Intro to Convolutional Neural Networks

Deep Learning

Brad Quinton, Scott Chin

Learning Objectives

- Introduction to Convolutional Neural Networks and their applications
- Working towards an Image Classification Model
- Discuss inefficiencies of using a fully-connected neural network for image applications
- Become familiar with the Convolution operation and Convolutional Filters for 2D and 3D data
- Understand edge detection using manually chosen filter values

Brad Quinton, Scott Chin

What is a Convolutional Neural Network

Convolutional Neural Networks (CNNs, ConvNets) are a class of Neural Networks typically used for image analysis and computer vision.

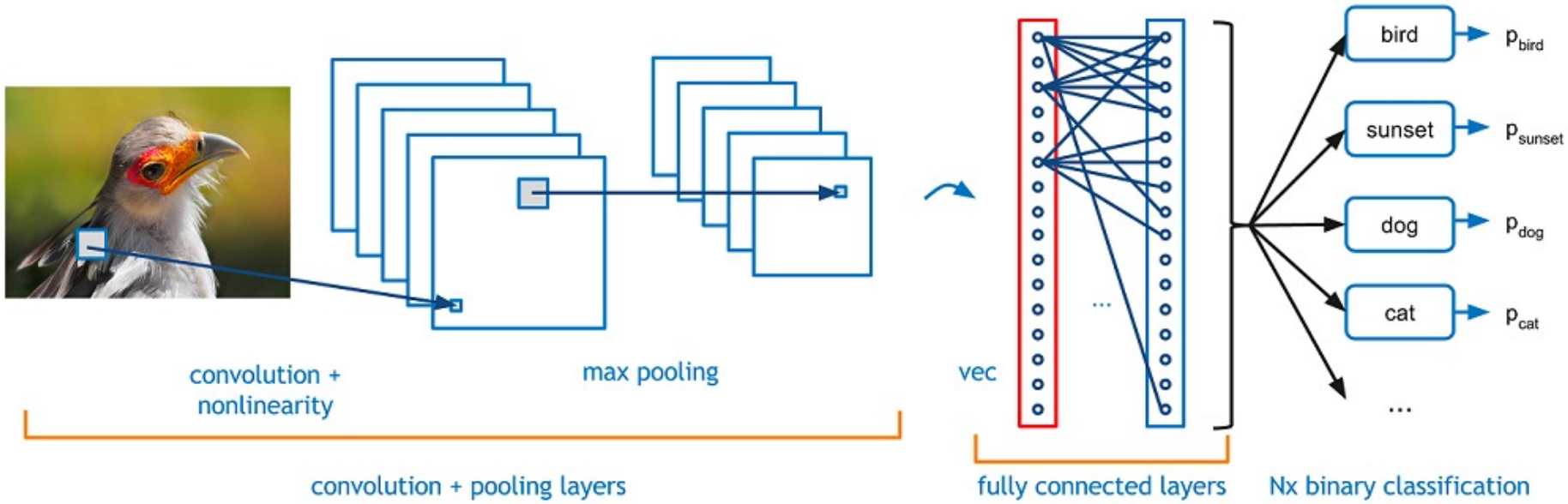
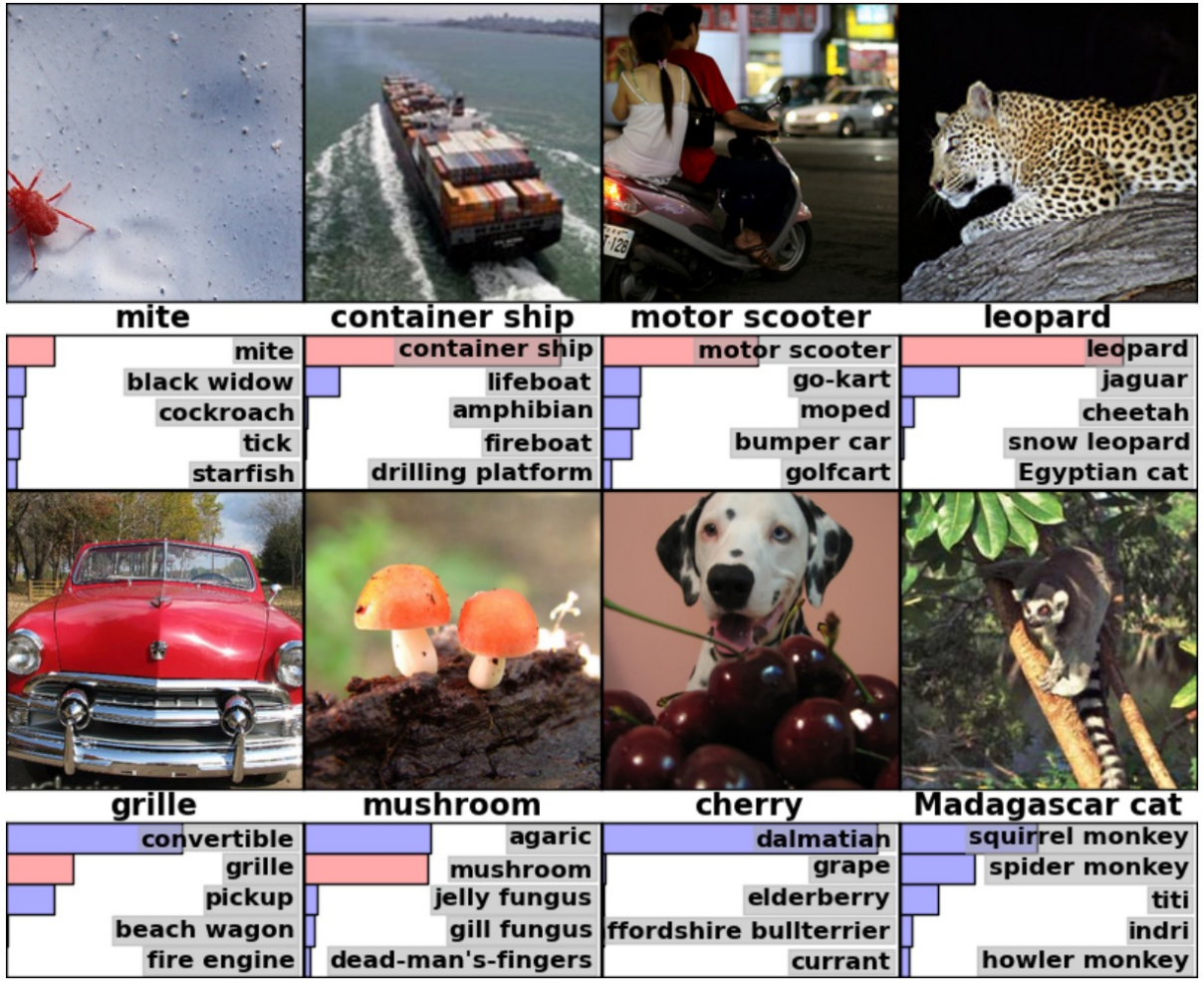


