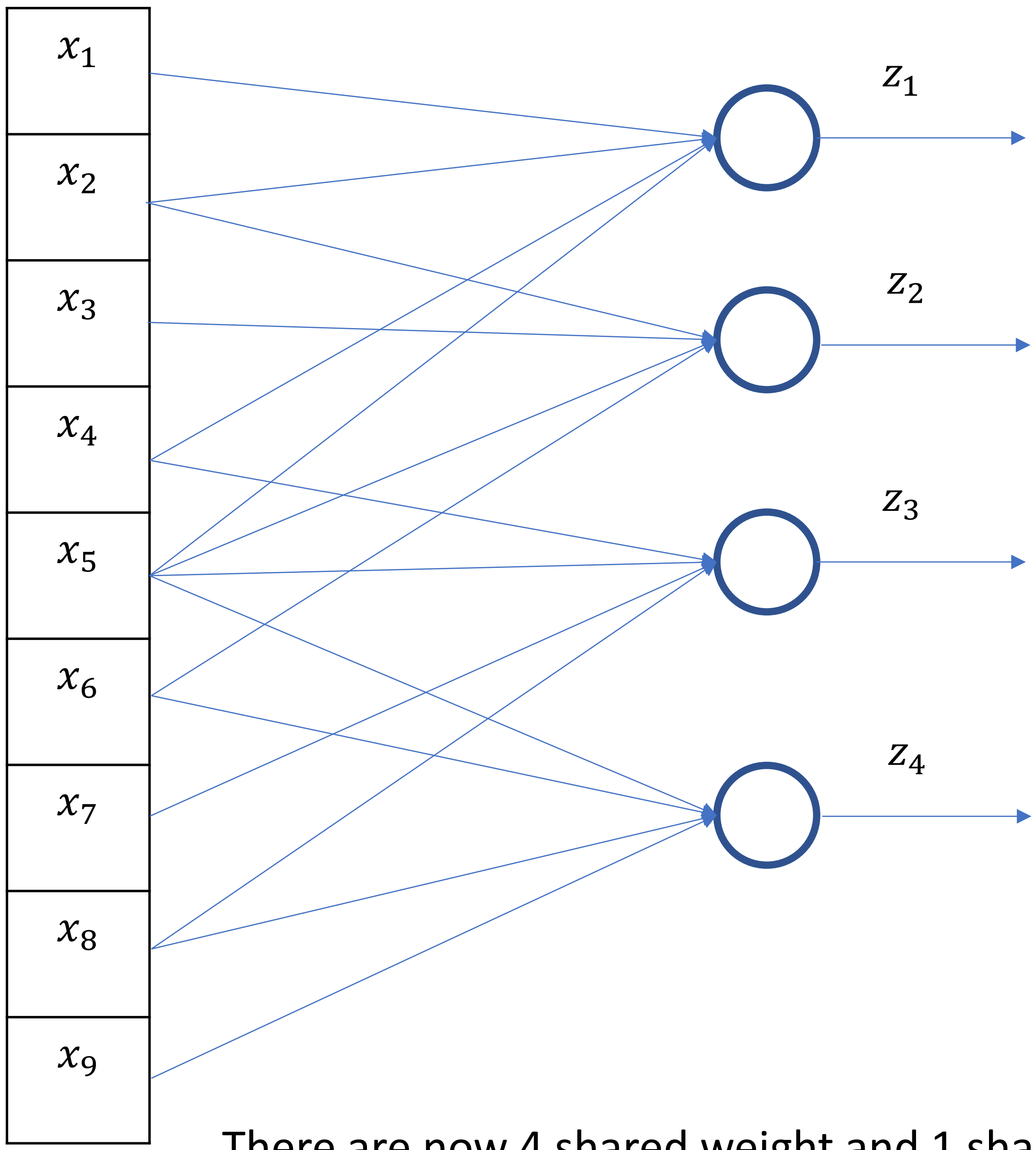


- Consider a Fully Connected Layer with 4 units
- Each unit would have a "connection" and weight for each input to the layer

Sparsely Connected

- Now let's get rid of some of these connections (in this case, instead of 9 connections, only 4)
- Each neuron is connected to a different subset of the inputs

There are 4*4=16 weight and 4 bias parameters



- Consider a Fully Connected Layer with 4 units
- Each unit would have a "connection" and weight for each input to the layer

Sparsely Connected

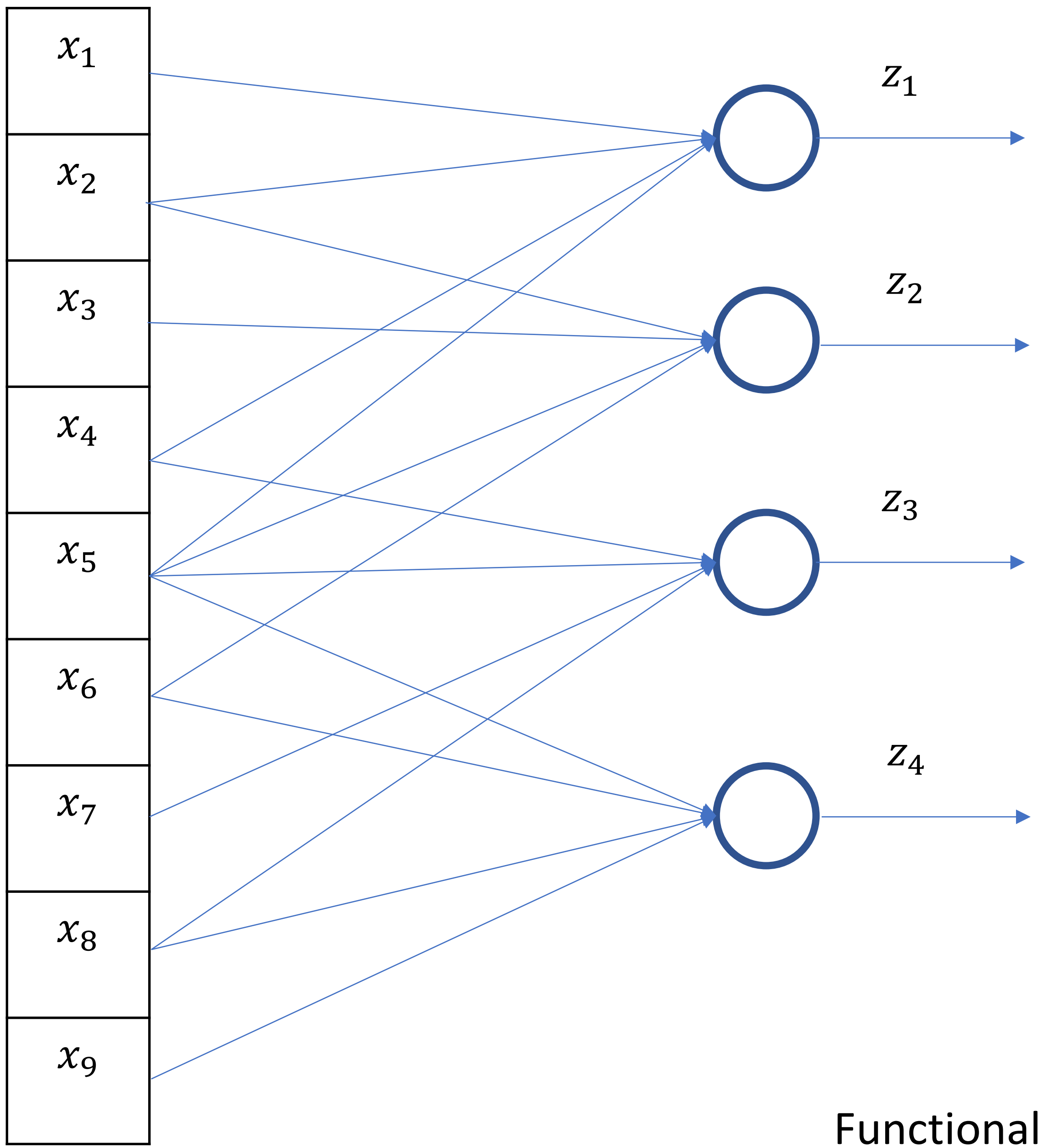
- Now let's get rid of some of these connections (in this case, instead of 9 connections, only 4)
- Each neuron is connected to a different subset of the inputs

Parameter Sharing

- Instead of each neuron having it's own 4 weight and 1 bias parameters, they all share the same 4 weights and 1 bias parameter

There are now 4 shared weight and 1 shared bias parameters

Brad Quinton, Scott Chin



- Consider a Fully Connected Layer with 4 units
- Each unit would have a "connection" and weight for each input to the layer

Sparsely Connected

- Now let's get rid of some of these connections (in this case, instead of 9 connections, only 4)
- Each neuron is connected to a different subset of the inputs

Parameter Sharing

- Instead of each neuron having it's own 4 weight and 1 bias parameters, they all share the same 4 weights and 1 bias parameter

$$z_1 = w_1 x_1 + w_2 x_2 + w_3 x_4 + w_4 x_5 + b_1$$
$$z_2 = w_1 x_2 + w_2 x_3 + w_3 x_5 + w_4 x_6 + b_1$$
$$z_3 = w_1 x_4 + w_2 x_5 + w_3 x_7 + w_4 x_8 + b_1$$
$$z_4 = w_1 x_5 + w_2 x_6 + w_3 x_8 + w_4 x_9 + b_1$$

Functionally/Mathematically equivalent to our convolutional layer

Brad Quinton, Scott Chin

Example for more than one filter filter:

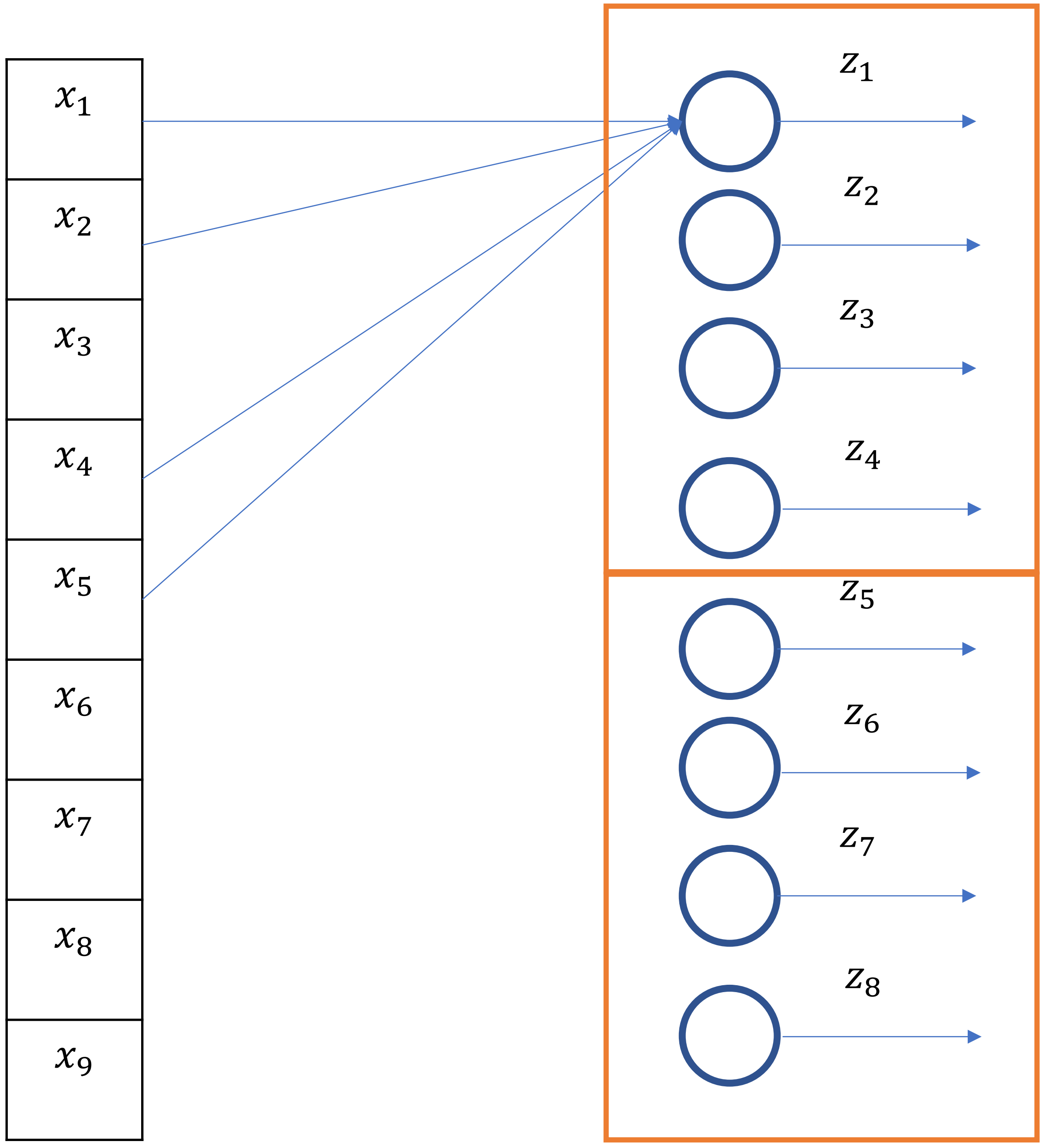
$$\underset{\text{3x3}}{\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix}} \ \text{conv} \ \overset{\text{2x2}}{\begin{bmatrix} w_1 & w_2 \\ w_3 & w_4 \end{bmatrix}} + \boxed{b_1} = \overset{\text{2x2}}{\begin{bmatrix} z_1 & z_2 \\ z_3 & z_4 \end{bmatrix}}$$

Example for more than one filter filter:

$$\begin{array}{c}\begin{bmatrix}x_1 & x_2 & x_3\\ x_4 & x_5 & x_6\\ x_7 & x_8 & x_9\end{bmatrix}\\ \text{3x3}\end{array} \ \text{conv}\ \begin{array}{c}\text{2x2}\\ \begin{bmatrix}w_1 & w_2\\ w_3 & w_4\end{bmatrix}\end{array} + \begin{bmatrix}b_1\end{bmatrix} = \begin{array}{c}\text{2x2}\\ \begin{bmatrix}z_1 & z_2\\ z_3 & z_4\end{bmatrix}\end{array}$$

$$\begin{bmatrix}w_5 & w_6\\ w_7 & w_8\end{bmatrix} + \begin{bmatrix}b_2\end{bmatrix} = \begin{bmatrix}z_5 & z_6\\ z_7 & z_8\end{bmatrix}$$

Brad Quinton, Scott Chin



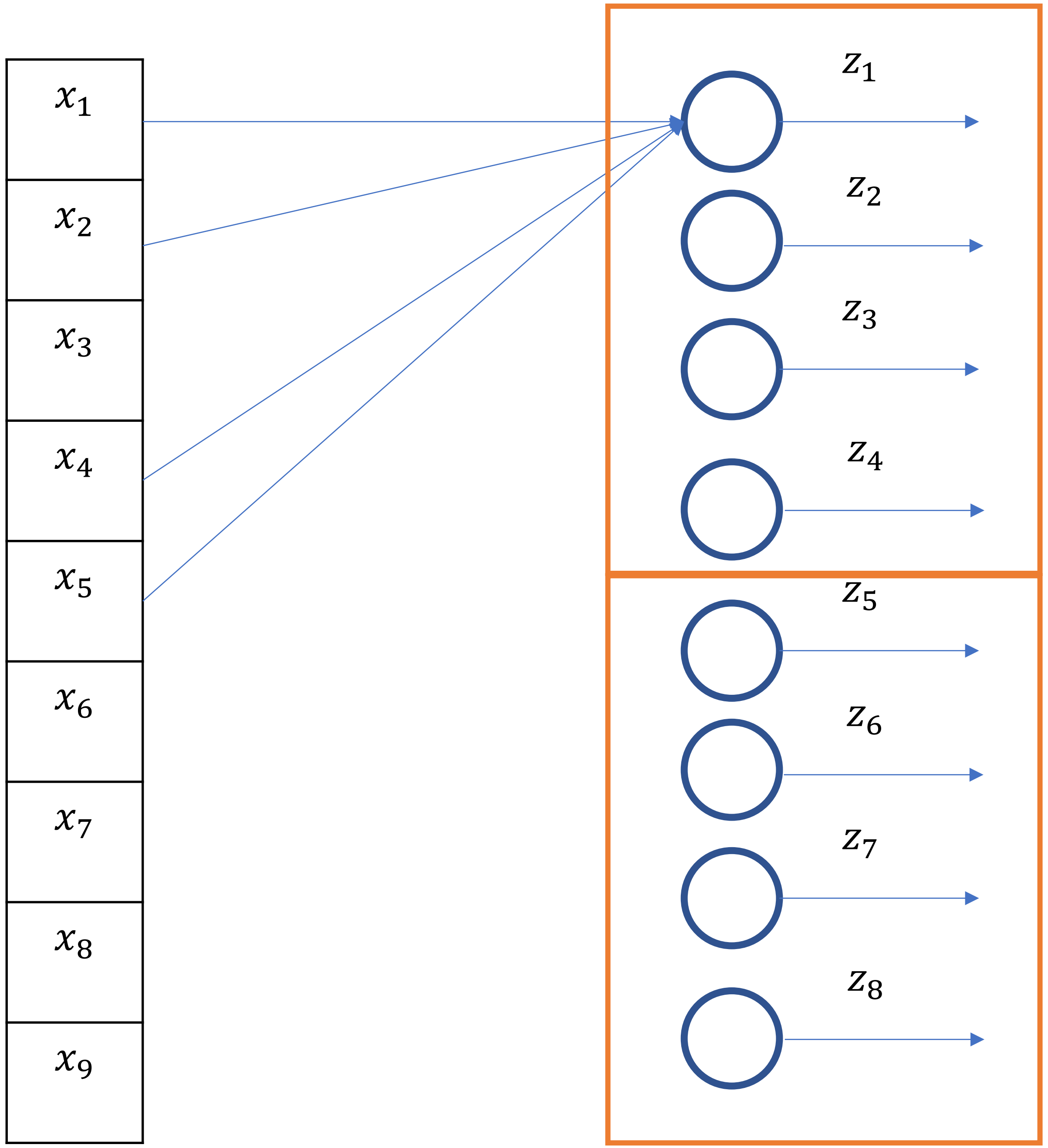
(Only connections for one neuron shown)

Filter 1

$$z_1 = w_1x_1 + w_2x_2 + w_3x_4 + w_4x_5 + b_1$$
$$z_2 = w_1x_2 + w_2x_3 + w_3x_5 + w_4x_6 + b_1$$
$$z_3 = w_1x_4 + w_2x_5 + w_3x_7 + w_4x_8 + b_1$$
$$z_4 = w_1x_5 + w_2x_6 + w_3x_8 + w_4x_9 + b_1$$

Filter 2

$$z_5 = w_5x_1 + w_6x_2 + w_7x_4 + w_8x_5 + b_2$$
$$z_6 = w_5x_2 + w_6x_3 + w_7x_5 + w_8x_6 + b_2$$
$$z_7 = w_5x_4 + w_6x_5 + w_7x_7 + w_8x_8 + b_2$$
$$z_8 = w_5x_5 + w_6x_6 + w_7x_8 + w_8x_9 + b_2$$



(Only connections for one neuron shown)

Filter 1

$$z_1 = w_1x_1 + w_2x_2 + w_3x_4 + w_4x_5 + b_1$$
$$z_2 = w_1x_2 + w_2x_3 + w_3x_5 + w_4x_6 + b_1$$
$$z_3 = w_1x_4 + w_2x_5 + w_3x_7 + w_4x_8 + b_1$$
$$z_4 = w_1x_5 + w_2x_6 + w_3x_8 + w_4x_9 + b_1$$

Filter 2

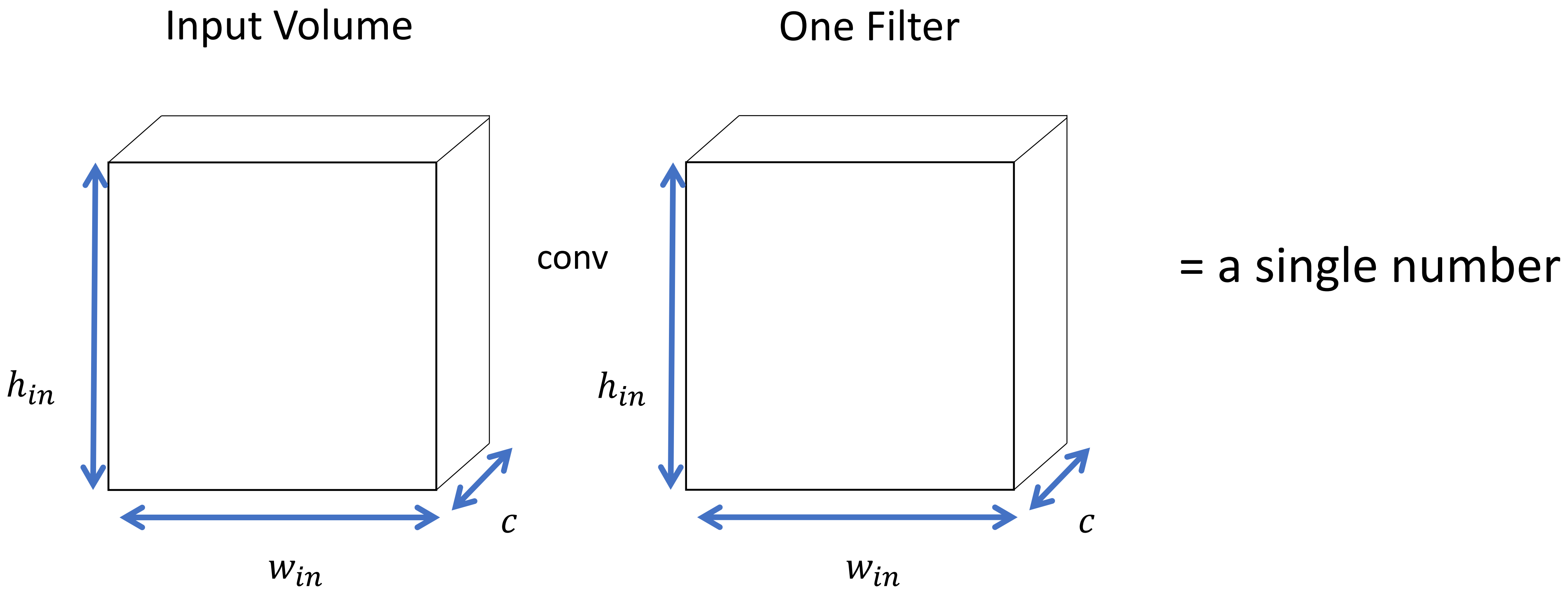
$$z_5 = w_5x_1 + w_6x_2 + w_7x_4 + w_8x_5 + b_2$$
$$z_6 = w_5x_2 + w_6x_3 + w_7x_5 + w_8x_6 + b_2$$
$$z_7 = w_5x_4 + w_6x_5 + w_7x_7 + w_8x_8 + b_2$$
$$z_8 = w_5x_5 + w_6x_6 + w_7x_8 + w_8x_9 + b_2$$

Key Takeaway: Convolution Layer is just a Fully-Connected layer with sparse connectivity and parameter sharing

Comparison to Fully Connected

- Fully-Connected Neural Network Layer:
  - $a^{[l]} = g\left(W^{[l]} \cdot a^{[l-1]} + b^{[l]}\right)$
- Convolutional Neural Network Layer
  - $a^{[l]} = g\left(conv\left(W^{[l]}, a^{[l-1]}\right) + b^{[l]}\right)$
- You can still use backward propagation and optimization techniques as before to train a CNN

Consider a filter the same shape as the input



Example for one filter (K=1):

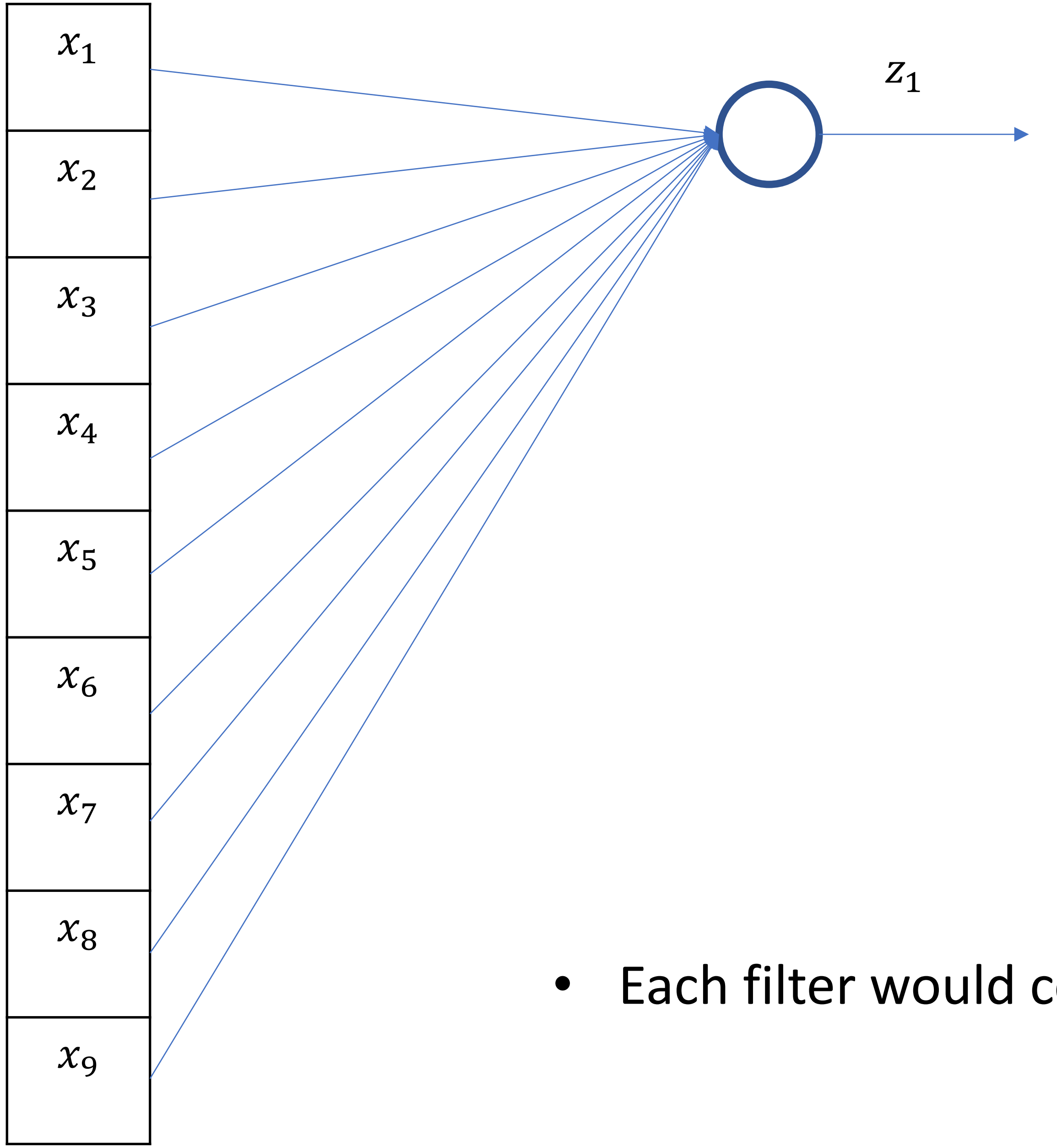
$$\begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix} \text{ conv } \begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7 & x_8 & x_9 \end{bmatrix} + \boxed{b_1} = \boxed{z_1}$$

3x3 3x3 2x2

$$z_1 = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + w_7x_7 + w_8x_8 + w_9x_9 + b_1$$

If we make the filter the same shape as the input, we have the equivalent of a single fully-connected neuron

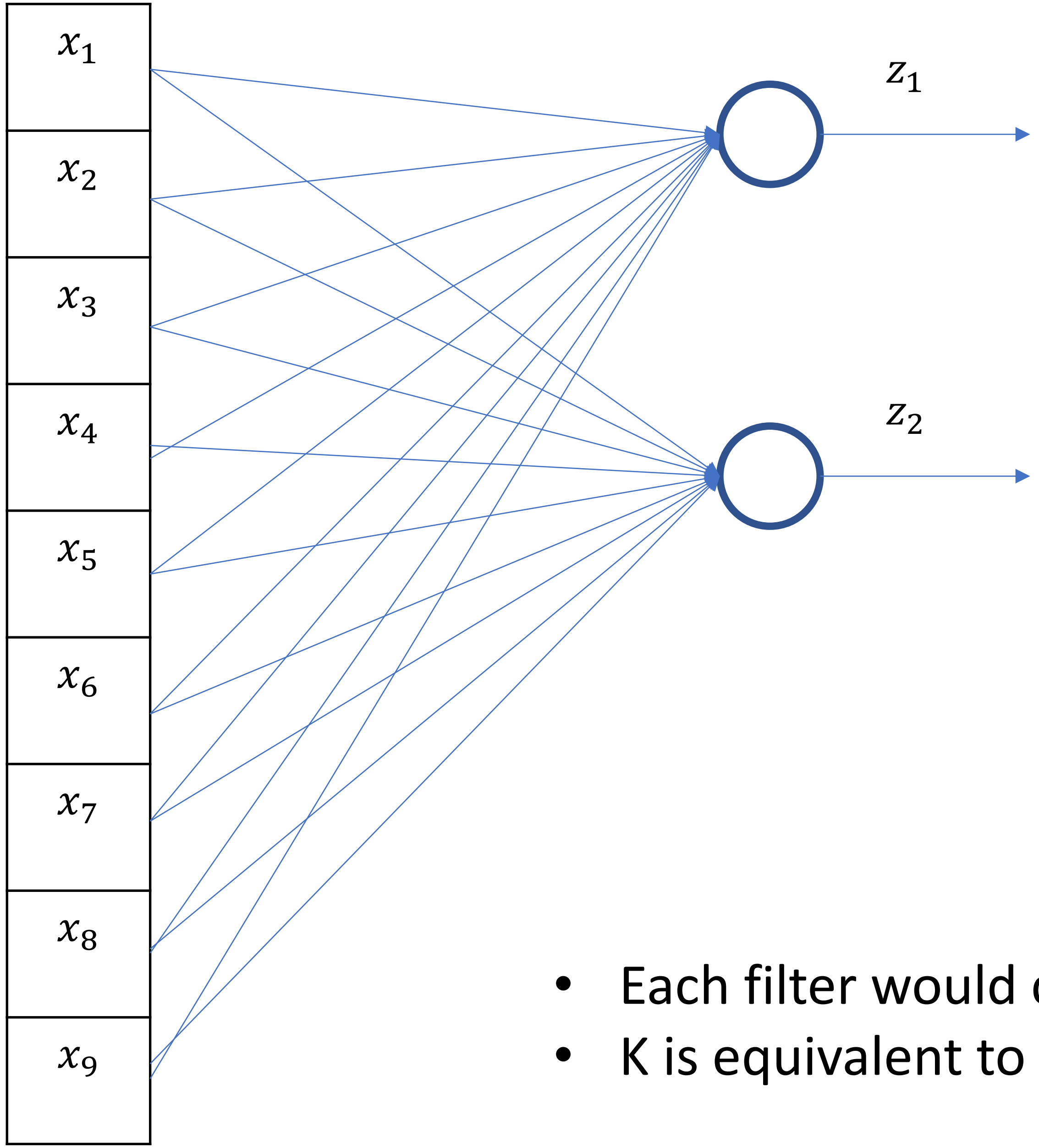
Number of filters: K=1



$$z_1 = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5$$
$$+w_6x_6 + w_7x_7 + w_8x_8 + w_9x_9 + b_1$$

- Each filter would correspond to a single fully-connected neuron

Number of filters: K=2



$$z_1 = w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6 + w_7x_7 + w_8x_8 + w_9x_9 + b_1$$

$$z_2 = w_{10}x_1 + w_{11}x_2 + w_{12}x_3 + w_{13}x_4 + w_{14}x_5 + w_{15}x_6 + w_{16}x_7 + w_{17}x_8 + w_{18}x_9 + b_2$$

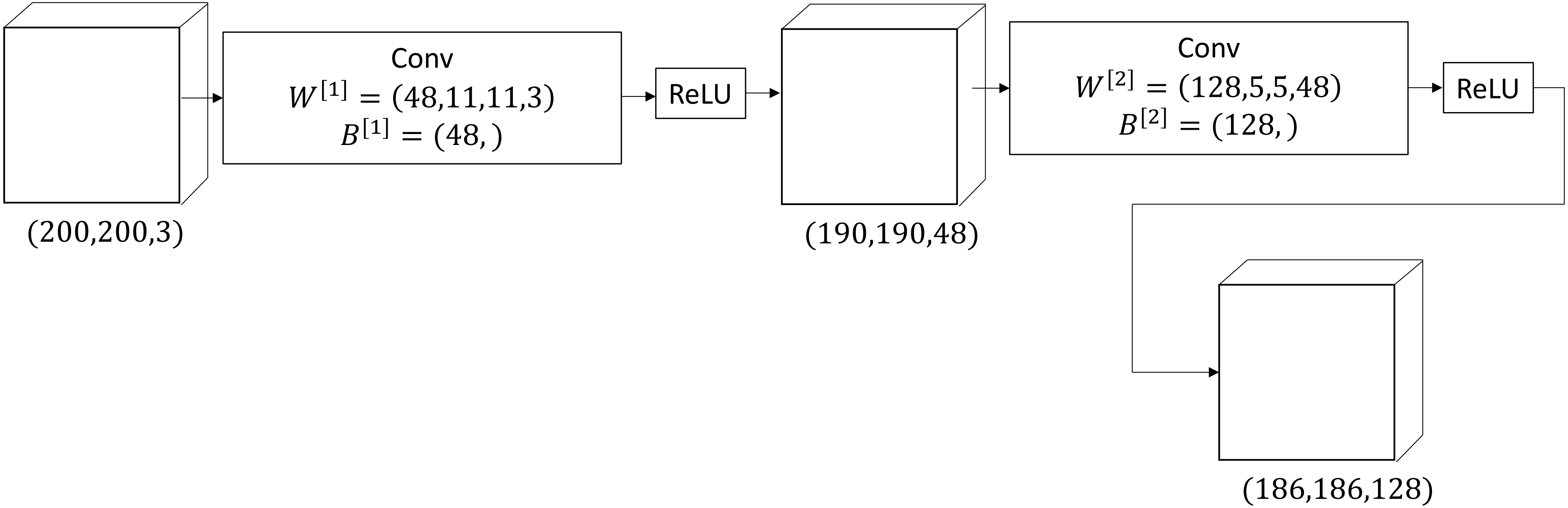
- Each filter would correspond to a single fully-connected neuron
- K is equivalent to n_h

Summary of Fully-Connected to Conv and Back

- You can transform a Fully-Connected layer to a Convolutional layer by employing sparse connectivity and parameter sharing
- You can transform a Convolutional layer to a Fully-Connected layer by making the filters the same shape as the input, and having one filter for each fully-connected neuron.
- We will see the latter case come up again when we talk about Object Detection

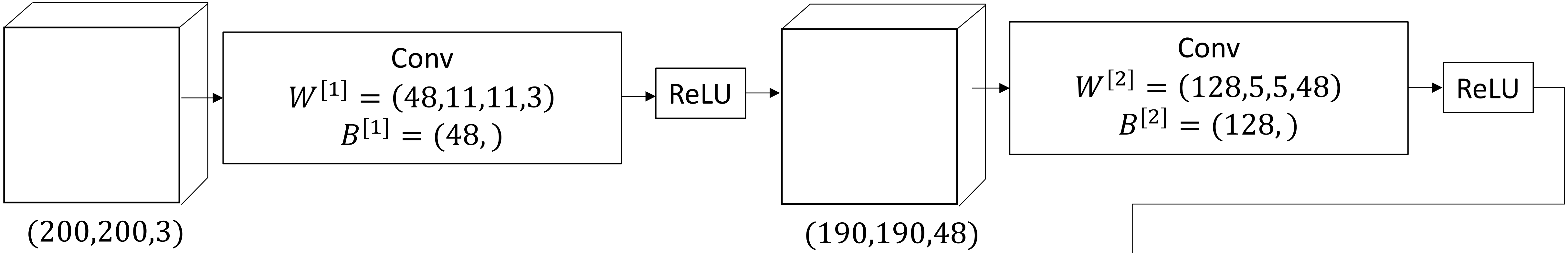
Convolutional Layers: Padding

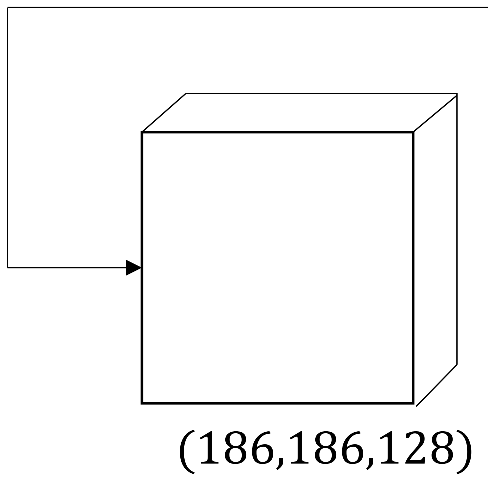
Simple CNN Example with numbers



Brad Quinton, Scott Chin

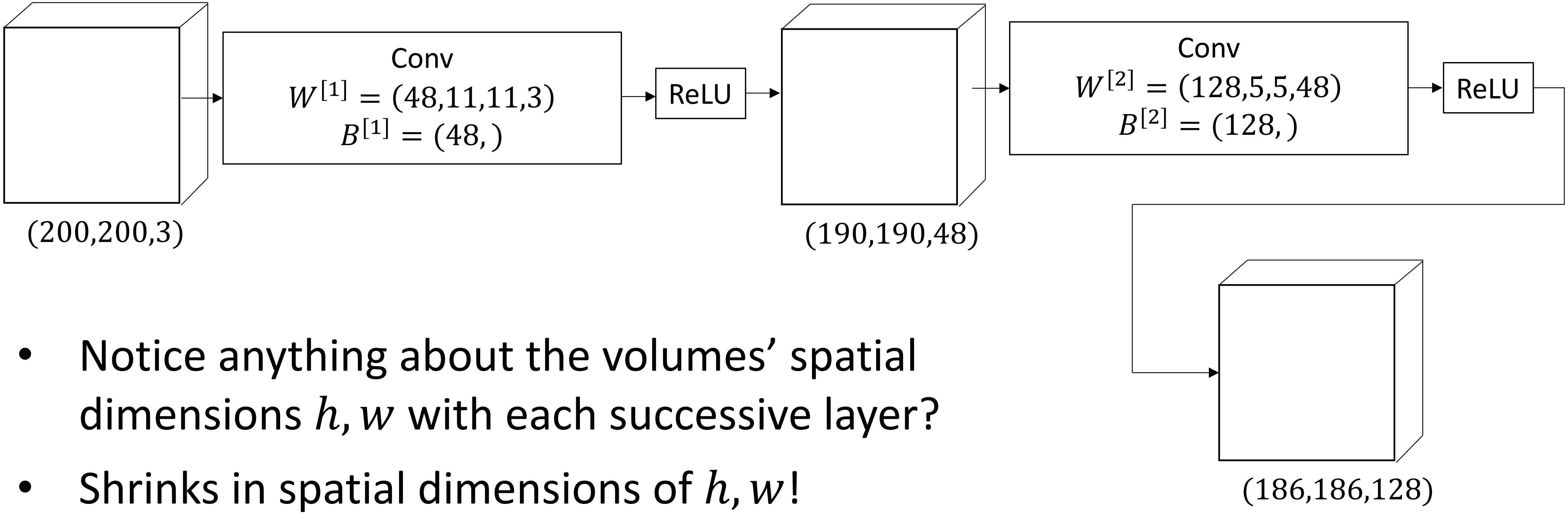
Simple CNN Example with numbers





- Notice anything about the volumes' spatial dimensions h, w with each successive layer?

Simple CNN Example with numbers



- Notice anything about the volumes' spatial dimensions h, w with each successive layer?
- Shrinks in spatial dimensions of h, w!
- Number of output activations depend on how many times you can fit the filter in the input

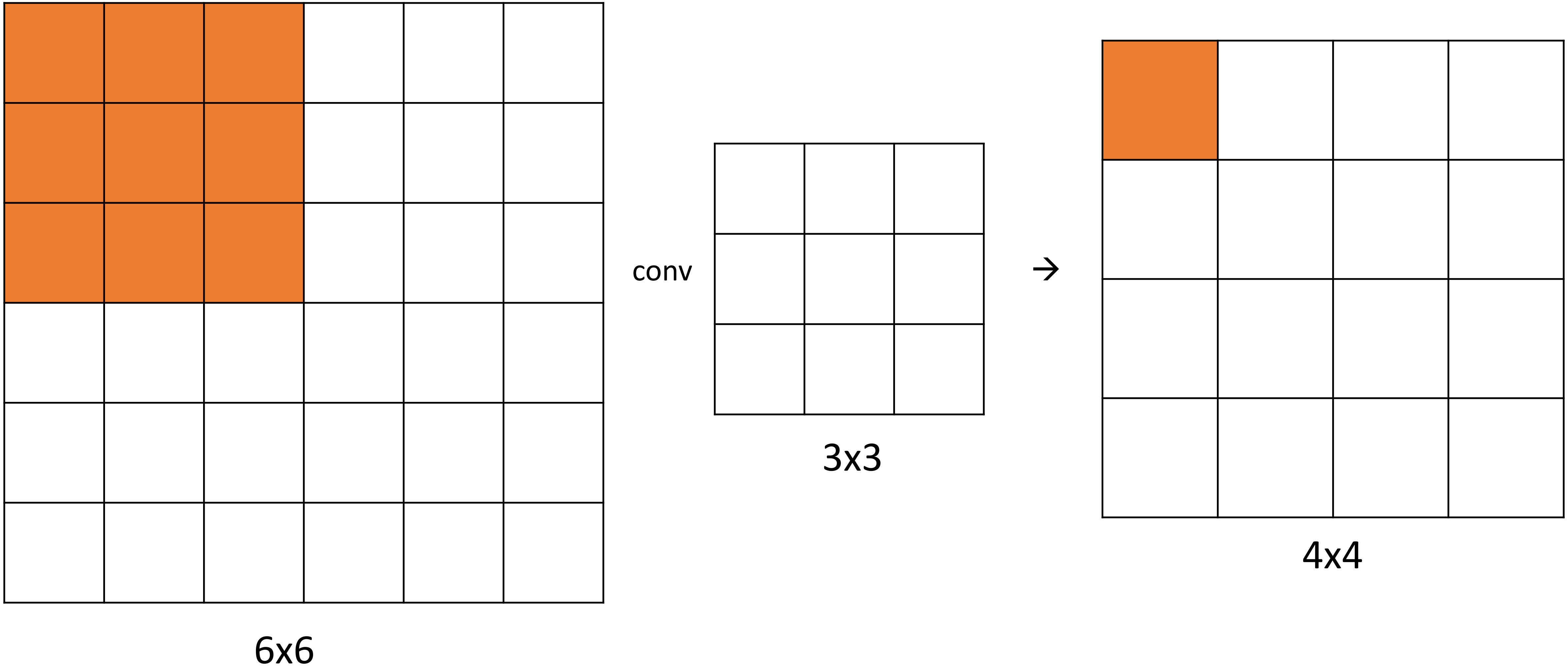
Brad Quinton, Scott Chin

Problem 1: Data Shrinks

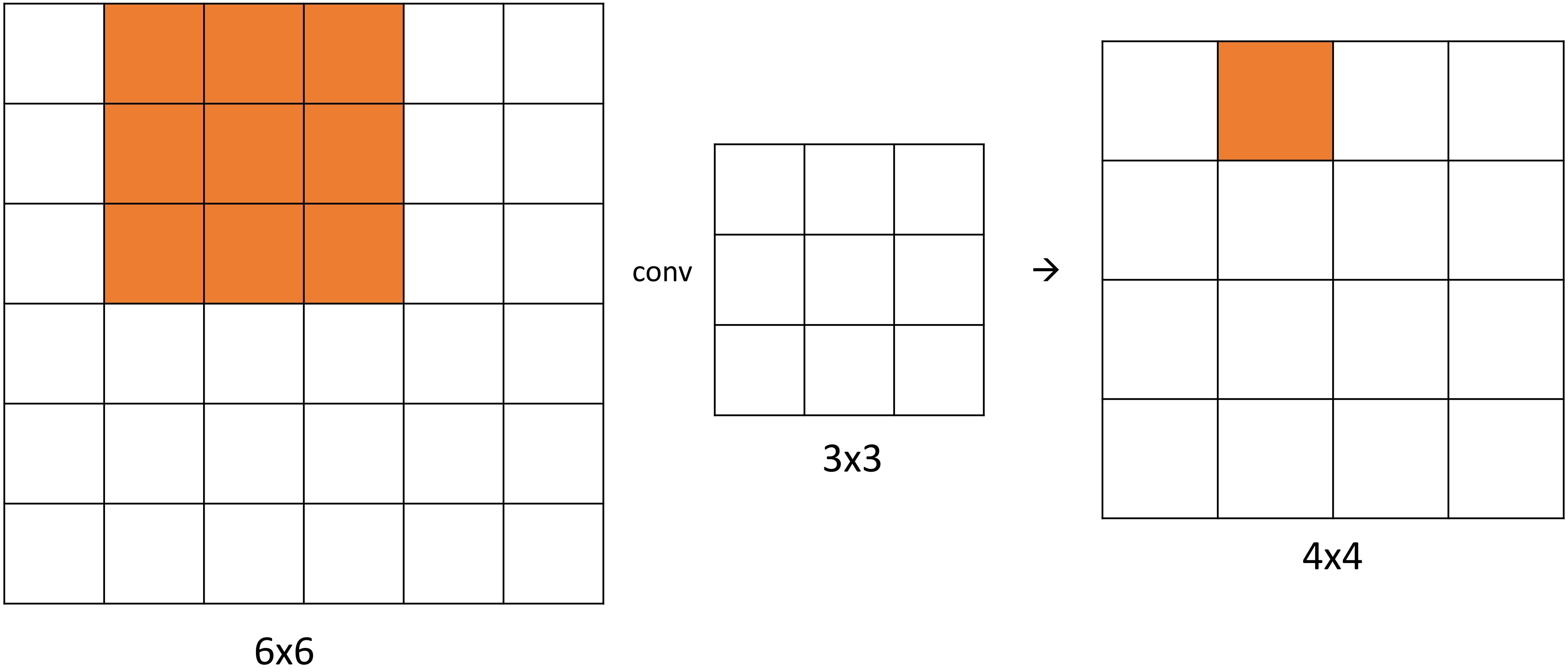
- Want to use the output of a Conv layer as the input to the next. But output volume's spatial dimensions shrink.
- This may limit how deep you can make your CNN