Image source: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

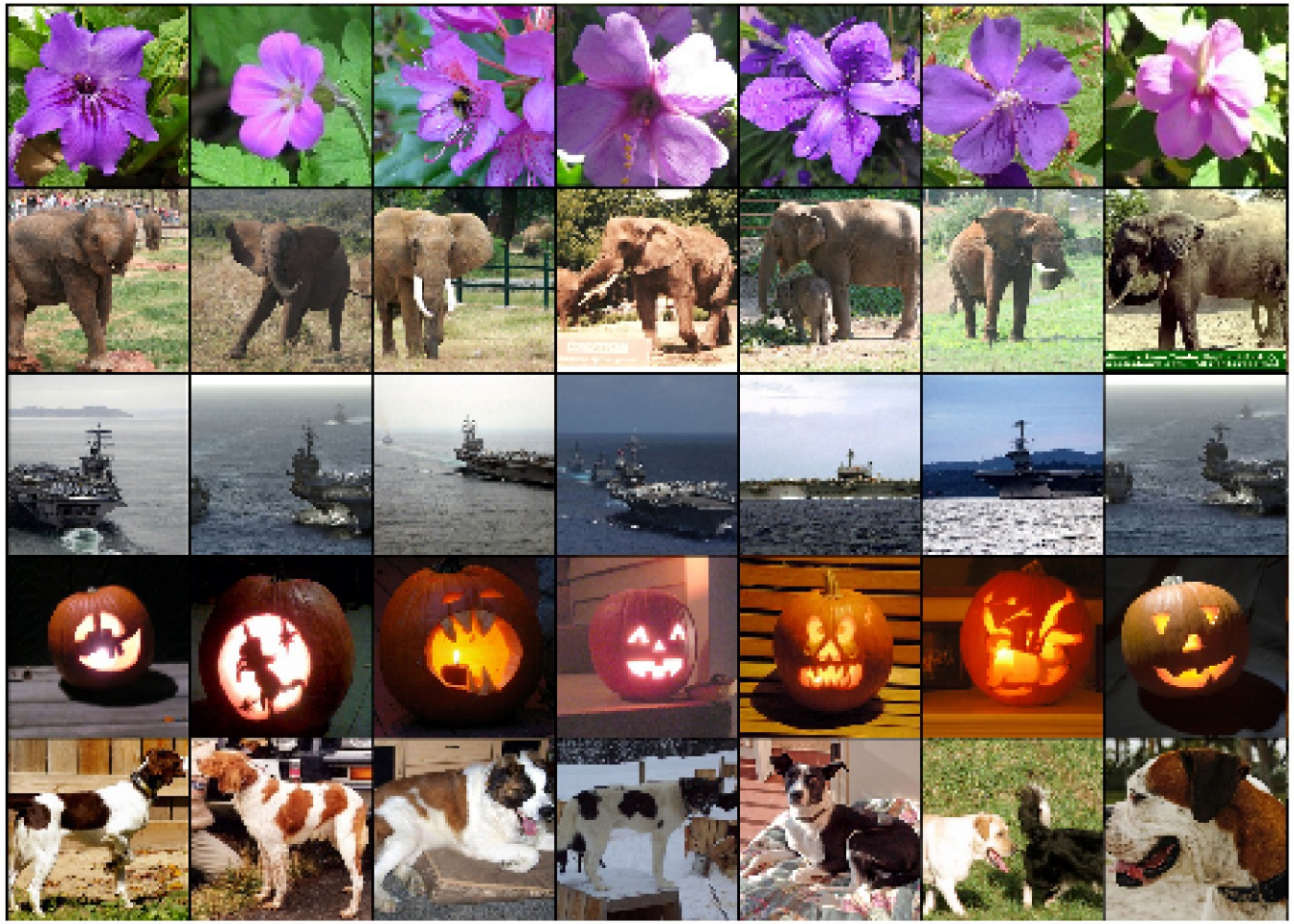
Brad Quinton, Scott Chin

CNN Applications – Image Classification



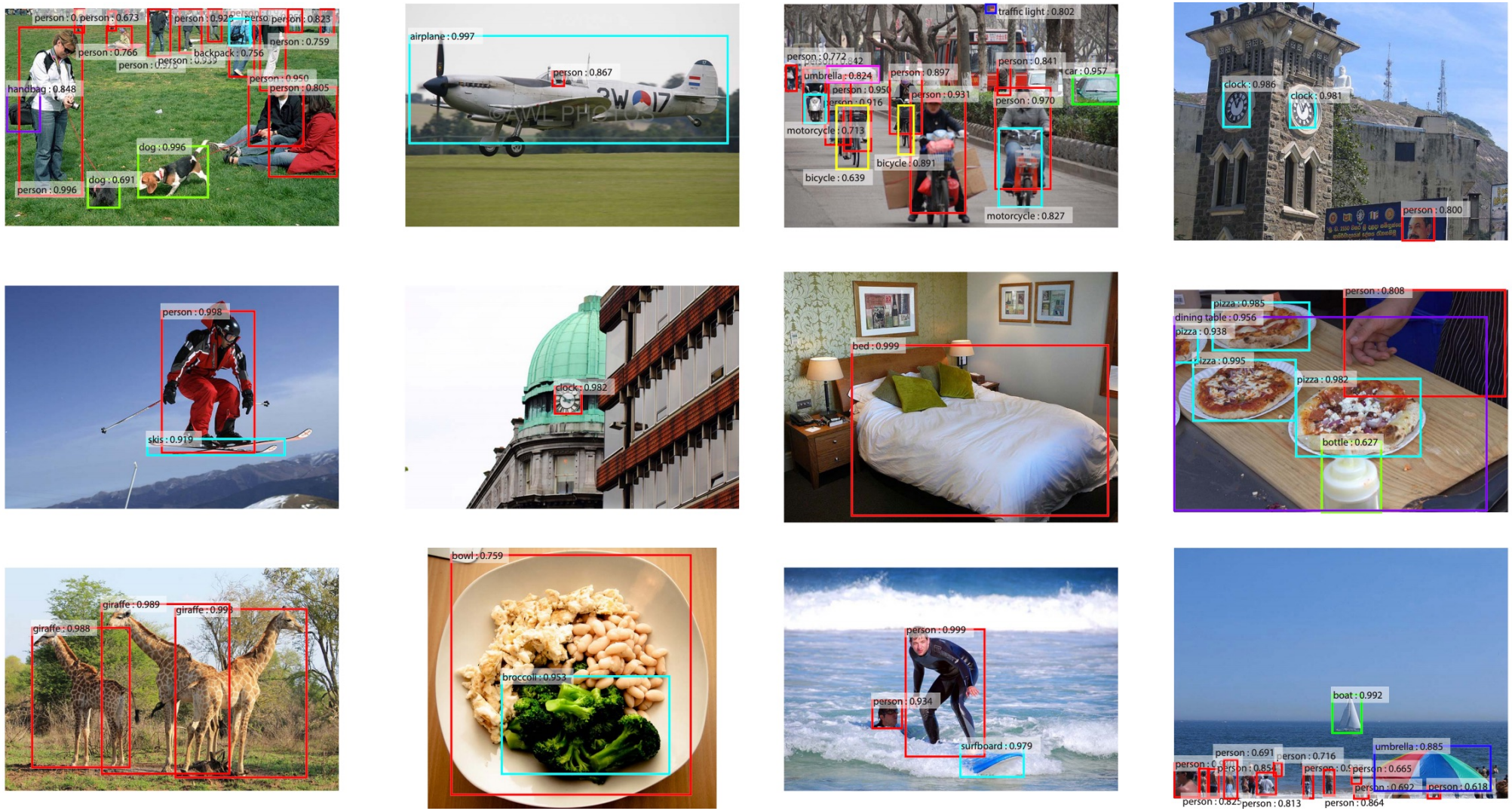
"ImageNet Classification with Deep Convolutional Neural Networks", Krizhevsky, Sutskever, Hinton, 2012, https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

CNN Applications – Retrieval



"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

CNN Applications - Object Detection



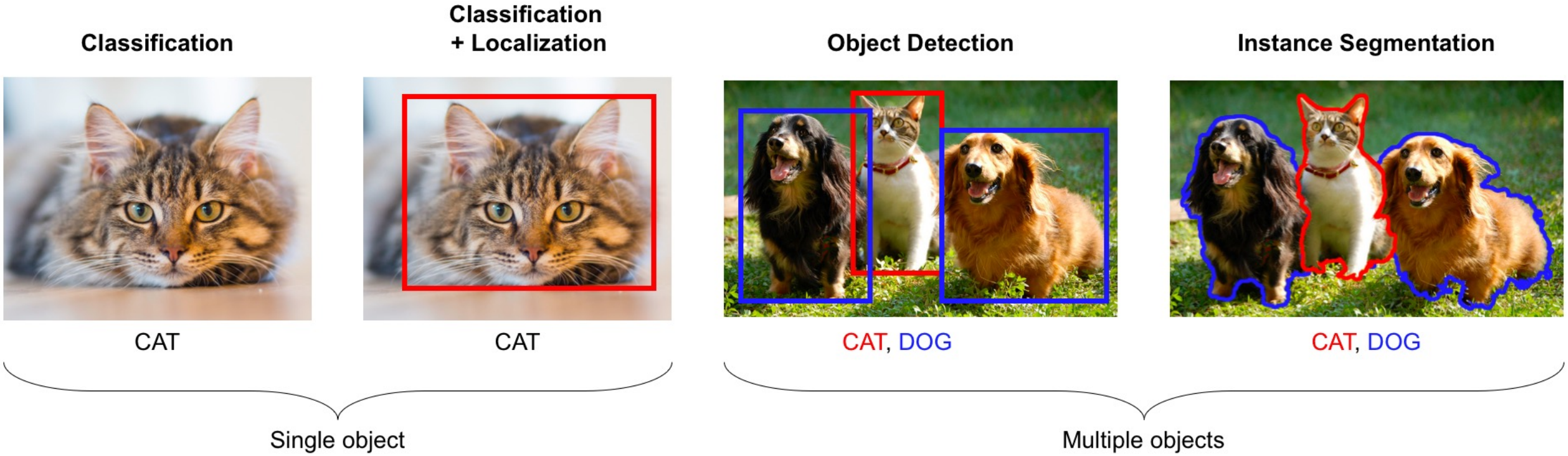
"Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", Ren, He, Girshick, Sun, 2015, https://arxiv.org/abs/1506.01497


LEFT REARWARD VEHICLE CAMERA
MEDIUM RANGE VEHICLE CAMERA
RIGHT REARWARD VEHICLE CAMERA
MOTION FLOW
LANE LINES
LANE LINES
ROAD FLOW
IN-PATH OBJECTS
ROAD LIGHTS
OBJECTS
ROAD SIGNS

CNN Applications – Object Segmentation

slide 8/107 "Fully Convolutional Networks for Semantic Segmentation", Long, Shelhamer, Darrell, 2015, https://people.eecs.berkeley.edu/~jonlong/long_shelhamer_fcn.pdf Brad Quinton, Scott Chin