(32,32,3) conv (3,3) → (30,30) conv (3,3) → (28,28) conv (5,5) → (24,24) → …

Shrinking Volumes

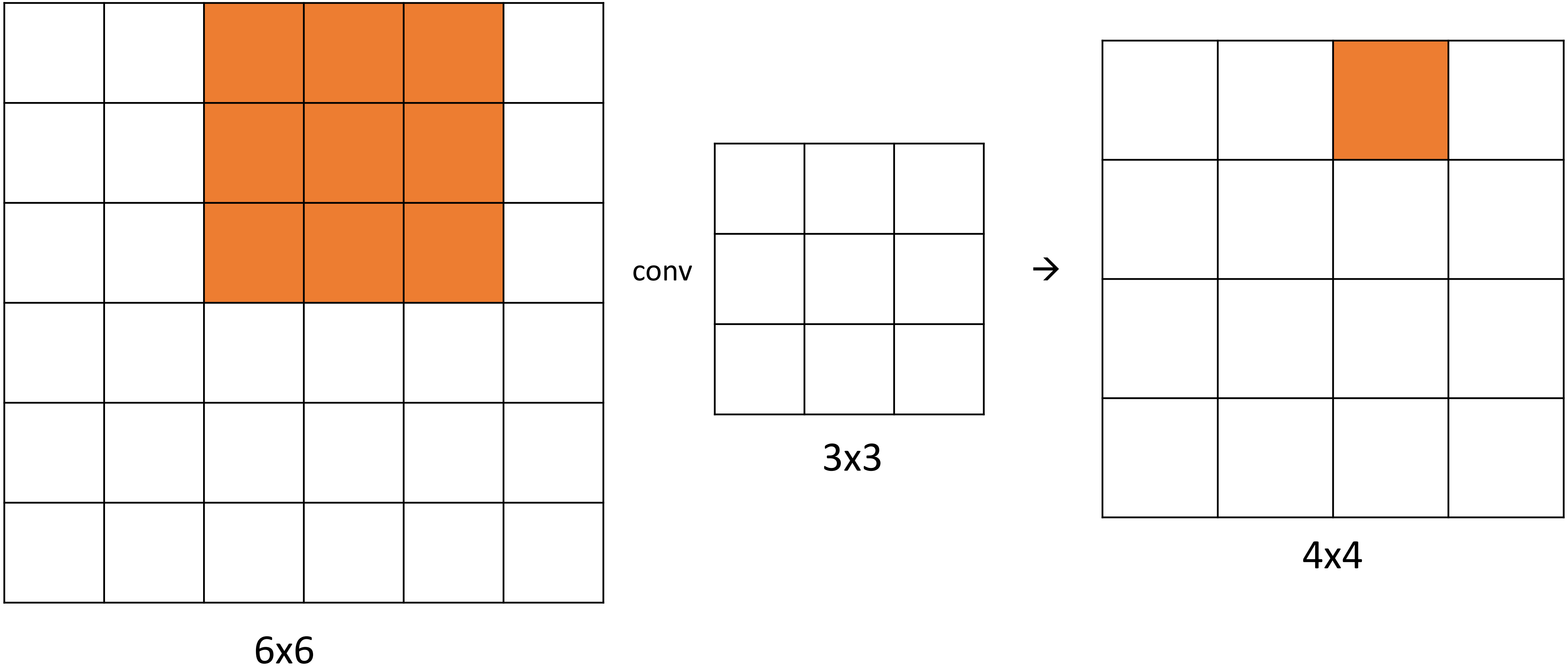


Shrinking Volumes



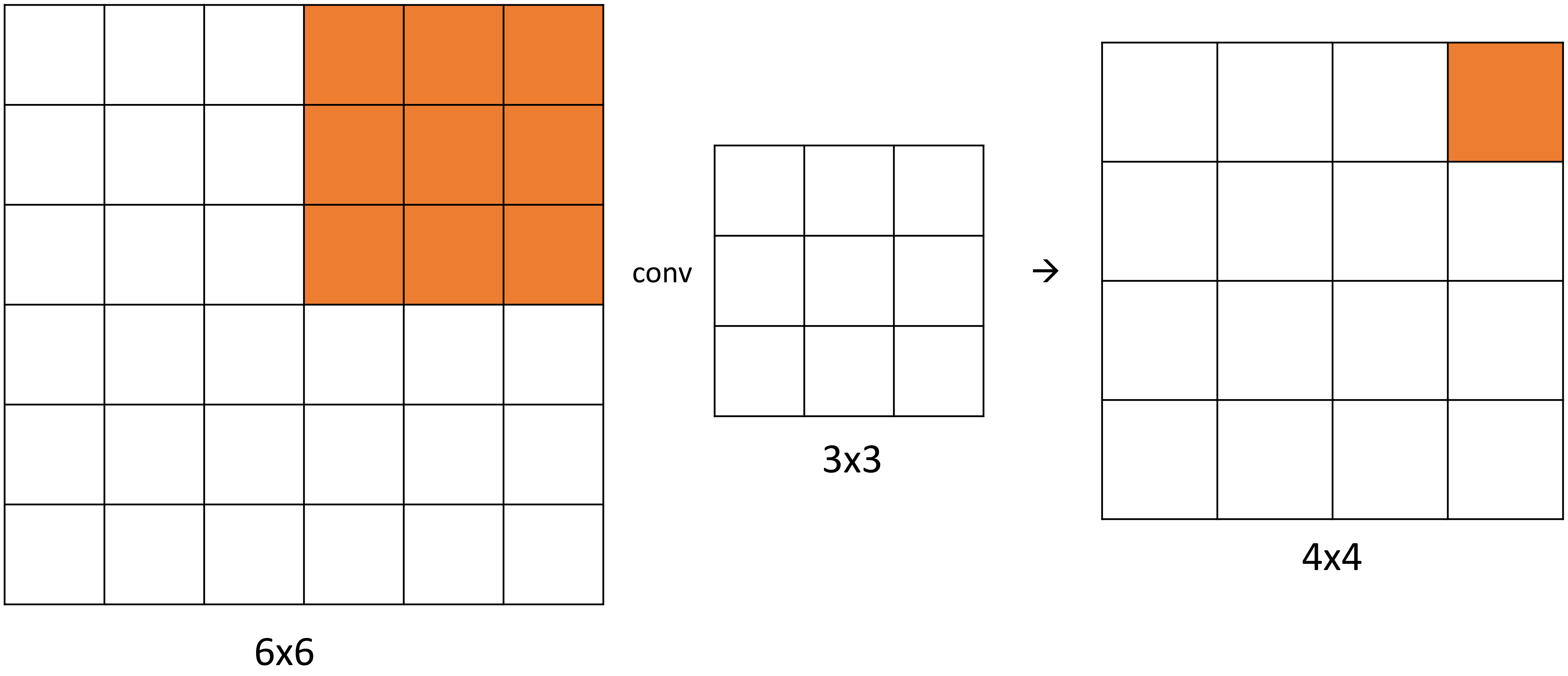
Brad Quinton, Scott Chin

Shrinking Volumes



Brad Quinton, Scott Chin

Shrinking Volumes

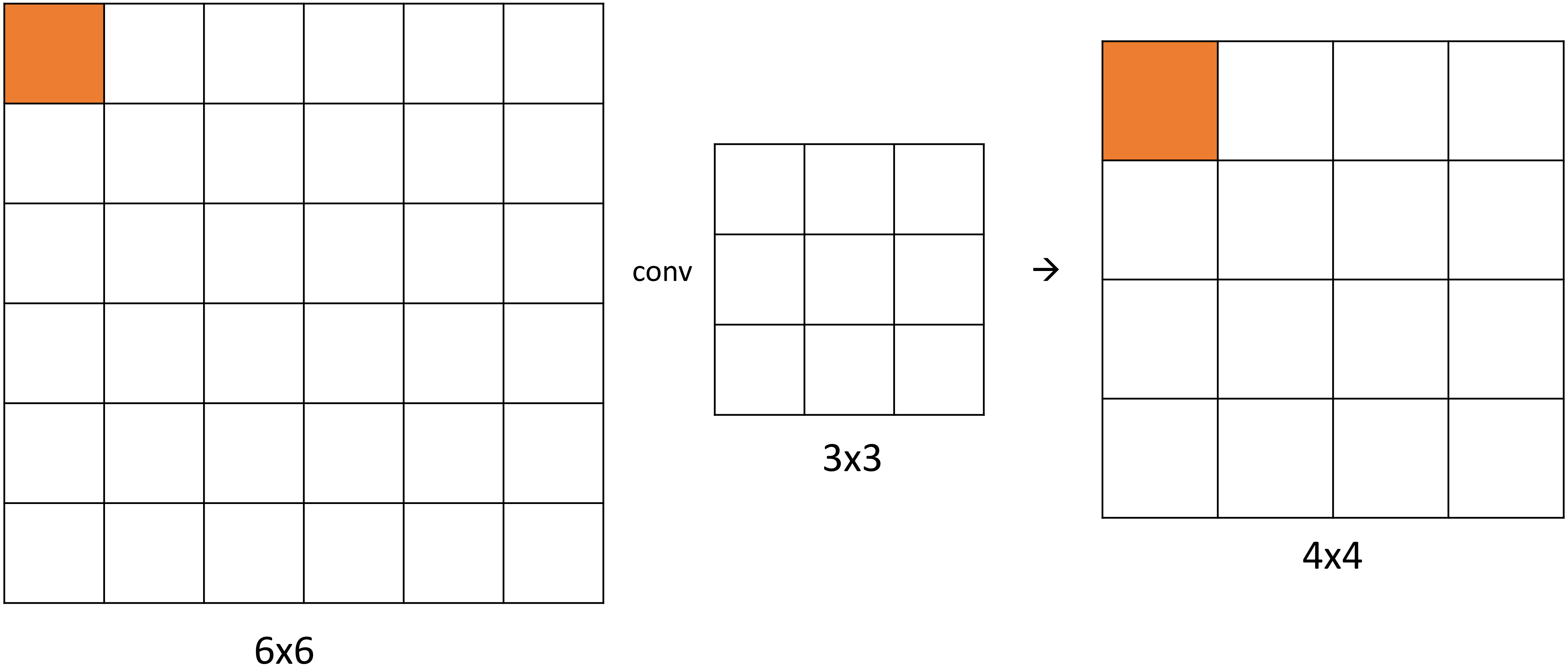


Brad Quinton, Scott Chin

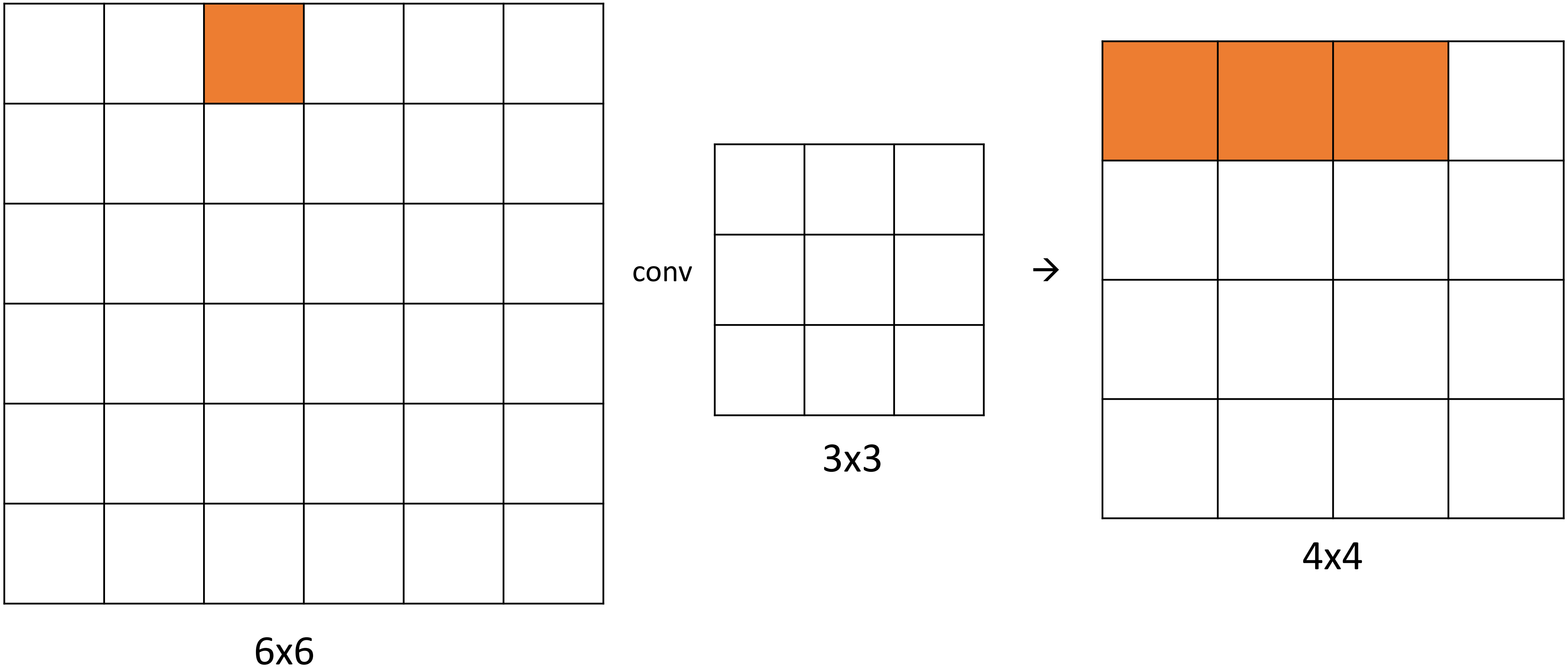
Problem 2: Data at the edge

- Input data at the edges influence fewer output values than input data in the middle.

Data at Corner

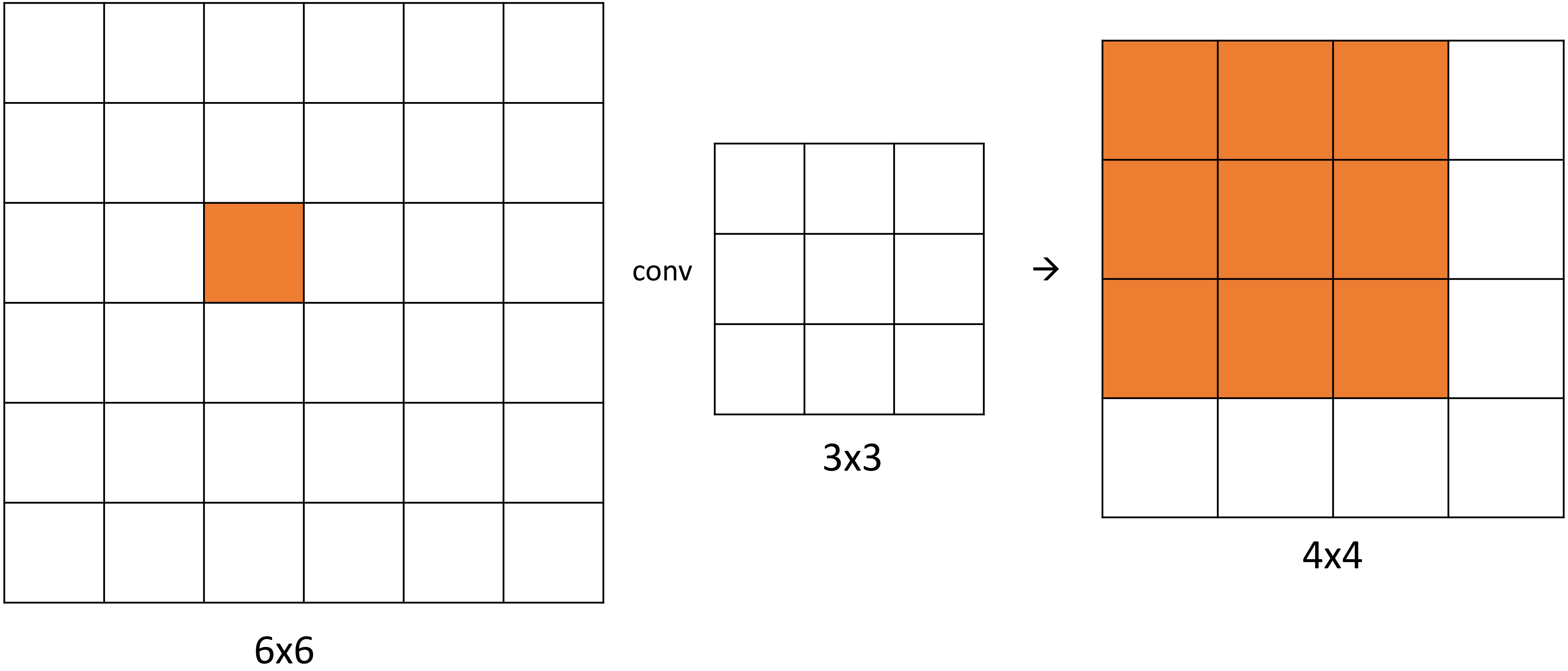


Data at Edge (non-corner)



Brad Quinton, Scott Chin

Data in the middle



Solution: Padding

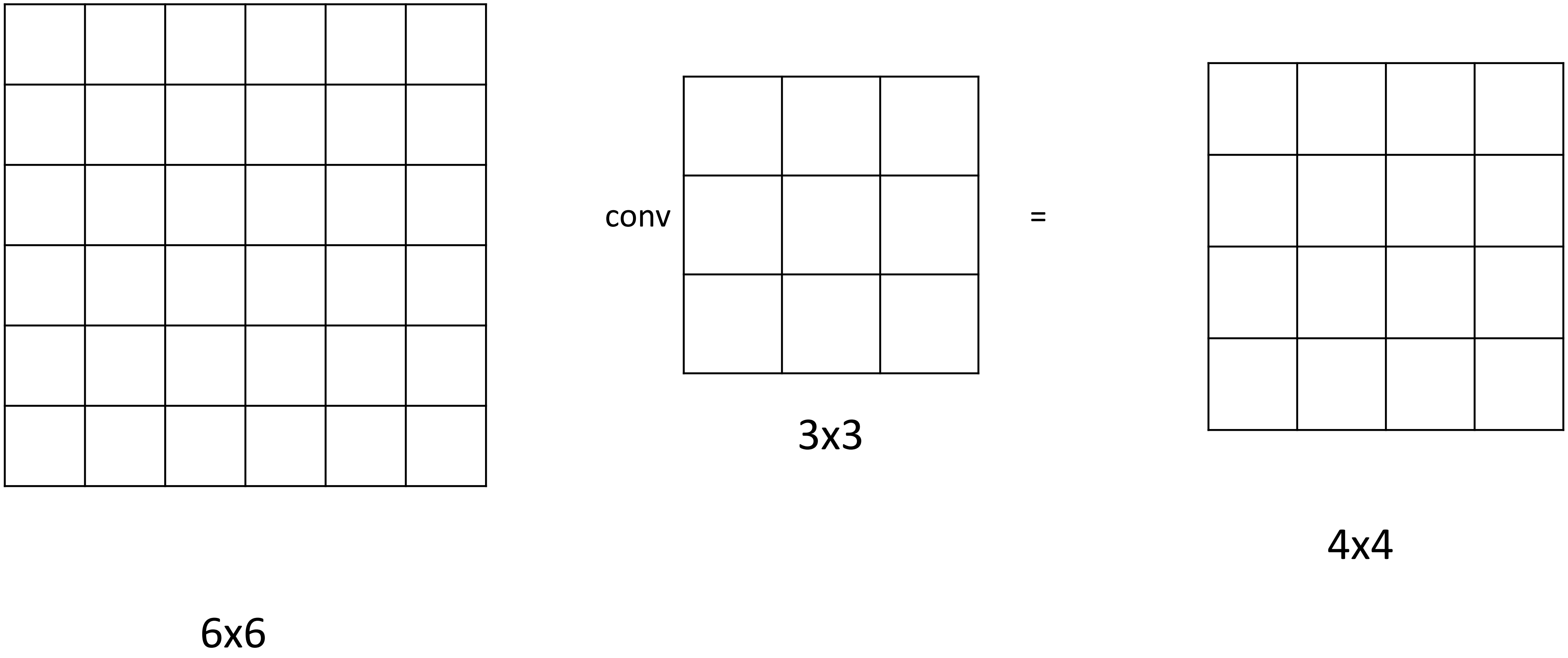
- Pad the perimeter of the input volume before convolution operation

Example:

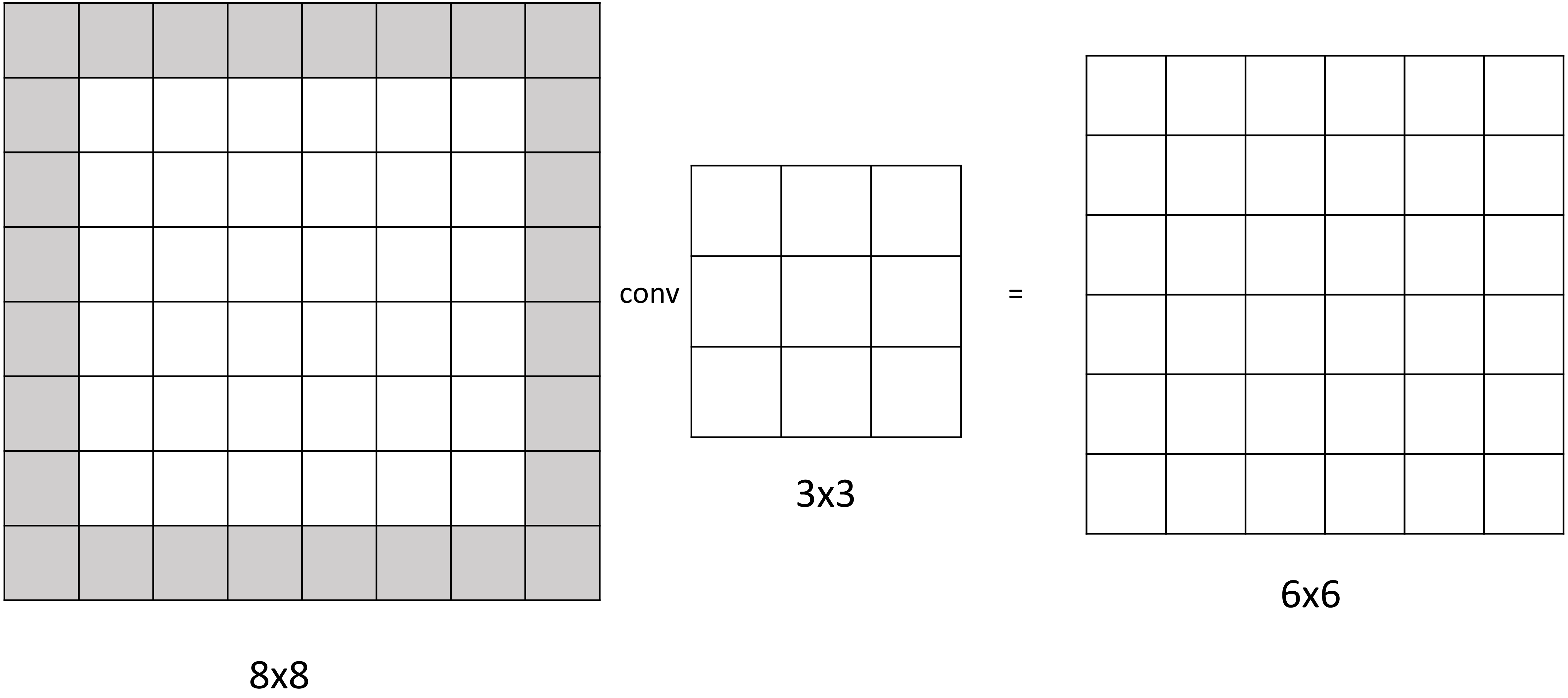
- Pad a 6x6 image with 1 element **all around** perimeter becomes 8x8
- Output is now 6x6 → This preserves original spatial dimensions
- Original perimeter data now affects more output values
- Both problems solved …

Brad Quinton, Scott Chin

Example: Padding

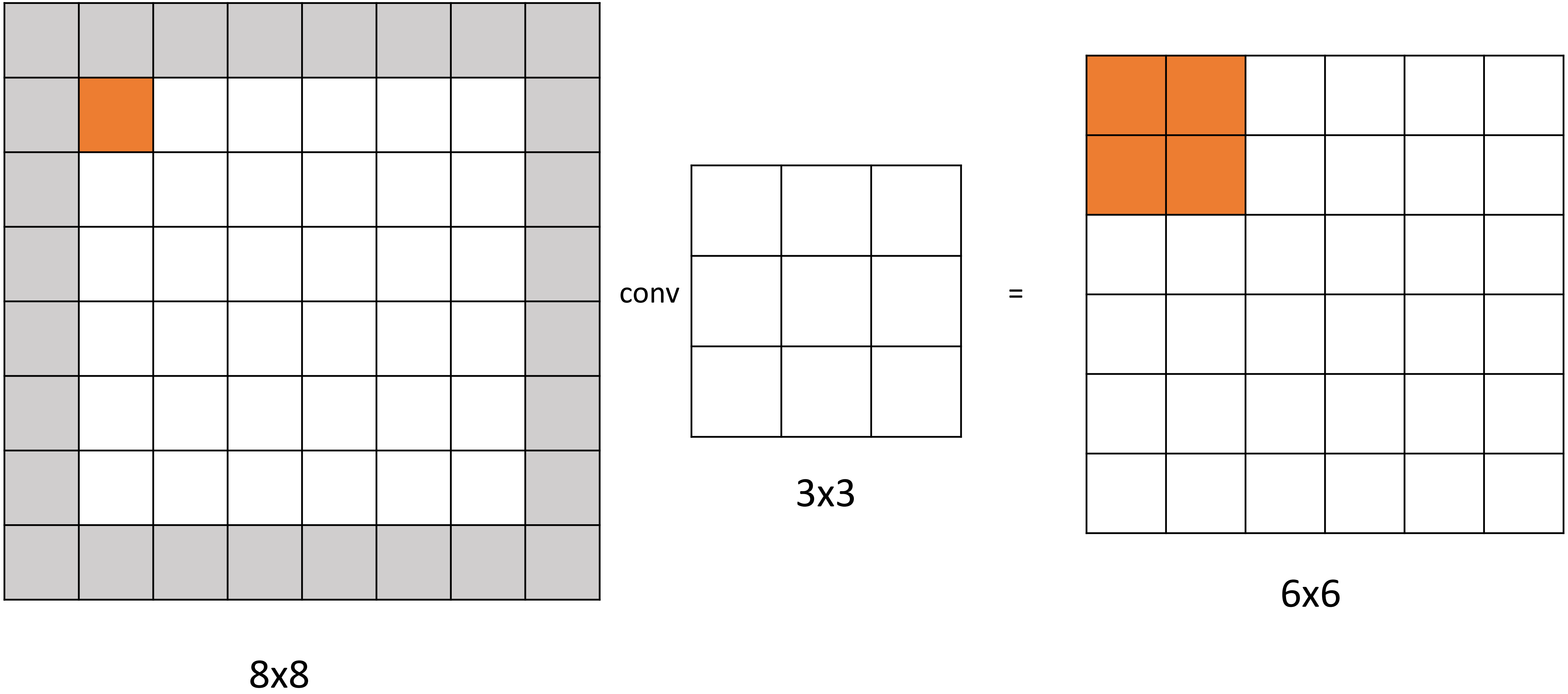


Example: Padding



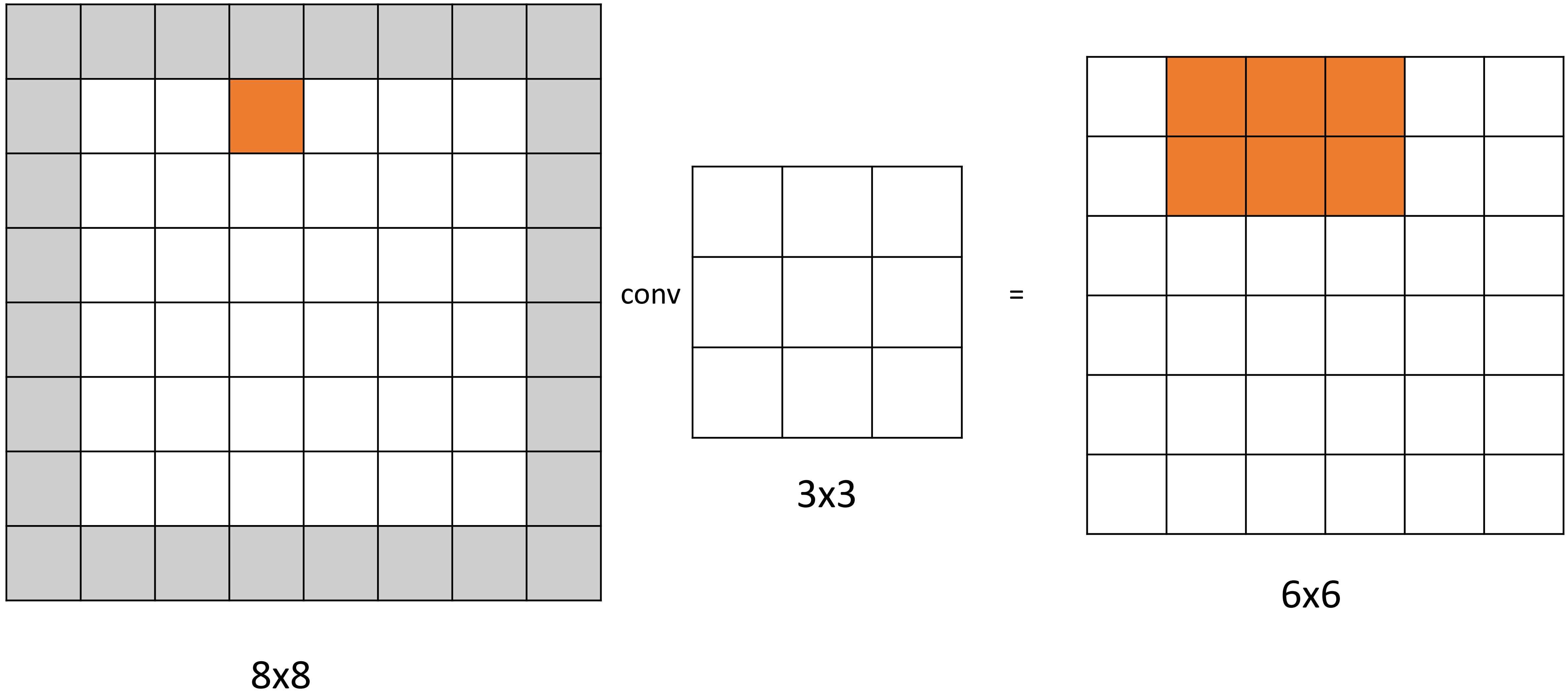
Brad Quinton, Scott Chin

Example: Corner



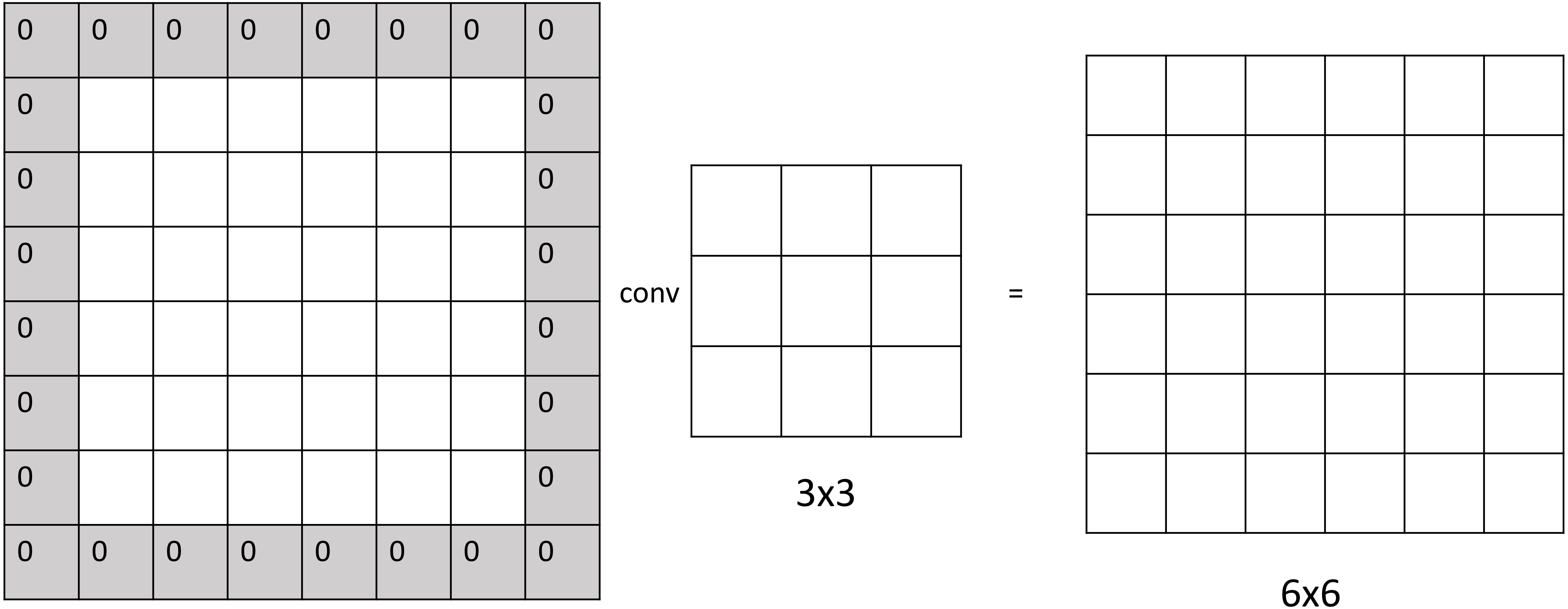
Brad Quinton, Scott Chin

Example: Edge



Brad Quinton, Scott Chin

Typically pad with a value of 0



8x8

3x3

6x6

Output Volume Dimensions with Padding

- Input volume: (h, w, c)
- Filter size: (f, f, c)
- Output volume without padding: $(h - f + 1, w - f + 1)$
- With padding $(h + 2p - f + 1, W + 2p - f + 1)$

Padding In Practice

- Typically pad in one of two ways:
  1. Don't pad
  2. Pad so that output volume spatial dimensions is the same as the input volume.
- For the latter, you usually don't manually specify p

Padding In Practice

- Typically pad in one of two ways:
    1. Don't pad
    2. Pad so that output volume spatial dimensions is the same as the input volume.
- For the latter, you usually don't manually specify p

Some terminology:

- No padding ← VALID
- Pad so that output volume is the same as the input volume ← SAME

Brad Quinton, Scott Chin

“Same” Padding

- Pad so that output volume is the same as the input volume

<u>Example</u>:

- Input is (h, w), output is $(h + 2p - f + 1, w + 2p - f + 1)$
- What to set p so that input is equal to output?

Brad Quinton, Scott Chin

"Same" Padding

- Pad so that output volume is the same as the input volume

<u>Example</u>:

- Input is (h, w), output is $(h + 2p - f + 1, w + 2p - f + 1)$
- What to set p so that input is equal to output?
- $h = h + 2p - f + 1 \rightarrow p = \frac{f-1}{2}$
- $w = w + 2p - f + 1 \rightarrow p = \frac{f-1}{2}$

- Doesn't depend on input volume. Just depends on filter size.

Brad Quinton, Scott Chin

“Same” Padding

- Works well for odd sized filters
- What about even? You can do asymmetric padding (i.e. one side has more padding than the other)
- But due to long traditional reasons in computer vision, filters are almost always odd and square
  - Nice to have a central pixel and notion of top/bottom/left/right

- Just use odd sized filters …

Same Padding

- What if p doesn't work out to an integer?

$$p = \frac{f - 1}{2}$$

Same Padding

- What if p doesn't work out to an integer?

$$p = \frac{f-1}{2}$$

- You would typically design your CNN architecture so that p works out to be an integer

Brad Quinton, Scott Chin

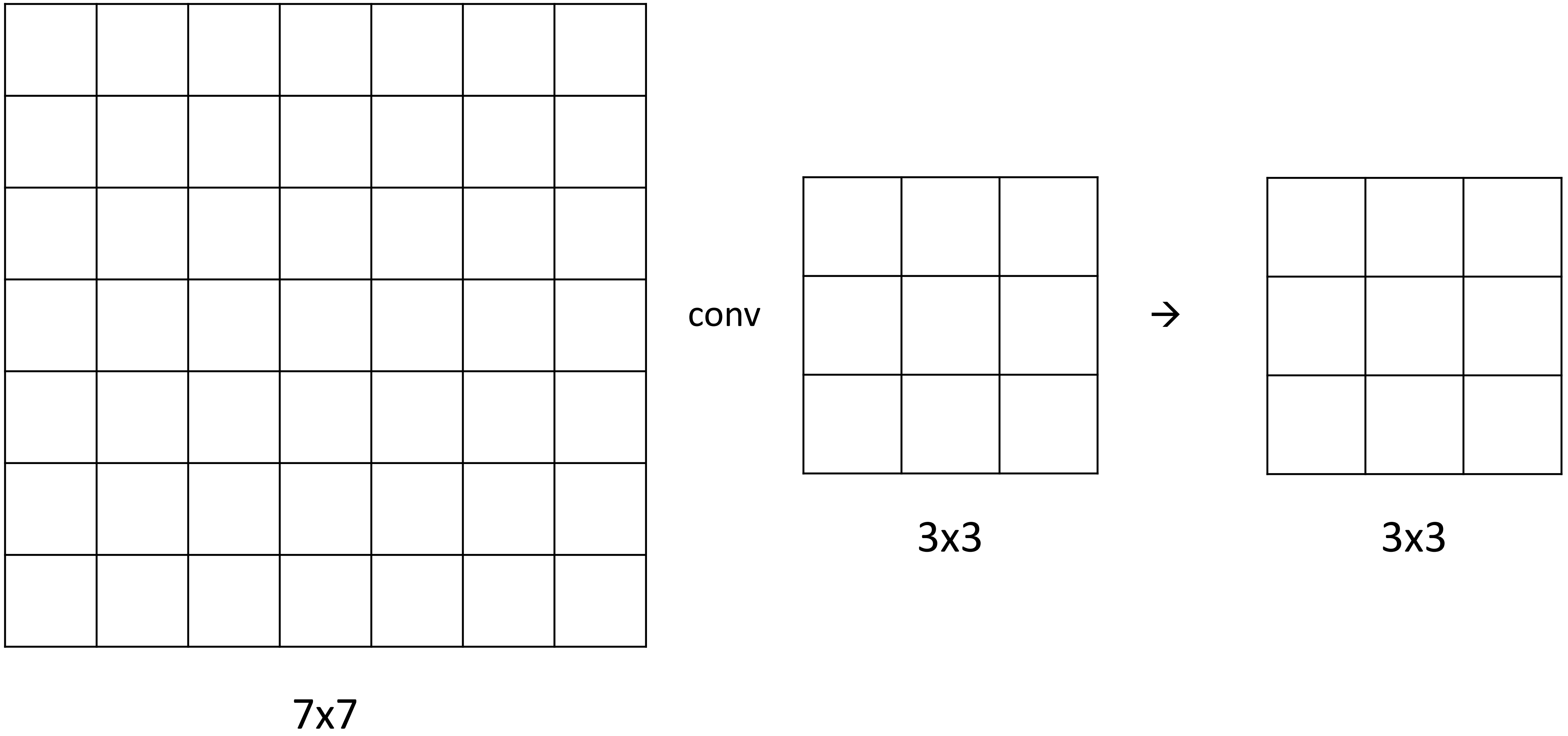
Stride

Stride

- So far, when we perform the convolutions, the filter “slides” 1 value at a time across the data
- But you could slide by larger steps.
- This is another hyperparameter of your CNN architecture
- The amount by which you step is referred to as the “stride”