CNN Applications



Brad Quinton, Scott Chin

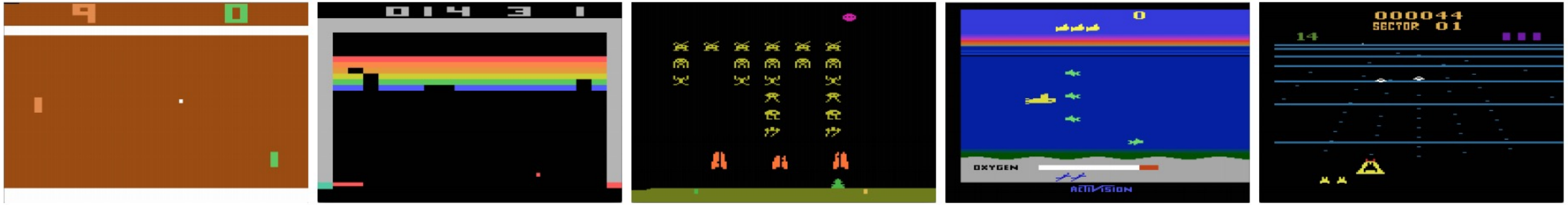
CNN Applications – Pose Estimation

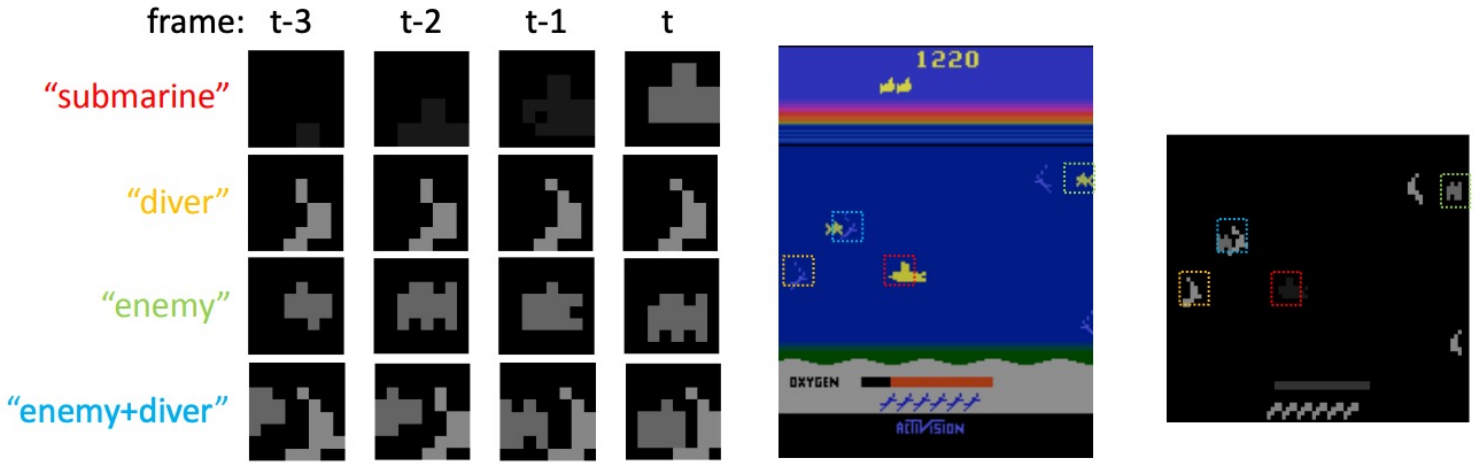
"DeepPose: Human Pose Estimation via Deep Neural Networks", Toshev, Szededy, 2013, https://arxiv.org/abs/1312.4659

CNN Applications – Pose Estimation

"Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields", Cao et al., 2017, https://arxiv.org/pdf/1611.08050.pdf

CNN Applications – Vision Based Reinforcement Learning



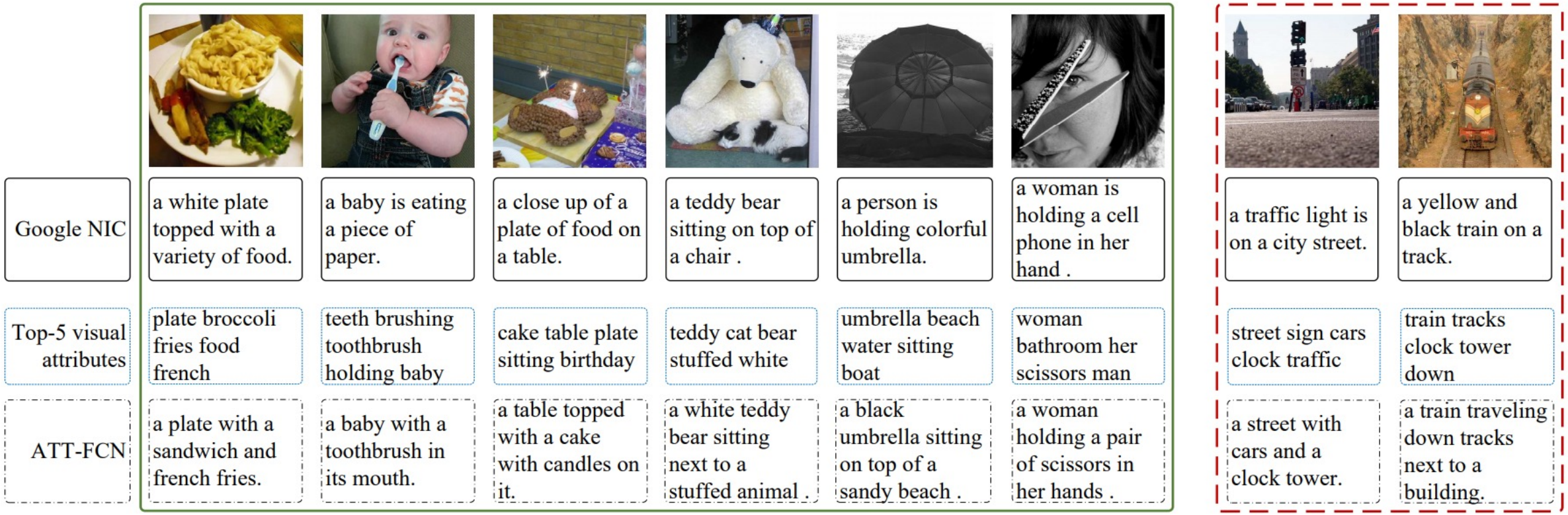


"Playing Atari with Deep Reinforcement Learning", Mnih et al., 2013, https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf

"Action-Conditional Video Prediction using Deep Networks in Atari Games", Oh et al., 2015, https://arxiv.org/pdf/1507.08750.pdf

Brad Quinton, Scott Chin

CNN Applications – Image Captioning



"Image Captioning with Semantic Attention", You et al., 2016,
"http://openaccess.thecvf.com/content_cvpr_2016/papers/You_Image_Captioning_With_CVPR_2016_paper.pdf"