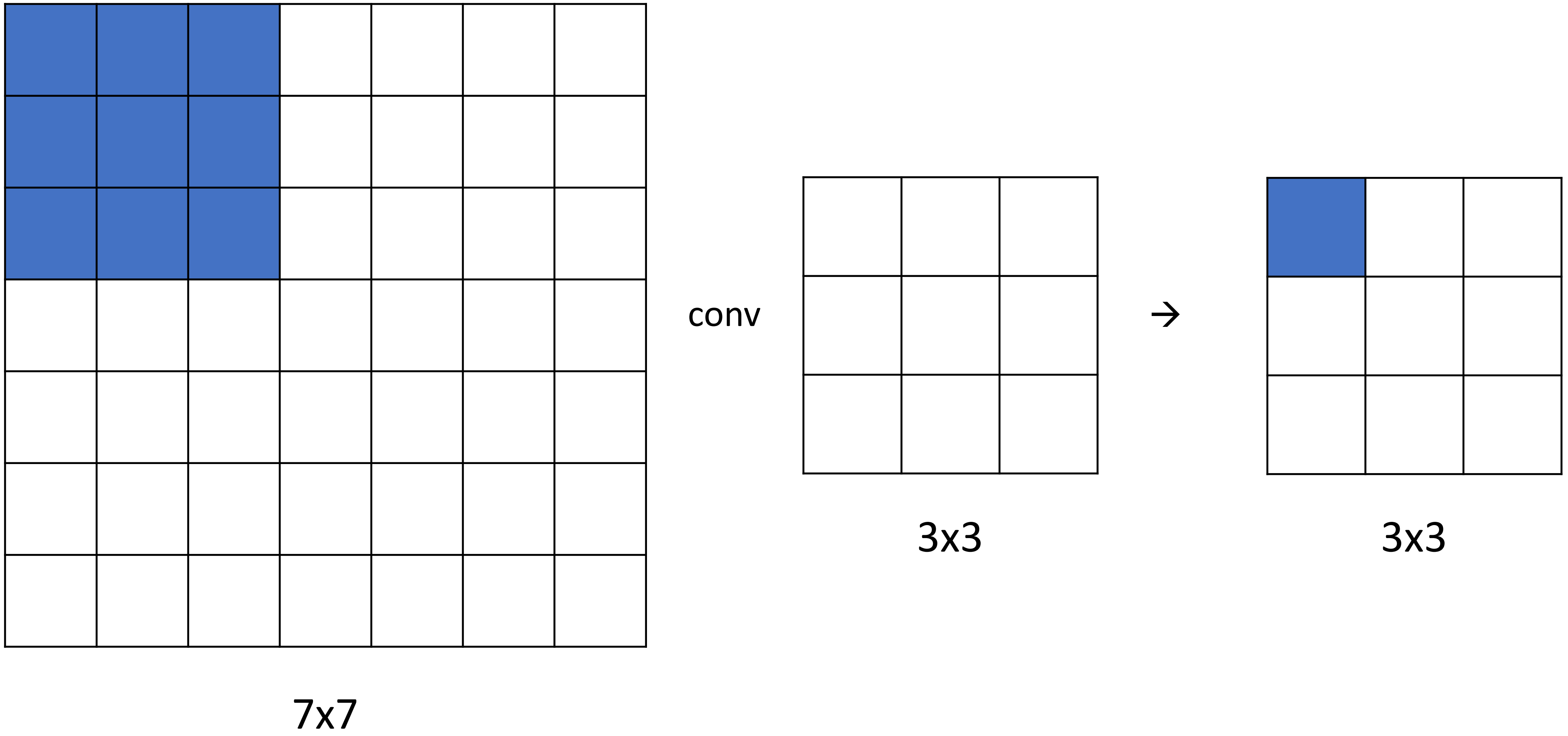
Brad Quinton, Scott Chin

Example: Stride = 2



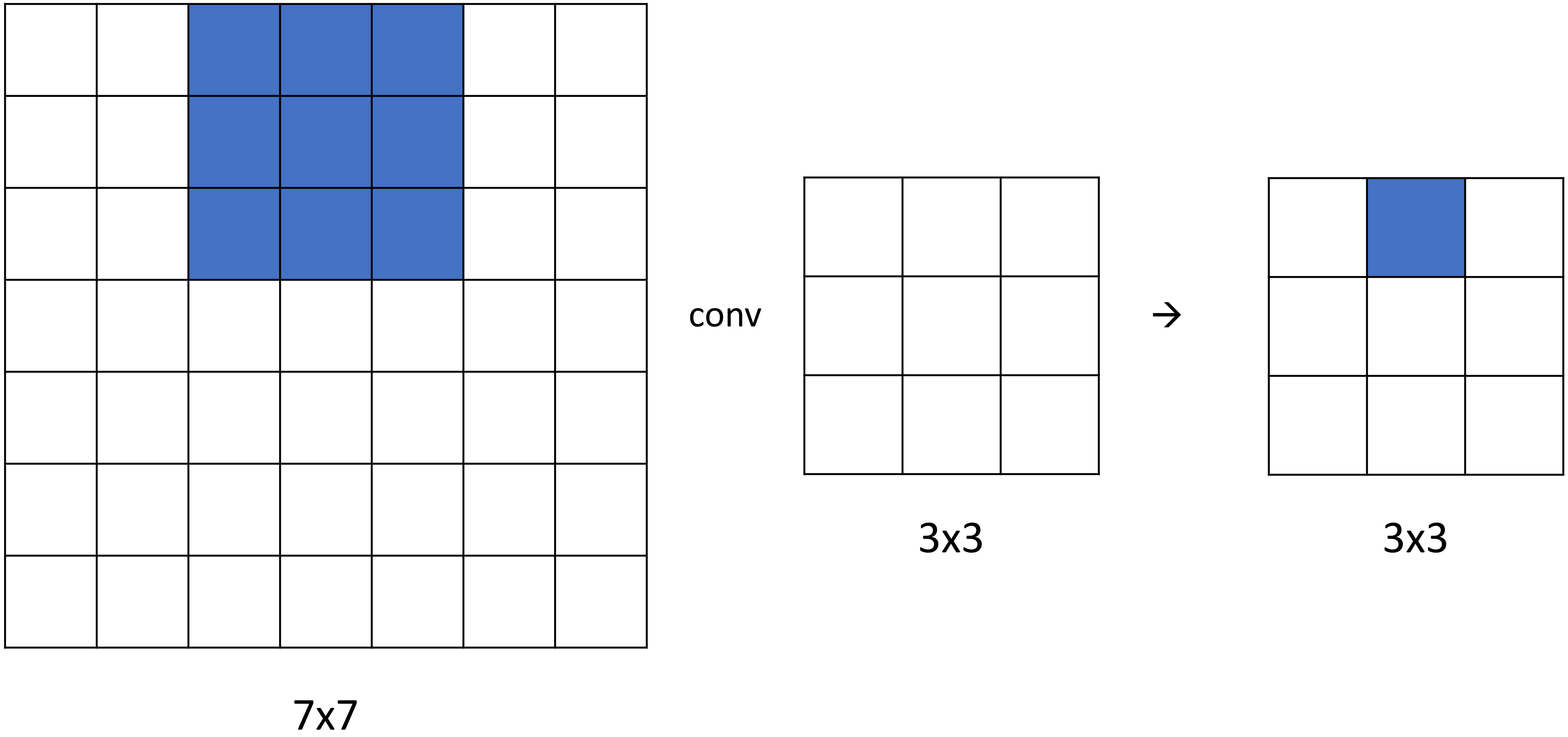
Brad Quinton, Scott Chin

Example: Stride = 2



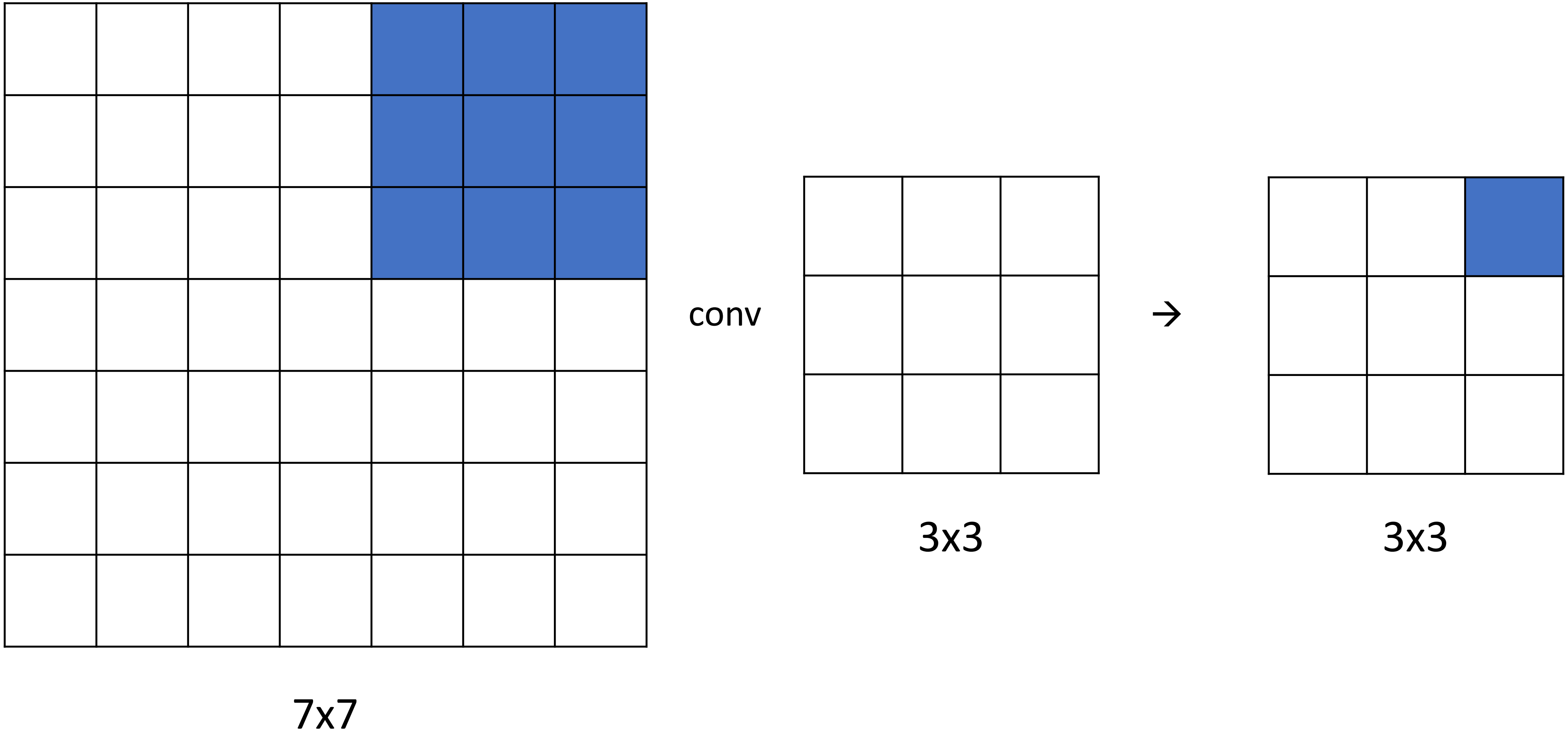
Brad Quinton, Scott Chin

Example: Stride = 2

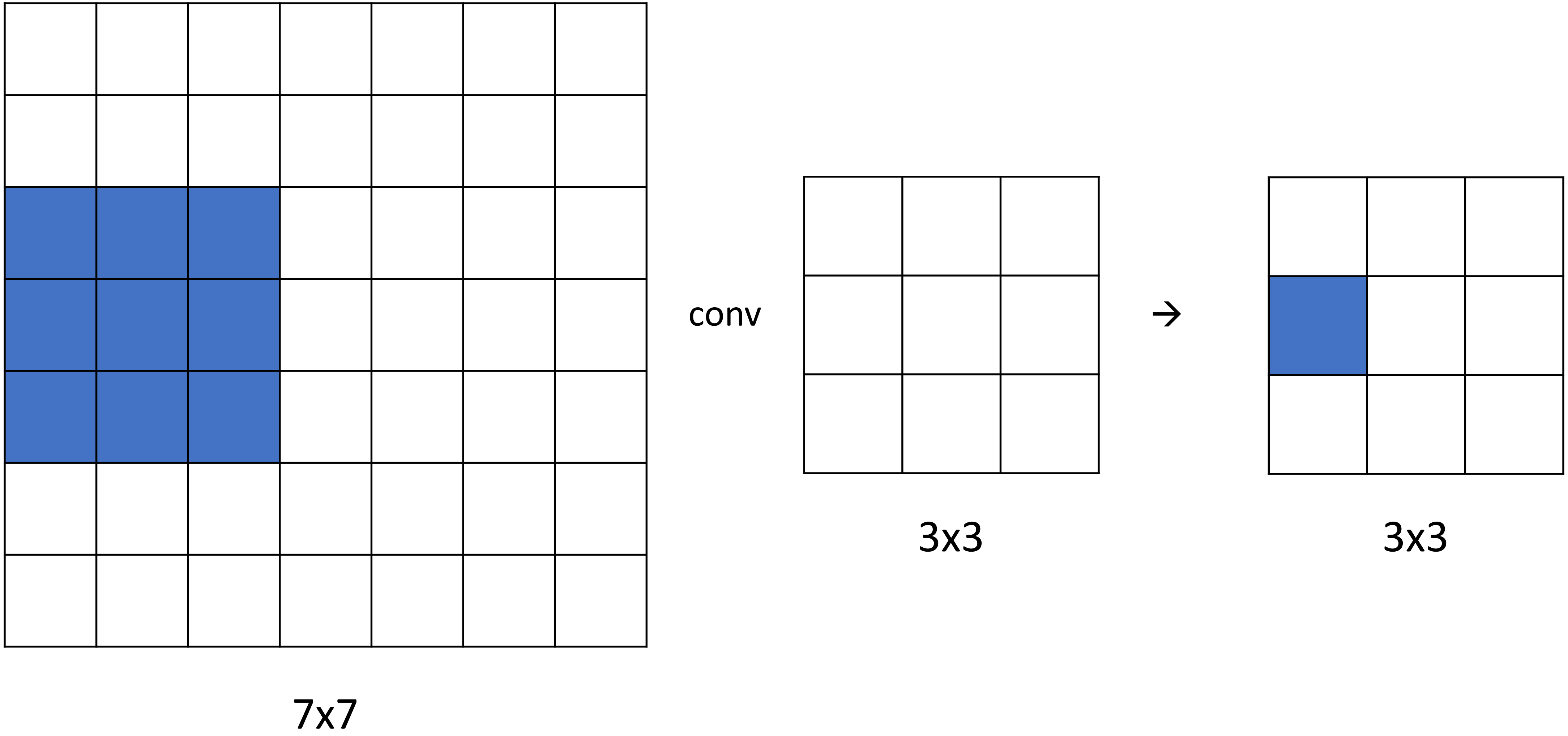


Brad Quinton, Scott Chin

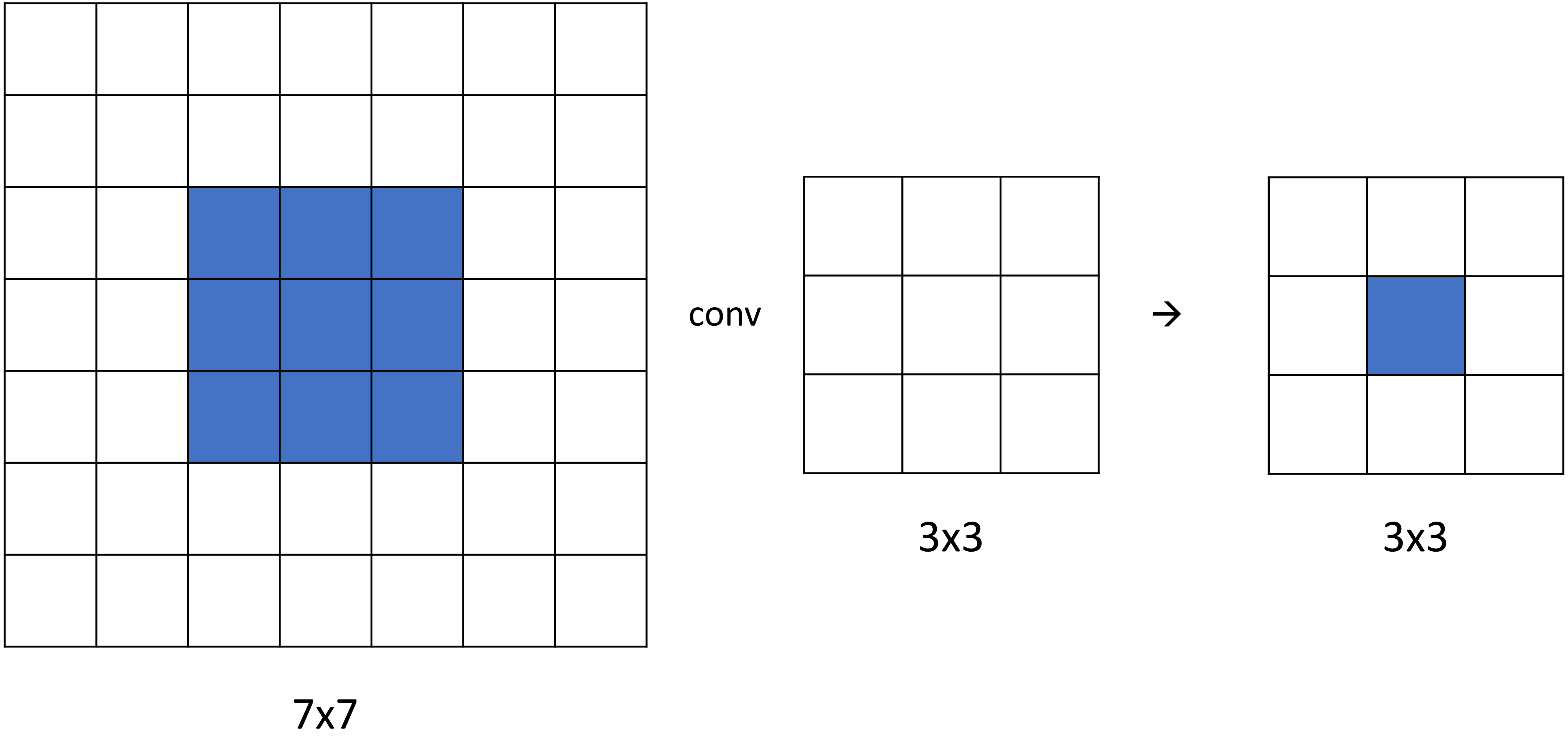
Example: Stride = 2



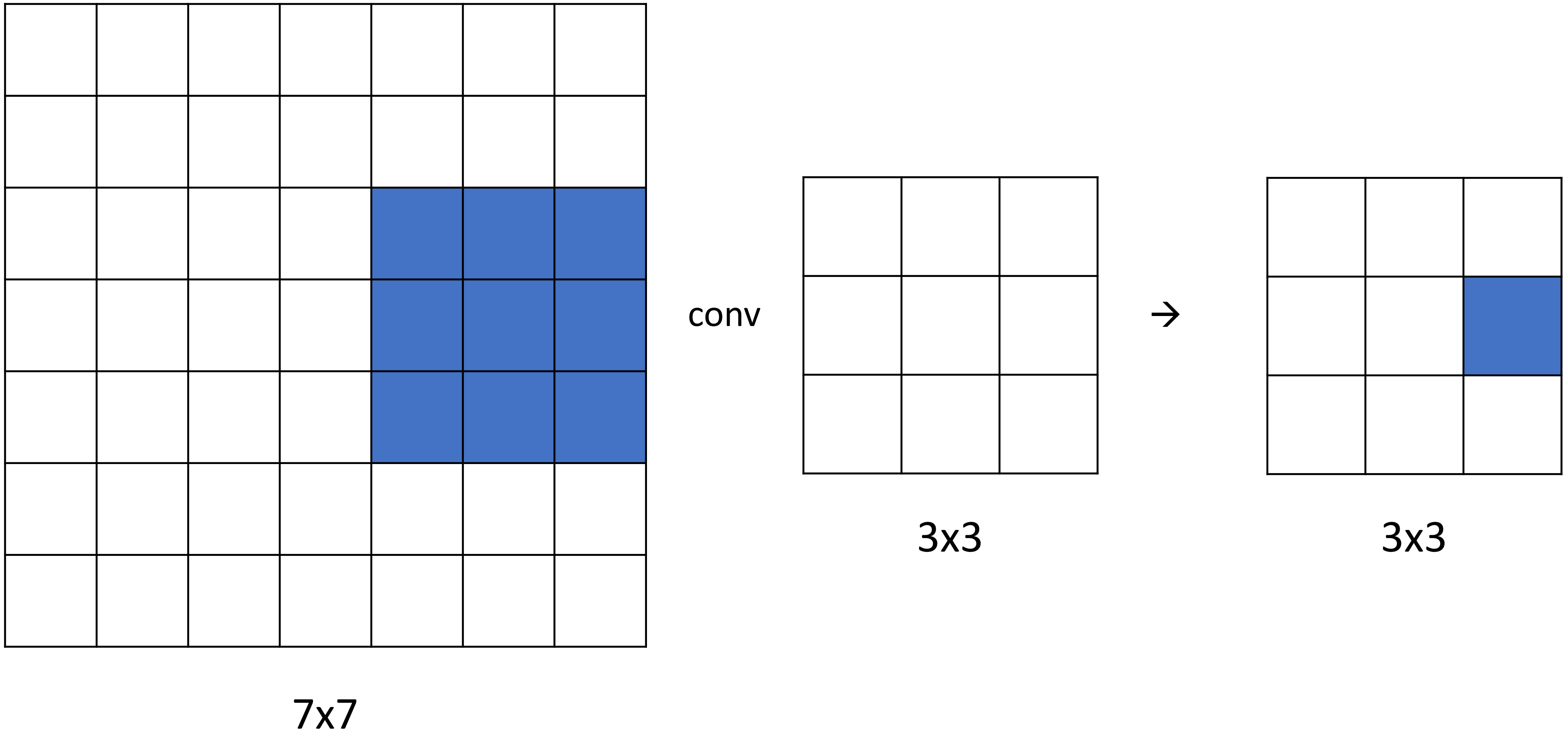
Example: Stride = 2



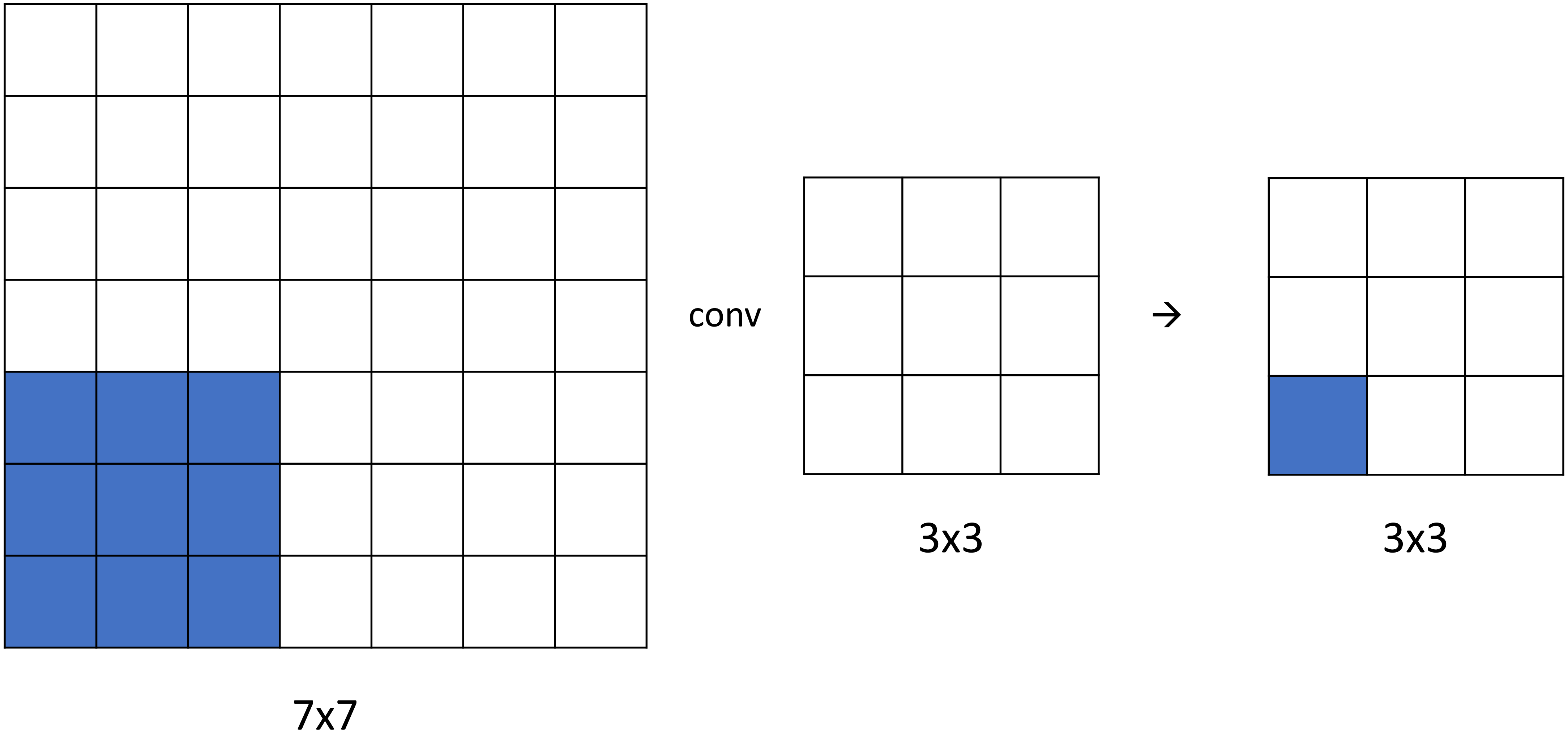
Example: Stride = 2



Example: Stride = 2

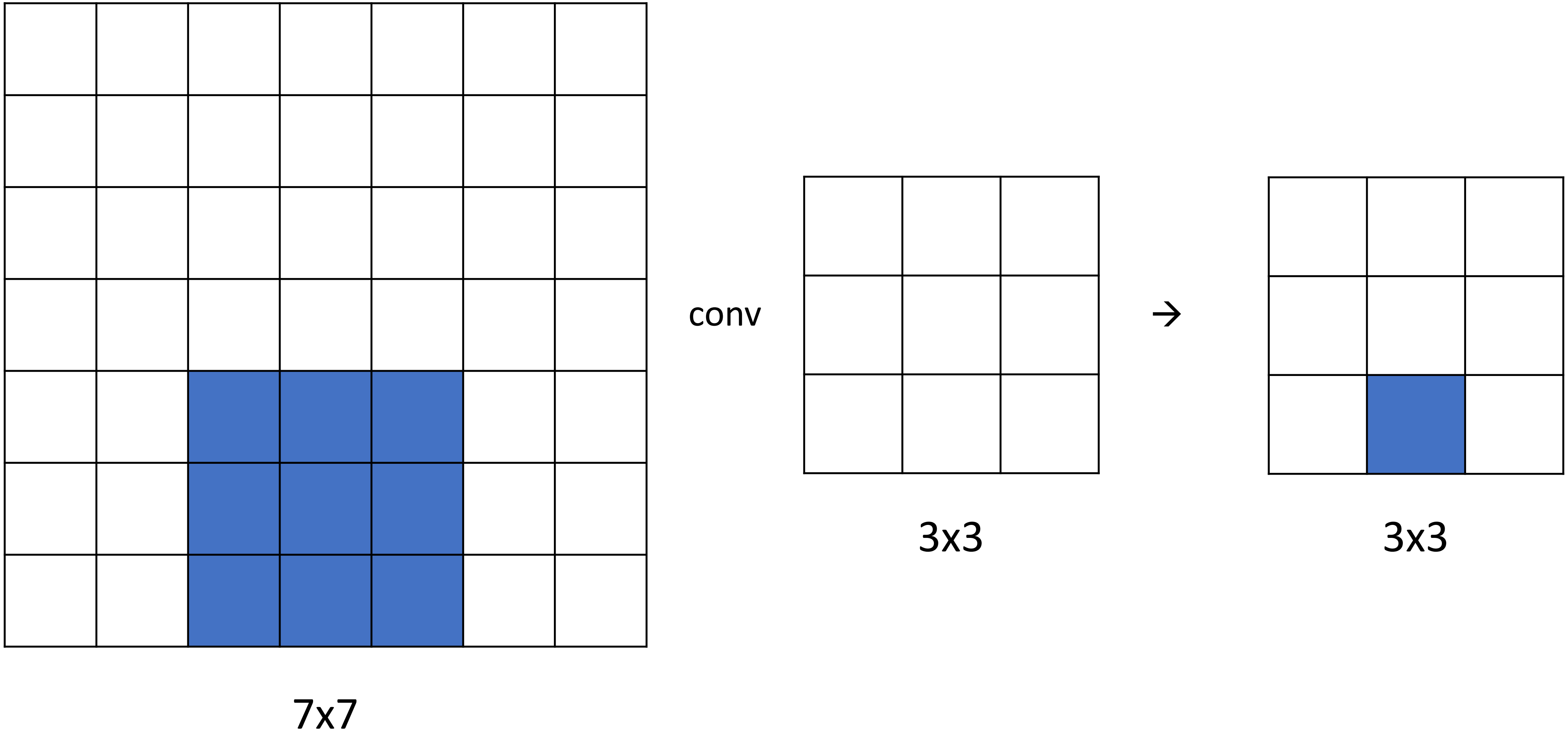


Example: Stride = 2

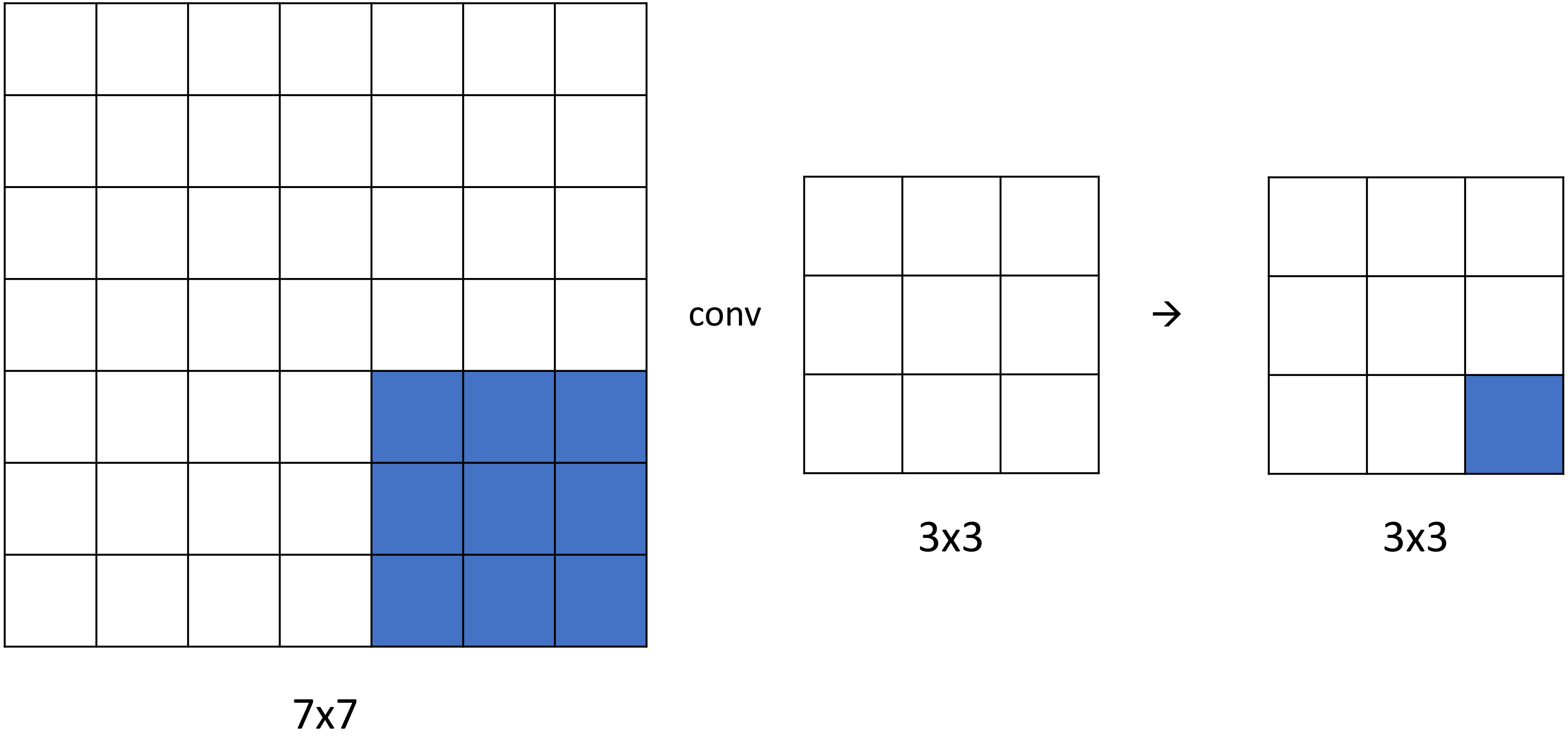


Brad Quinton, Scott Chin

Example: Stride = 2



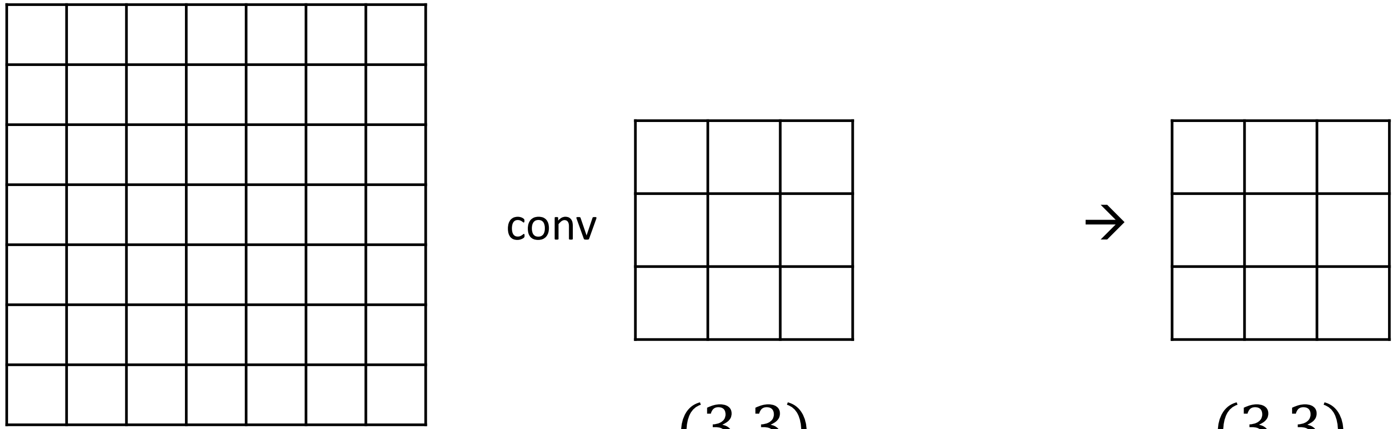
Example: Stride = 2



Brad Quinton, Scott Chin

Output Volume Size With Striding

- Input Volume: (h, w, c)
- Output Volume = $\left(\frac{h+2p-f}{s} + 1, \frac{w+2p-f}{s} + 1, K\right)$



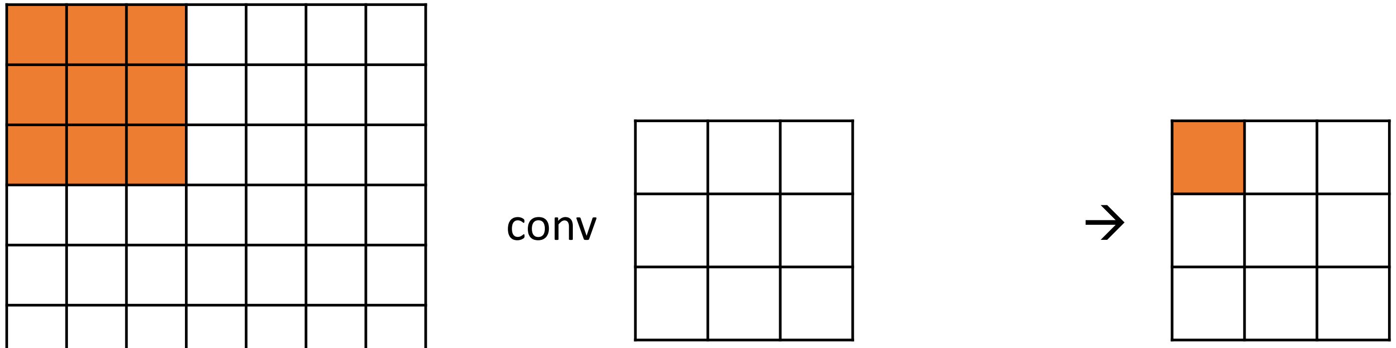
$(3,3)$ $(3,3)$

$$(h, w) = (7,7)$$
$$p = 0$$
$$s = 2$$

Brad Quinton, Scott Chin

Output Volume Size With Striding

- Input Volume: (h, w, c)
- Output Volume = $\left(\frac{h+2p-f}{s} + 1, \frac{w+2p-f}{s} + 1, K\right)$



$(h, w) = (7,7)$
$p = 0$
$s = 2$

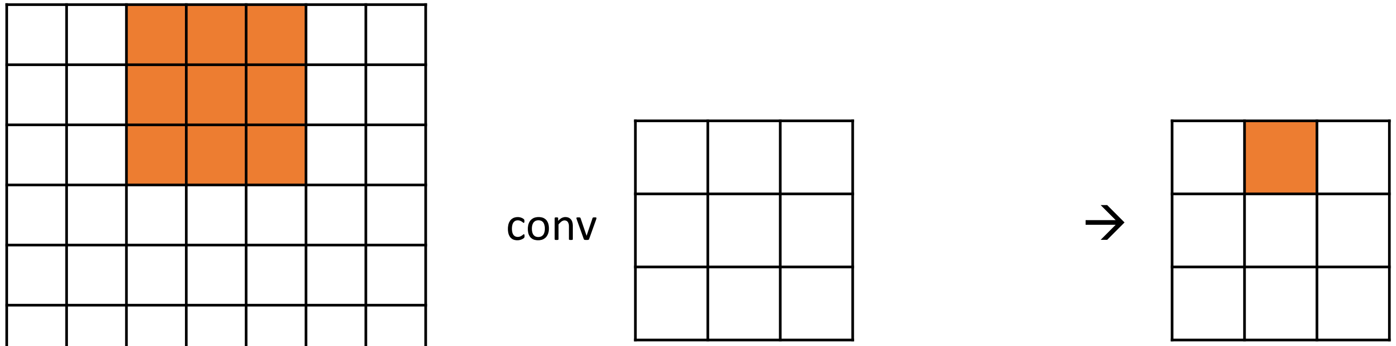
$(3,3)$

$(3,3)$

Brad Quinton, Scott Chin

Output Volume Size With Striding

- Input Volume: (h, w, c)
- Output Volume = $\left(\frac{h+2p-f}{s} + 1, \frac{w+2p-f}{s} + 1, K\right)$



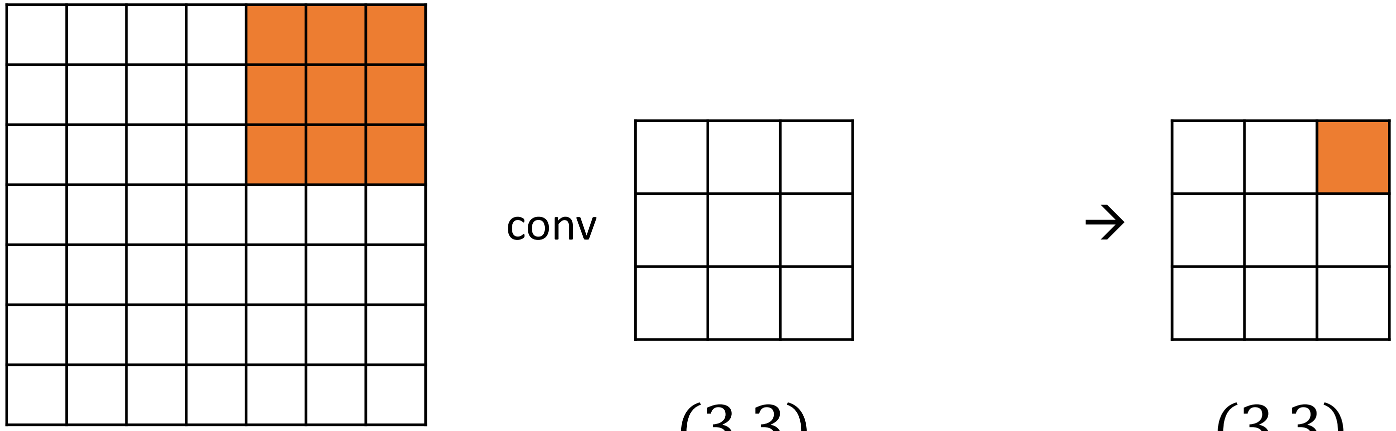
$(h, w) = (7,7)$
$p = 0$
$s = 2$

$(3,3)$

$(3,3)$

Output Volume Size With Striding

- Input Volume: (h, w, c)
- Output Volume = $\left(\frac{h+2p-f}{s} + 1, \frac{w+2p-f}{s} + 1, K\right)$



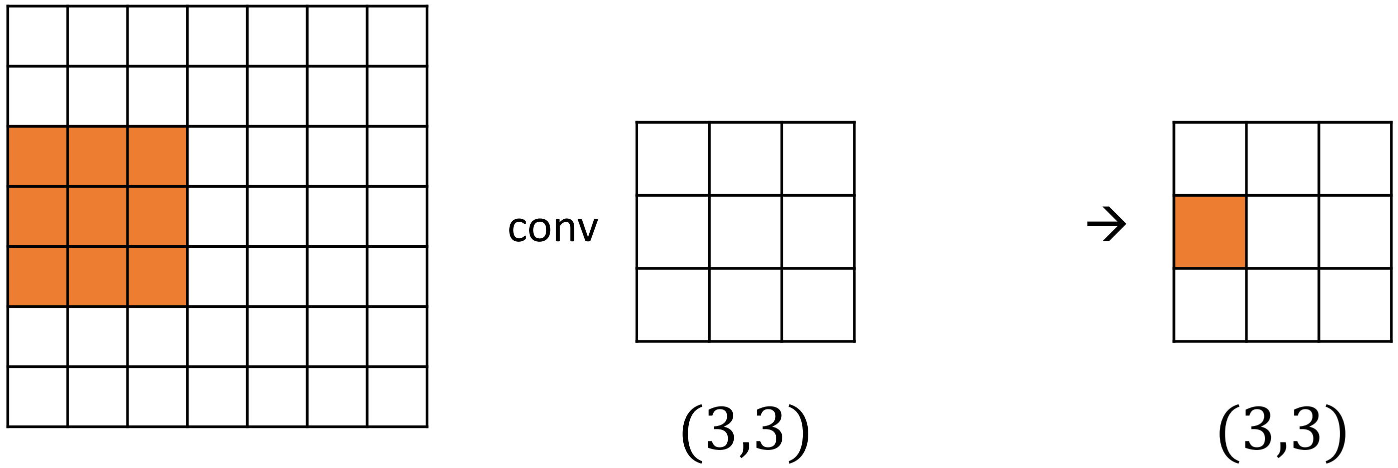
$(h, w) = (7,7)$
$p = 0$
$s = 2$

$(3,3)$ $(3,3)$

Brad Quinton, Scott Chin

Output Volume Size With Striding

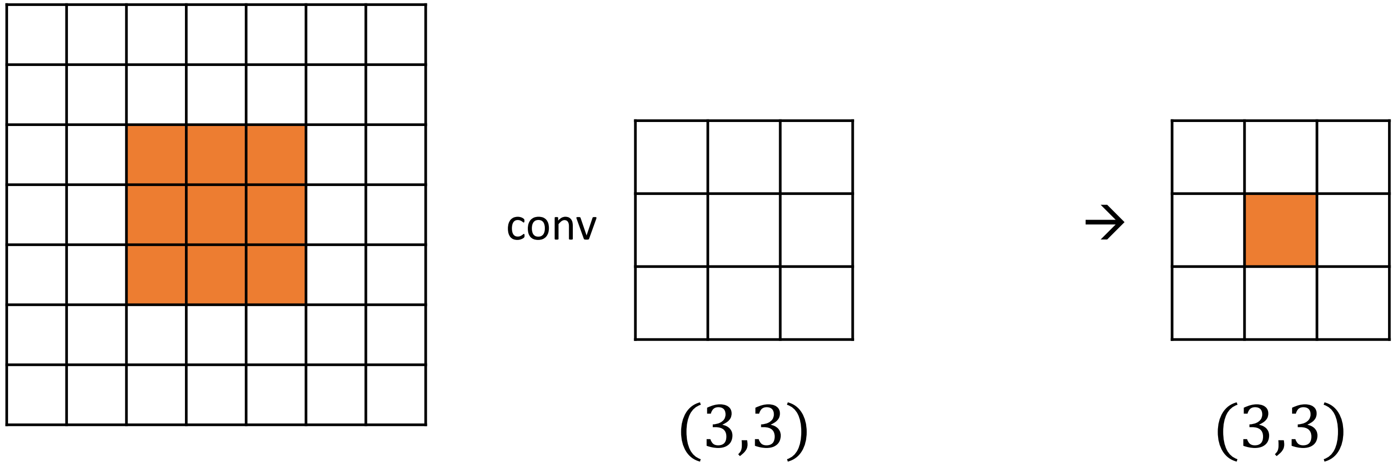
- Input Volume: (h, w, c)
- Output Volume = $\left(\frac{h+2p-f}{s} + 1, \frac{w+2p-f}{s} + 1, K\right)$



$(h, w) = (7,7)$
$p = 0$
$s = 2$

Output Volume Size With Striding

- Input Volume: (h, w, c)
- Output Volume = $\left(\frac{h+2p-f}{s} + 1, \frac{w+2p-f}{s} + 1, K\right)$