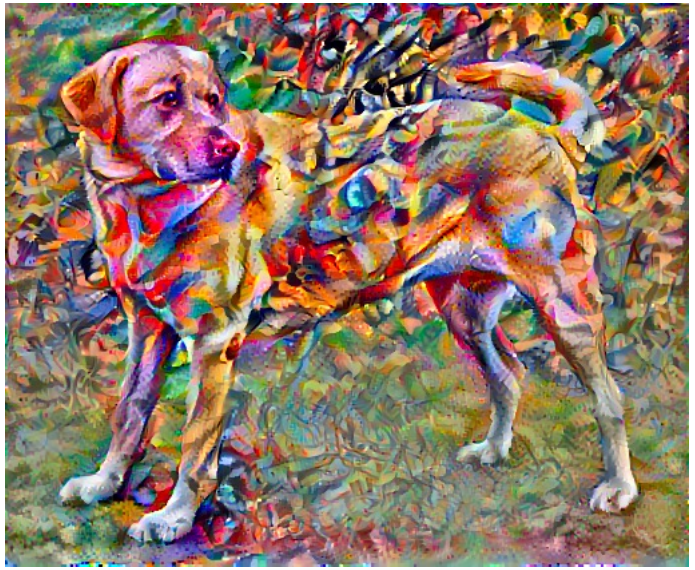
CNN Applications – More Object Detection

Brad Quinton, Scott Chin

CNN Applications – More Object Detection

Brad Quinton, Scott Chin

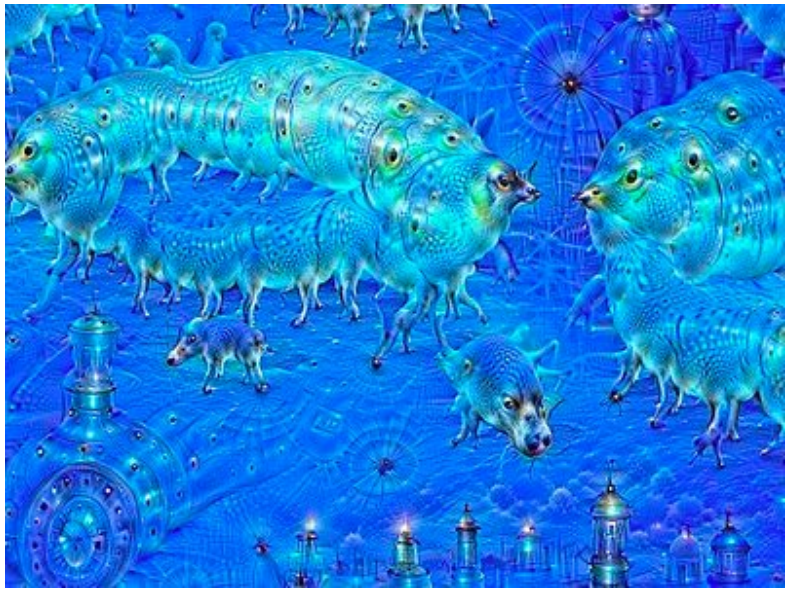
CNN Applications - Neural Style Transfer



Neural Style Transfer TensorFlow Tutorial, https://www.tensorflow.org/tutorials/generative/style_transfer

Deep Dream

Source



10 Iterations

50 Iterations

https://en.wikipedia.org/wiki/DeepDream

https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

Brad Quinton, Scott Chin

Deep Dream Generator

https://deepdreamgenerator.com/

Brad Quinton, Scott Chin

Image Classification with Fully-Connected Neural Network

Brad Quinton, Scott Chin

Image Data is Unstructured Data

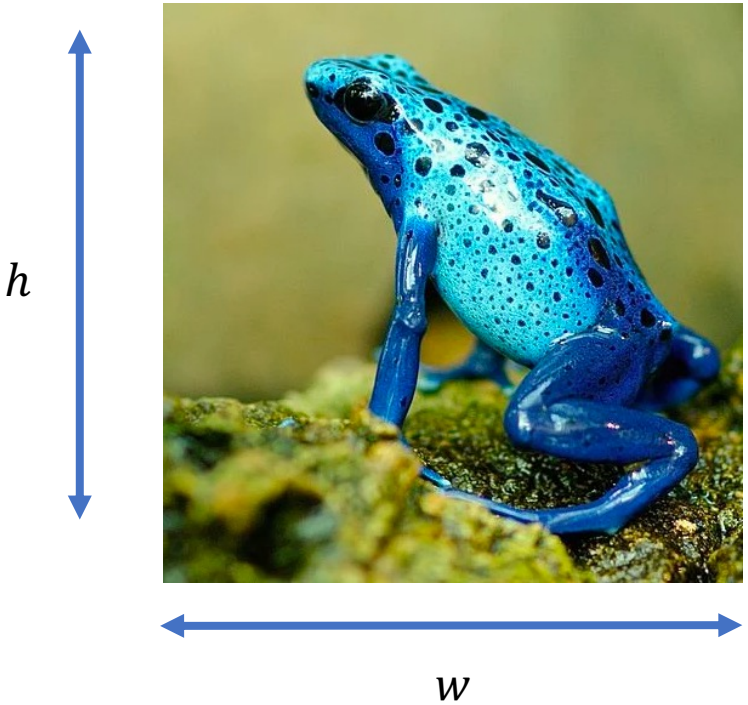
- Structured Data
  - Each feature represents a specific piece of information across all samples
  - Data that can be tabulated (e.g. summarized in a spreadsheet)
  - E.g. Housing data, maybe one feature is house price. This feature represents the same formal “thing” amongst all samples

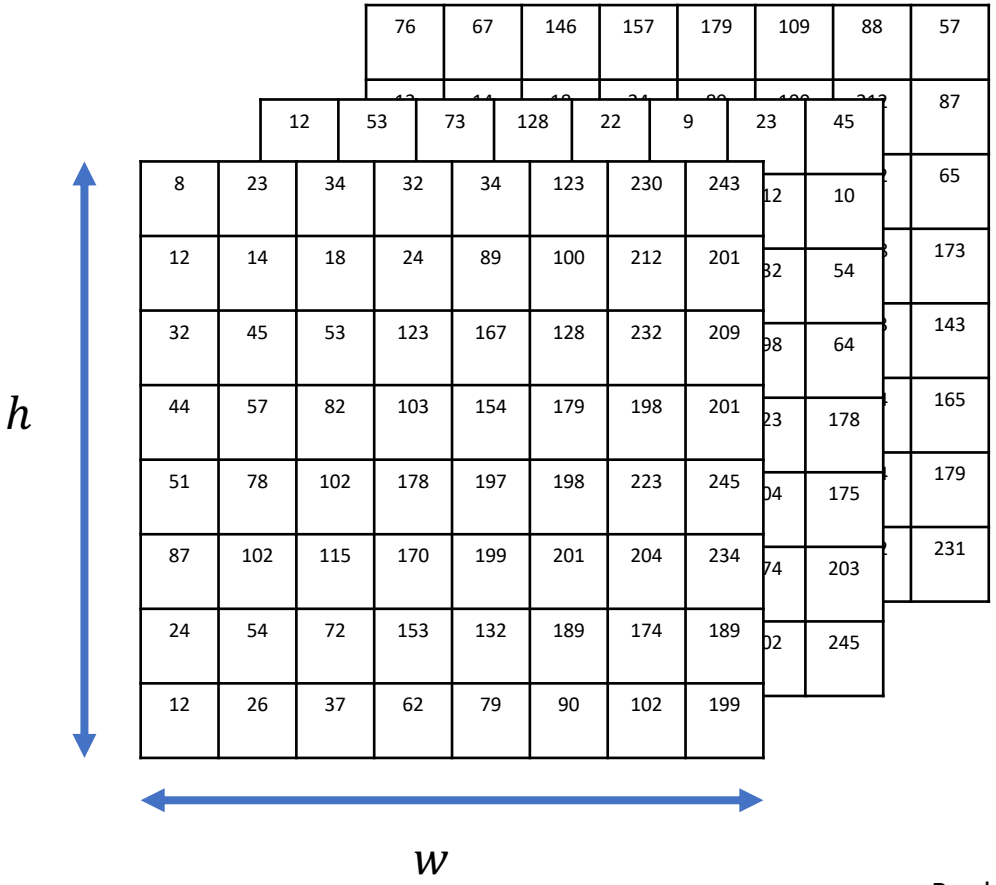
- Unstructured Data
  - No real relationship between samples of the same feature.
  - For example, considering an image, each pixel is a feature. But there is no formal relationship between pixel (233, 246) amongst all the samples

Brad Quinton, Scott Chin

Recall - Images as Input Data

- For color image, model as three channels (RGB) → $(h, w, 3)$
- Each array value is still 0-255
- R=255, G=153, B=0 → Orange



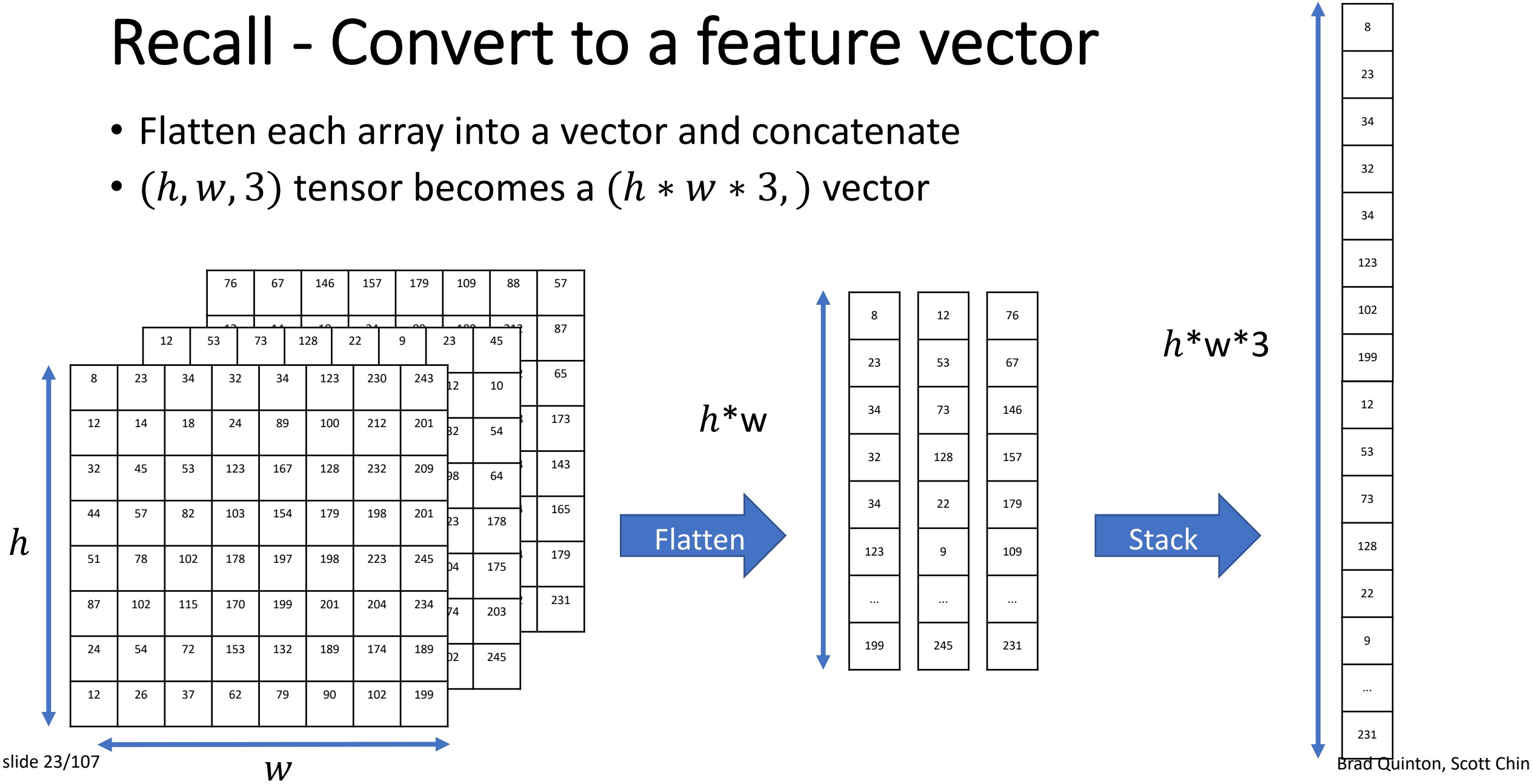


8	23	34	32	34	123	230	243
12	14	18	24	89	100	212	201
32	45	53	123	167	128	232	209
44	57	82	103	154	179	198	201
51	78	102	178	197	198	223	245
87	102	115	170	199	201	204	234
24	54	72	153	132	189	174	189
12	26	37	62	79	90	102	199