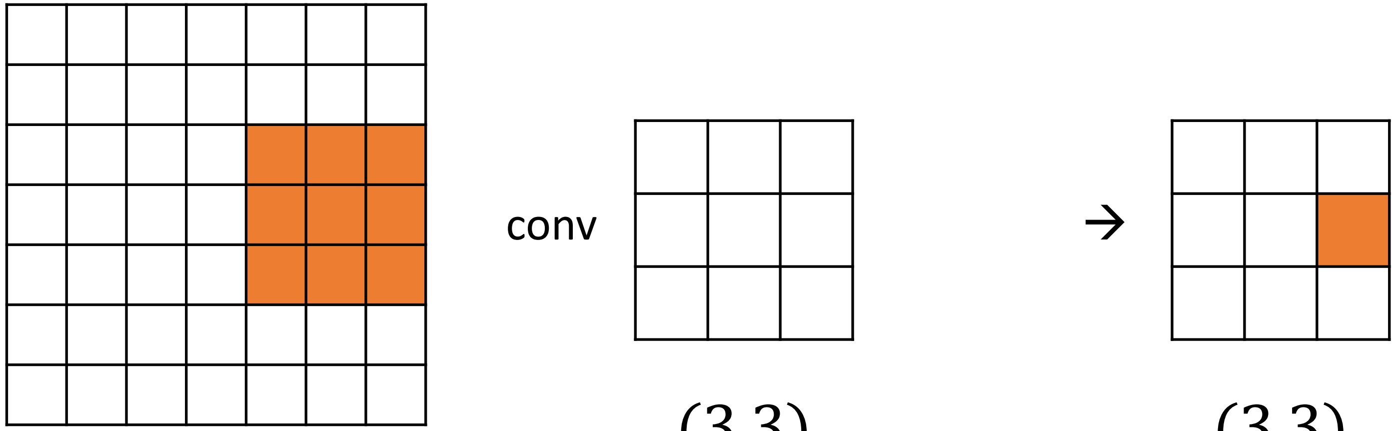


$(h, w) = (7,7)$
$p = 0$
$s = 2$

Brad Quinton, Scott Chin

Output Volume Size With Striding

- Input Volume: (h, w, c)
- Output Volume = $\left(\frac{h+2p-f}{s} + 1, \frac{w+2p-f}{s} + 1, K\right)$



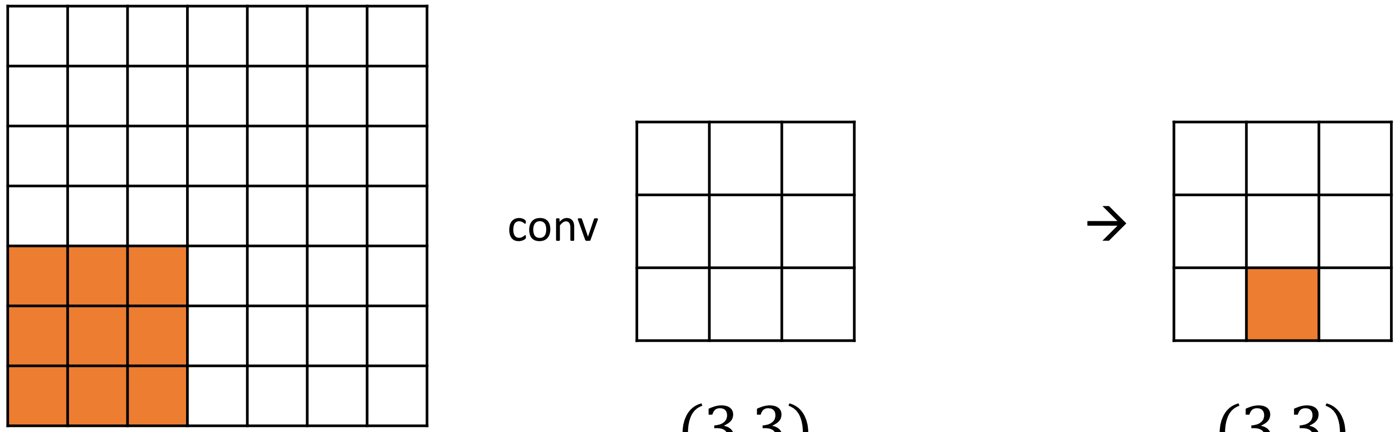
$(h, w) = (7,7)$
$p = 0$
$s = 2$

$(3,3)$

$(3,3)$

Output Volume Size With Striding

- Input Volume: (h, w, c)
- Output Volume = $\left(\frac{h+2p-f}{s} + 1, \frac{w+2p-f}{s} + 1, K\right)$



$(h, w) = (7,7)$
$p = 0$
$s = 2$

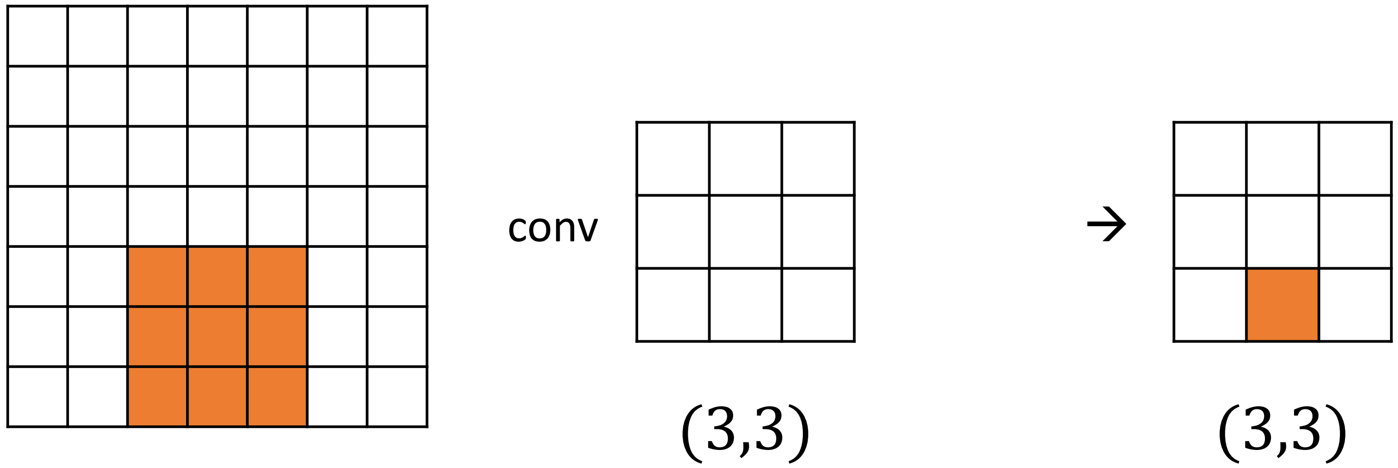
$(3,3)$

$(3,3)$

Brad Quinton, Scott Chin

Output Volume Size With Striding

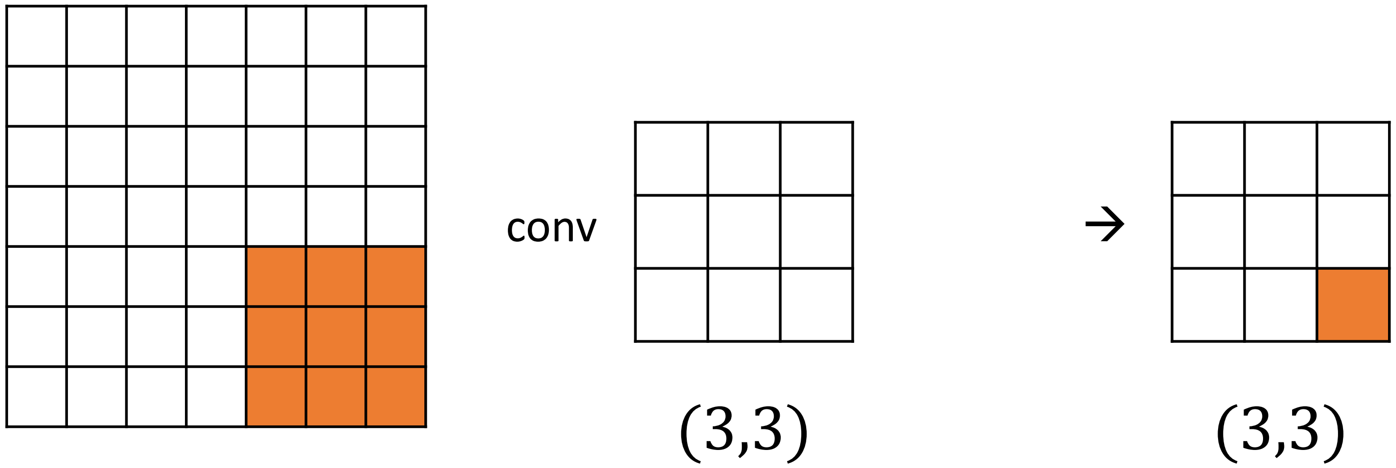
- Input Volume: (h, w, c)
- Output Volume = $\left(\frac{h+2p-f}{s} + 1, \frac{w+2p-f}{s} + 1, K\right)$



$(h, w) = (7,7)$
$p = 0$
$s = 2$

Output Volume Size With Striding

- Input Volume: (h, w, c)
- Output Volume = $\left(\frac{h+2p-f}{s} + 1, \frac{w+2p-f}{s} + 1, K\right)$



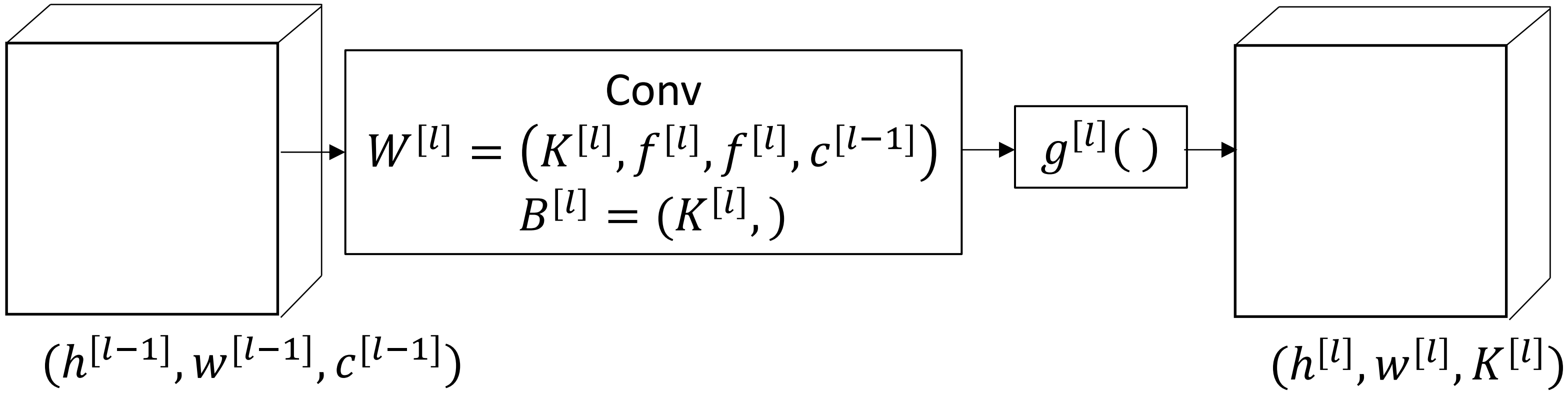
$(h, w) = (7,7)$
$p = 0$
$s = 2$

Why Stride > 1?

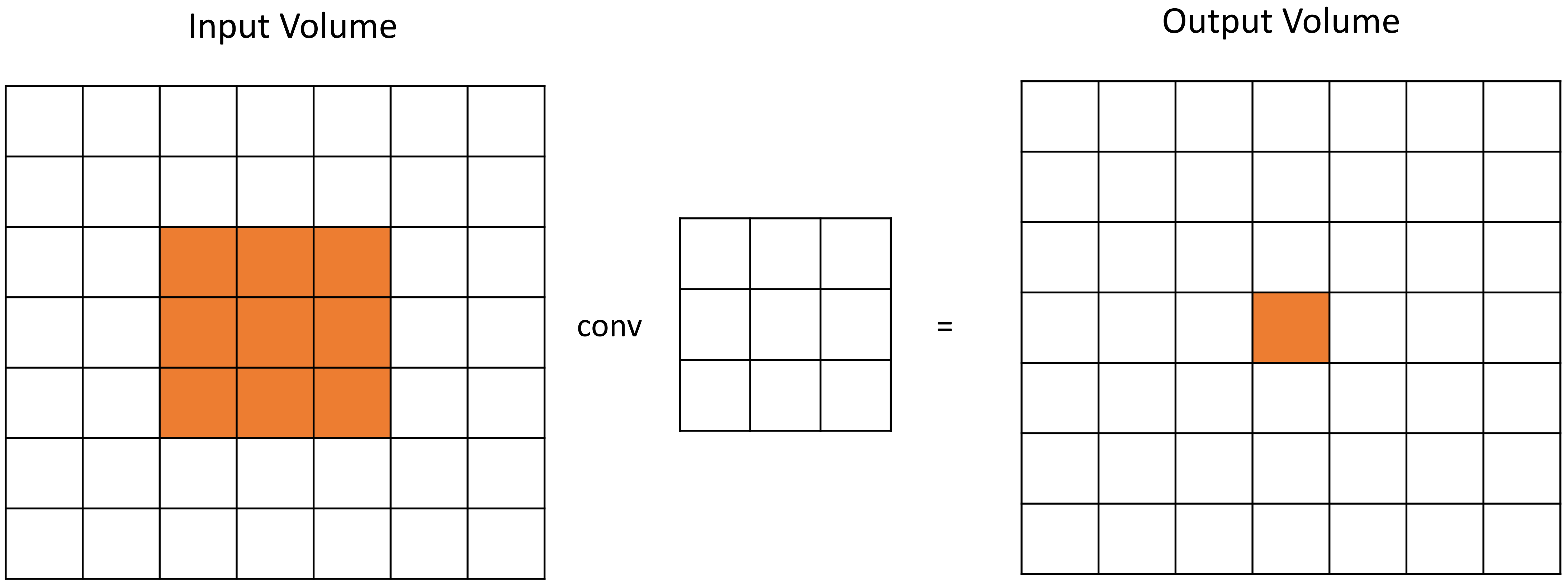
- A form of compression/downsampling of the feature map
- A way to shrink the volumes in a **controlled** fashion
- Shrinking volume is often necessary to control size before the final layer ← more on this soon
- Controls how quickly the receptive fields grows between layers

Final Summary of Convolutional Layer

- Hyperparameters
  - Number of filters K
  - Filter size (f, f)
  - Stride s
  - Padding p
- Input Volume
  - $(h^{[l-1]}, w^{[l-1]}, c^{[l-1]})$
- Output Volume
  - $\left(\frac{h^{[l-1]}+2p-f}{s} + 1, \frac{w^{[l-1]}+2p-f}{s} + 1, K\right)$
- # Learned parameters
  - $K * \left(f * f * c^{[l-1]} + 1\right)$



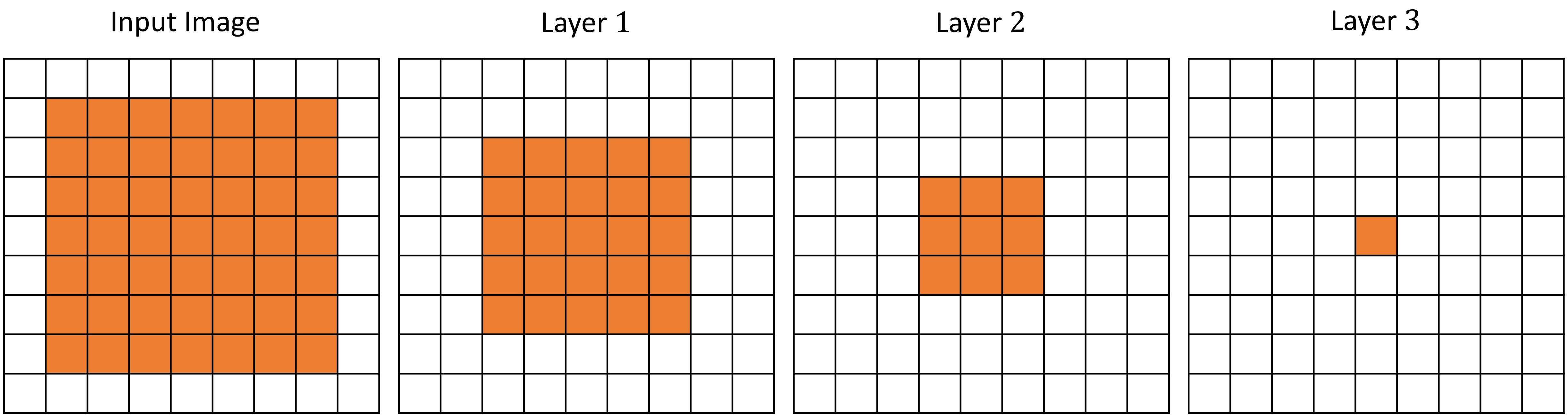
Receptive Fields



- Assuming a 3x3 filter in this layer
- Each output element "sees" a 3x3 region of the input

Brad Quinton, Scott Chin

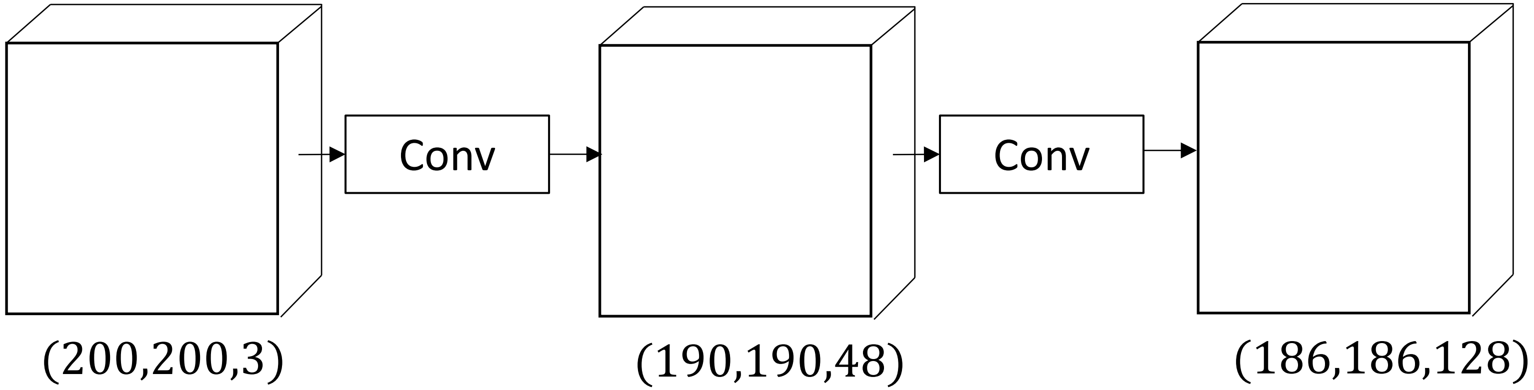
Receptive Fields



- Assuming 3x3 filters in all layers
- Each output element of each layer “sees” a 3x3 region of its input
- You can work backwards to see how much an output element ”sees” w.r.t outputs of earlier layer

Final Layers of a CNN

- Ok if we stack these, we still end up with a volume at the end
- Need a way to make a prediction using the final volume (e.g. final collection of feature activation maps)



Final Layers of a CNN Image Classifier

- Flatten the final volume
- Use one or more fully connected layer
- Final volume must therefore be of a manageable size
- Can think of conv layers as feature extractors

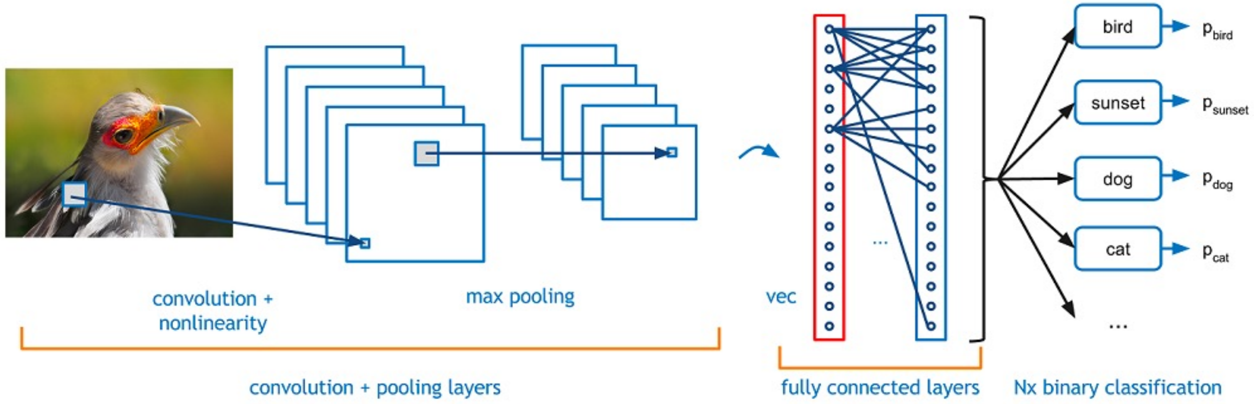
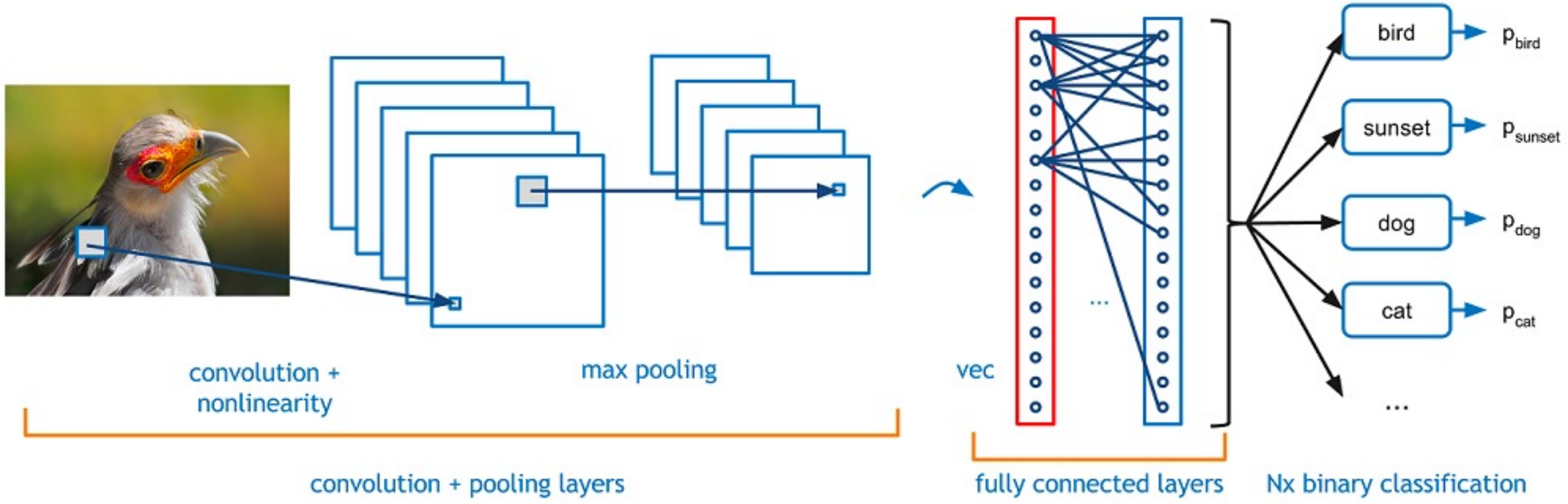


Image source: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Aside – One interpretation: Conv Layers are Feature Extractors



- Compress the image into a “signature”. Use the signature for classification
- Learn structure from unstructured data

Image source: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

How do we control the final volume size?