Brad Quinton, Scott Chin

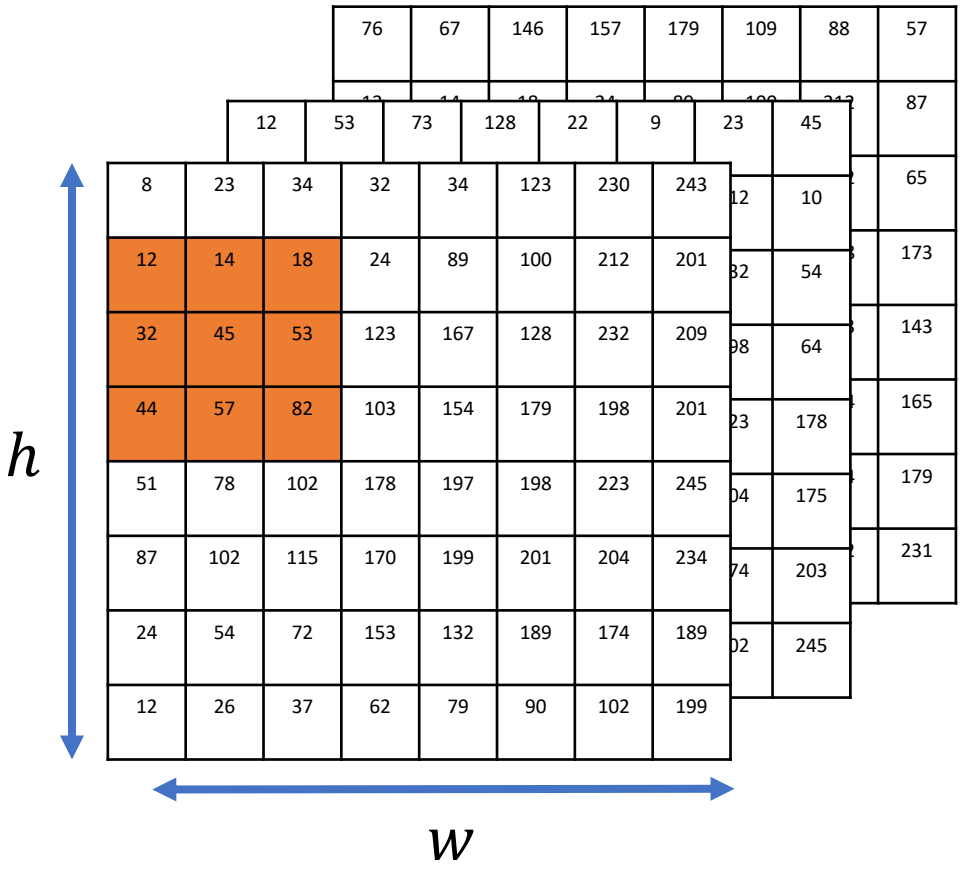
Recall - Convert to a feature vector

- Flatten each array into a vector and concatenate
- $(h, w, 3)$ tensor becomes a $(h * w * 3,)$ vector