- When using only Convolutional layers, your final volume depends on
- Stride, padding of your conv layers,

Pooling Layer

Example: Max Pooling

- Output is max value within each region

1	3	10	0
8	4	2	9
7	2	1	3
1	1	0	4

Max Pooling
$(f, f) = (2,2)$
Stride: 2

→

Example: Max Pooling

- Output is max value within each region

1	3	10	0
8	4	2	9
7	2	1	3
1	1	0	4

<u>Max Pooling</u>
$(f, f) = (2,2)$
Stride: 2

→

8	

Brad Quinton, Scott Chin

Example: Max Pooling

- Output is max value within each region

1	3	10	0
8	4	2	9
7	2	1	3
1	1	0	4

<u>Max Pooling</u>

$(f, f) = (2,2)$

Stride: 2

→

8	10

Example: Max Pooling

- Output is max value within each region

1	3	10	0
8	4	2	9
7	2	1	3
1	1	0	4

<u>Max Pooling</u>
$(f,f) = (2,2)$
Stride: 2

→

8	10
7	

Brad Quinton, Scott Chin

Example: Max Pooling

- Output is max value within each region

1	3	10	0
8	4	2	9
7	2	1	3
1	1	0	4

<u>Max Pooling</u>

$(f, f) = (2,2)$

Stride: 2

→

8	10
7	4

Example: Max Pooling

- Output is max value within each region

1	3	10	0
8	4	2	9
7	2	1	3
1	1	0	4

Max Pooling

$(f, f) = (2,2)$

Stride: 2

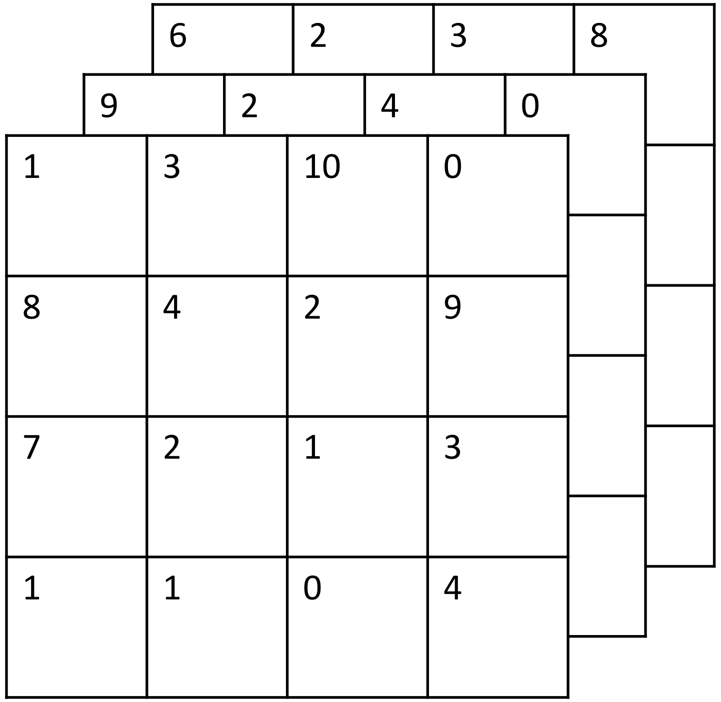
→

8	10
7	4

After pooling volume size has been reduced!

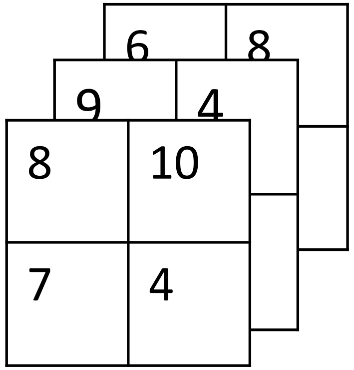
Pool each channel independently

- Apply this process to each activation map independently
- Does not change your channels size, only spatial dimensions of (h, w)



Max Pooling
$(f, f) = (2,2)$
Stride: 2

→



Max Pooling Intuition

- Compressing the data. Discarding all but the "strongest" signal
- Max Pooling adds a bit of flexibility to feature detection in the form of tolerance to translation (jitter)
- Once we know that a specific feature is in the original input volume, it's exact location is not as important as its relative location to other features.

1	3	10	0
8	4	2	9
7	2	1	3
1	1	0	4

<u>Max Pooling</u>

$(f, f) = (2{,}2)$

Stride: 2

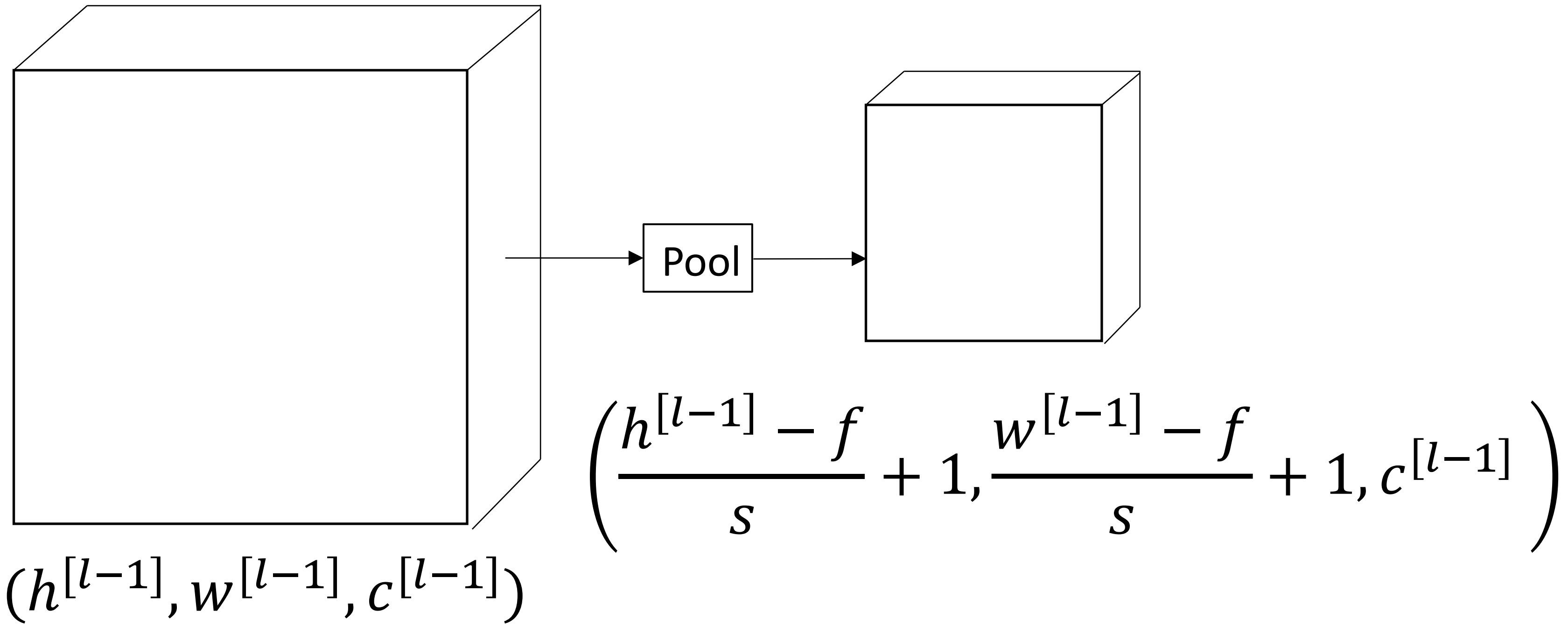
→

8	10
7	4

Brad Quinton, Scott Chin

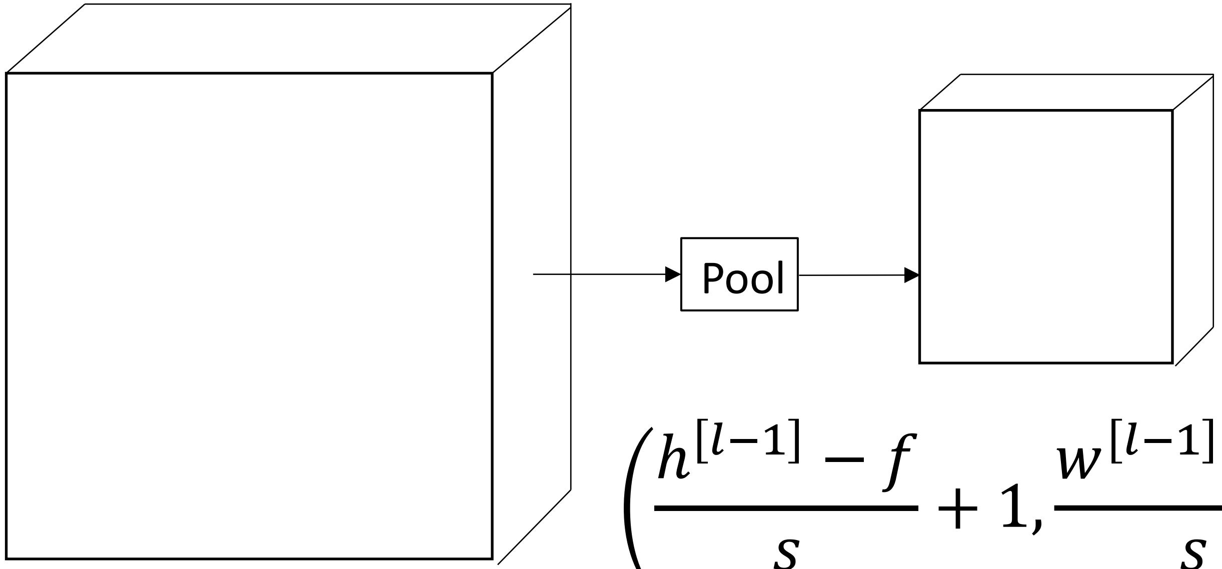
Pooling

- Hyperparameters
  - Pooling function
  - Pool size (f, f)
  - Stride s



- No learned parameters!
- Reduces spatial dimensions but NOT channel dimension

Pooling



$$\left(h^{[l-1]}, w^{[l-1]}, c^{[l-1]}\right)$$

$$\left(\frac{h^{[l-1]} - f}{s} + 1, \frac{w^{[l-1]} - f}{s} + 1, c^{[l-1]}\right)$$

Usually, $f = s$

$$\left(\frac{h^{[l-1]}}{s}, \frac{w^{[l-1]}}{s}, c^{[l-1]}\right)$$

- Hyperparameters
  - Pooling function
  - Pool size (f, f)
  - Stride s
- No learned parameters!
- Reduces spatial dimensions but NOT channel dimension

Example: Average Pooling

Example: Average Pooling

- Output is **average** value within each region

1	3	10	1
8	4	4	9
7	2	4	3
1	1	0	4

<u>Avg Pooling</u>
$(f, f) = (2,2)$
Stride: 2

→

Brad Quinton, Scott Chin

Example: Average Pooling

- Output is **average** value within each region

1	3	10	1
8	4	4	9
7	3	4	3
1	1	0	5

Avg Pooling
$(f, f) = (2,2)$
Stride: 2

→

4	

Brad Quinton, Scott Chin

Example: Average Pooling

- Output is **average** value within each region

1	3	10	1
8	4	4	9
7	3	4	3
1	1	0	5

<u>Avg Pooling</u>
$(f, f) = (2,2)$
Stride: 2

→

4	6

Brad Quinton, Scott Chin

Example: Average Pooling

- Output is **average** value within each region

1	3	10	1
8	4	4	9
7	3	4	3
1	1	0	5

<u>Avg Pooling</u>
$(f, f) = (2,2)$
Stride: 2

→

4	6
3	

Brad Quinton, Scott Chin

Example: Average Pooling

- Output is **average** value within each region

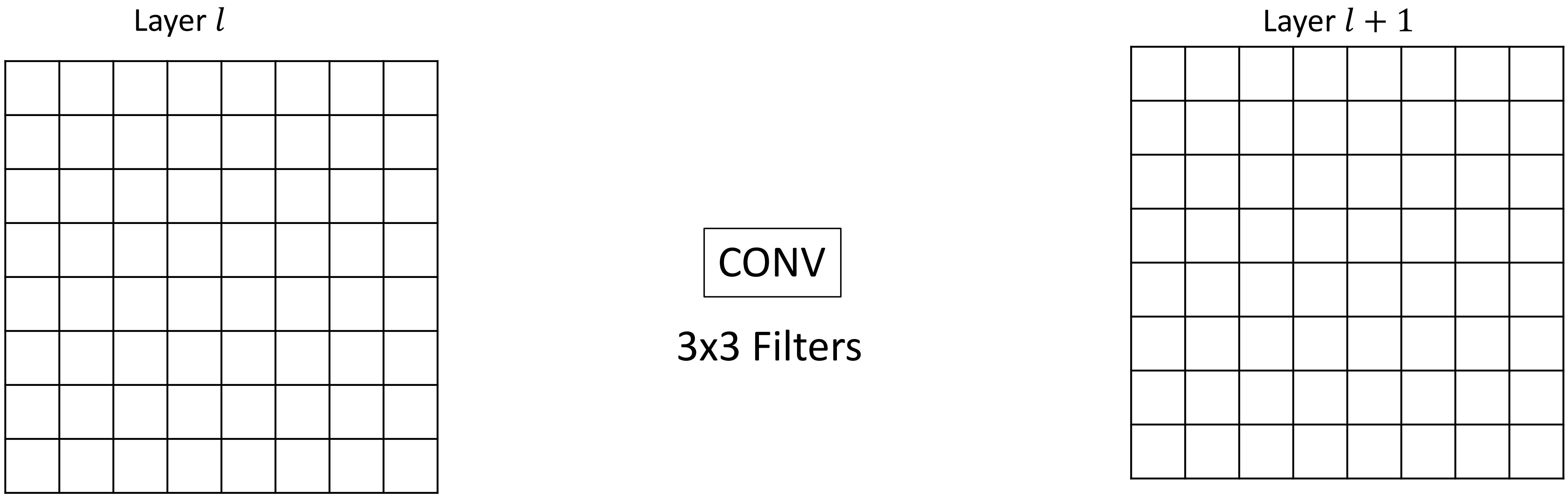
1	3	10	1
8	4	4	9
7	3	4	3
1	1	0	5

Avg Pooling
$(f, f) = (2,2)$
Stride: 2

→

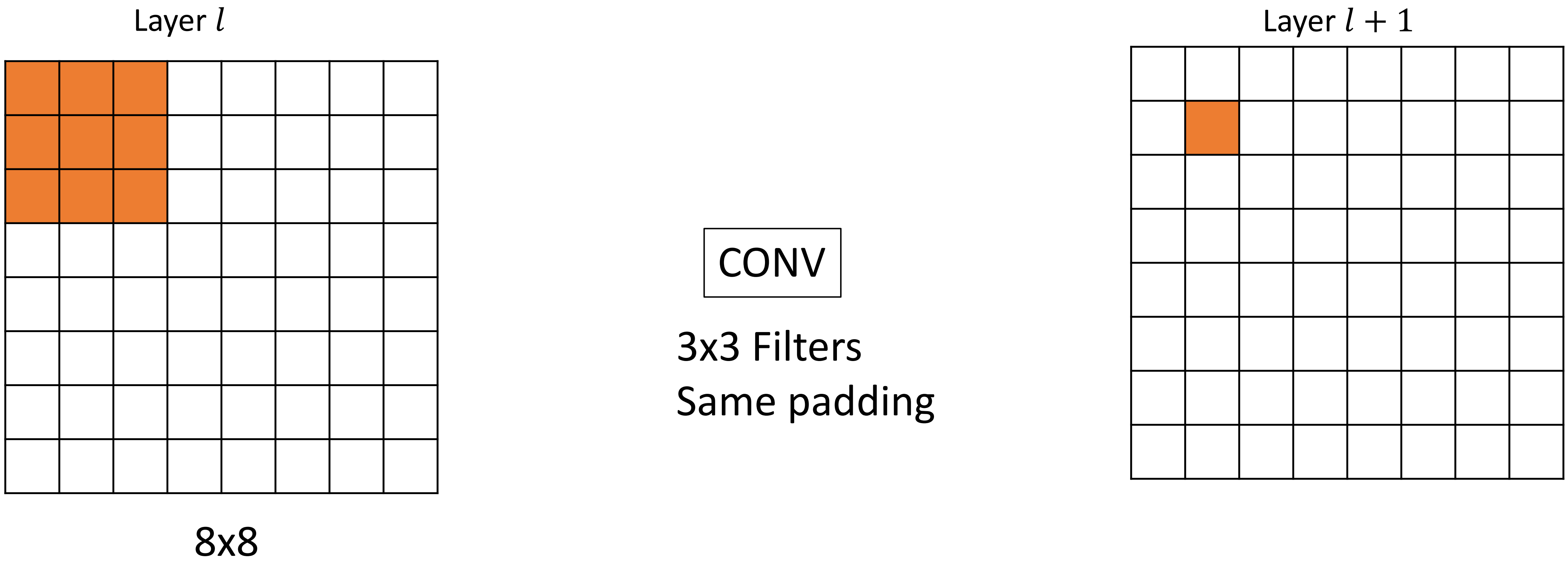
4	6
3	3

Pooling and Receptive Fields

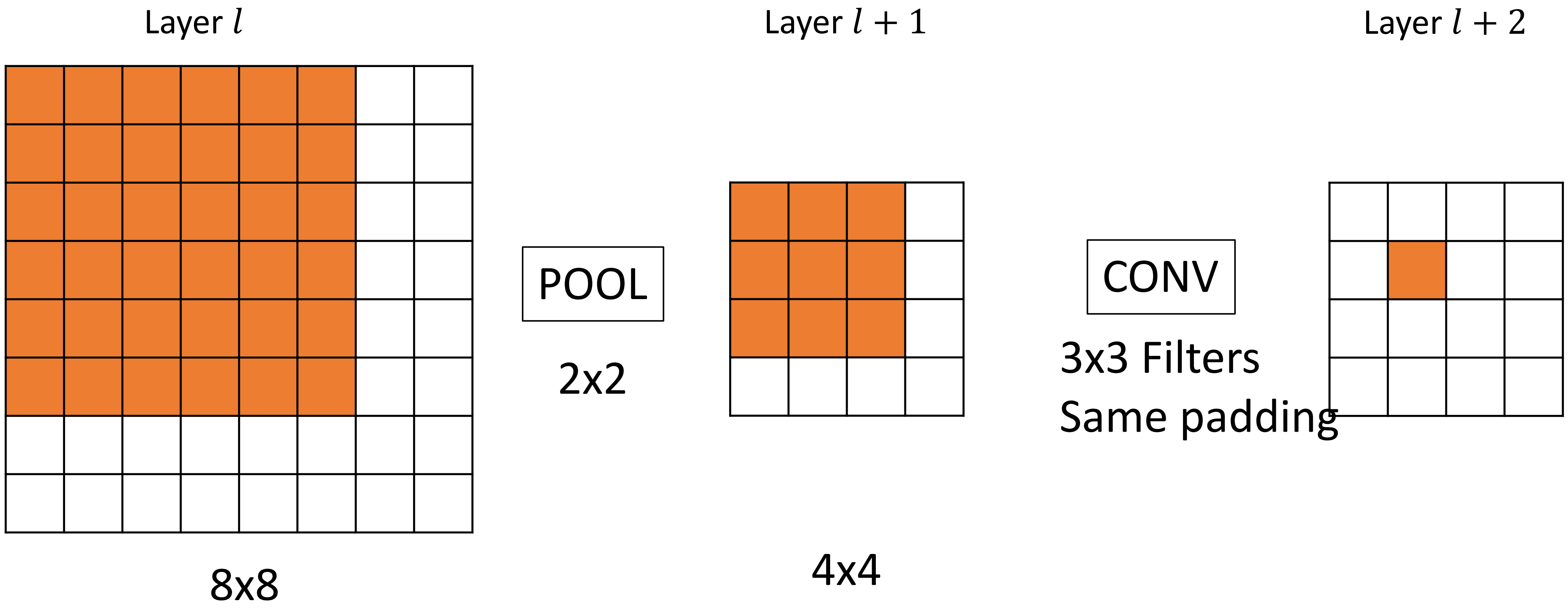


Brad Quinton, Scott Chin

Pooling and Receptive Fields



Pooling and Receptive Fields



Summary of Pooling Layer

- Reduces the size of the feature maps (i.e. volume size)
  - Controlled shrinkage of volume is required for final fully connected layer
- Pooling operation is specified → not learned (no parameters)
- Most often, stride is equal to filter size. $s = f$
- Max Pooling adds some translation tolerance. But you need to think about whether this is what you want for your application
- Operates over each activation map **independently**. Does not reduce channel size, only spatial dimensions
- People generally use Max Pooling over Average Pooling

Learning Objectives

- Understand how a Convolutional Layer works
- Understand how we can build a Convolutional Neural Network using Convolutional and Fully-Connected Layers
- Understand how we can ”transform” a Convolutional Layer to a Fully-Connected Layer and vice-versa
- Understand what stride and padding are
- Understand how Pooling Layers work

Brad Quinton, Scott Chin