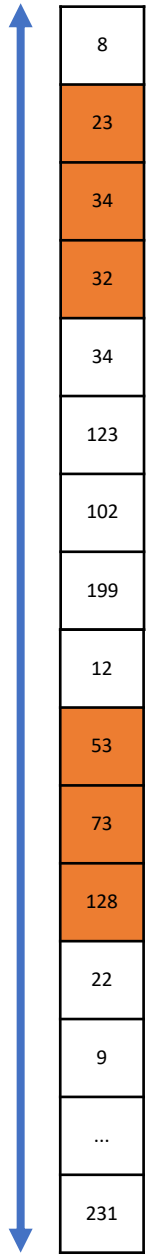


Brad Quinton, Scott Chin

Problem 1:
Throwing away Spatial Information



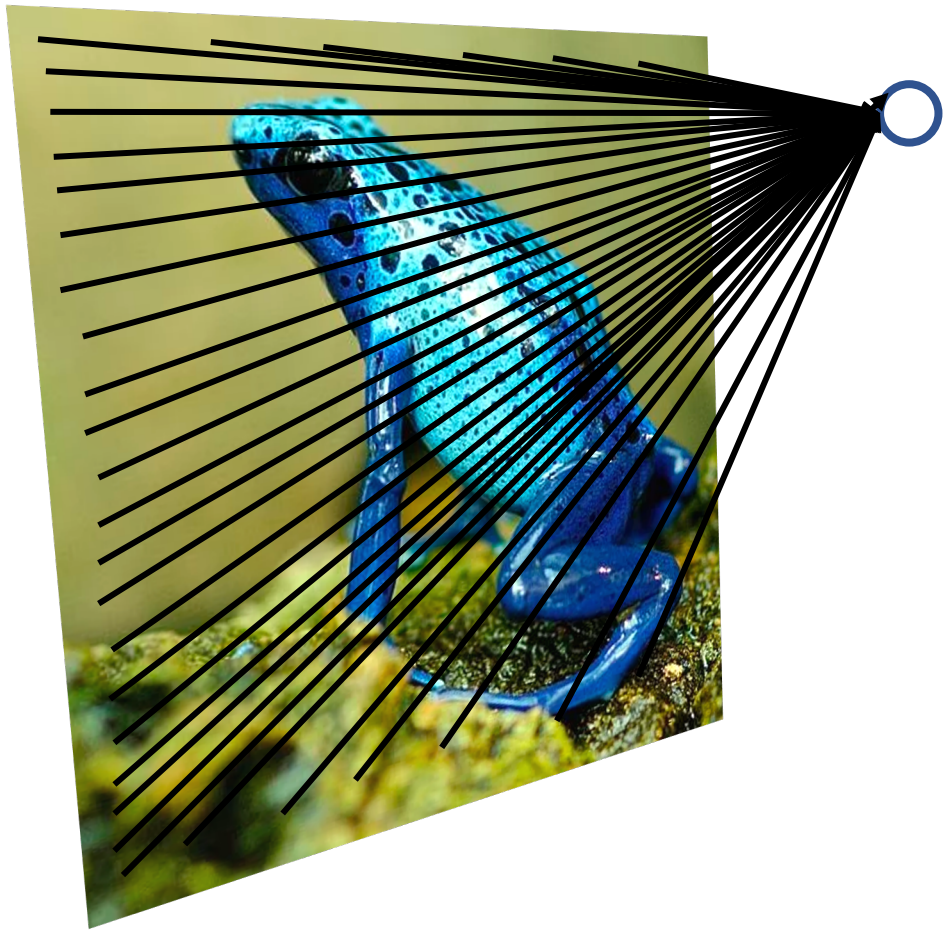
$h * w * 3$



Brad Quinton, Scott Chin

Ok Fine. Let’s look at it like a 2D structure

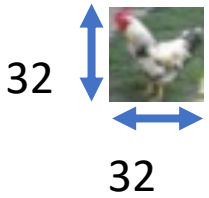
Number of Parameters

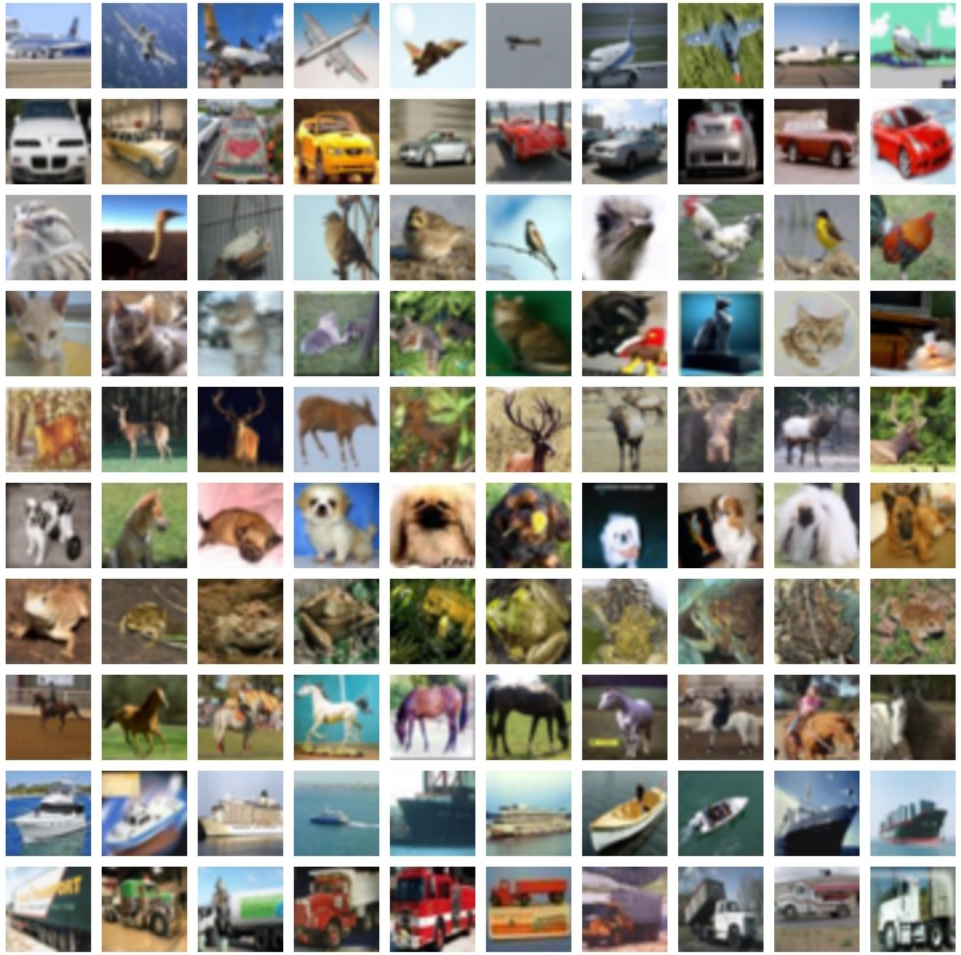


- Consider one fully connected neuron.
- Input is 32x32 color image
  → 32*32*3 = 3072 weights.
- Consider first layer has 1000 units
- $W^{[1]}$ is a $(1000, 3072)$ matrix
  → 3 million parameters to train
- $W^{[2]}$, $W^{[3]}$, …
- A bit big but manageable.

32x32 images are tiny…

- But this is what 32x32 images look like. So small.



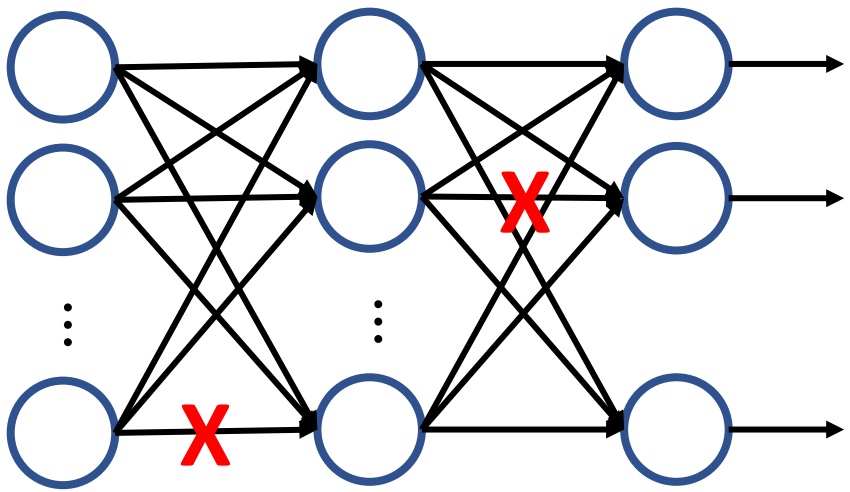


Number of Parameters

- iPhone 11 - 4032x3024x3
- That’s 36.5 million inputs
- First layer with 1000 units means 36.5 billion parameters

- Even for more reasonable size 200x200
- Still 120k features
  → 120million parameters

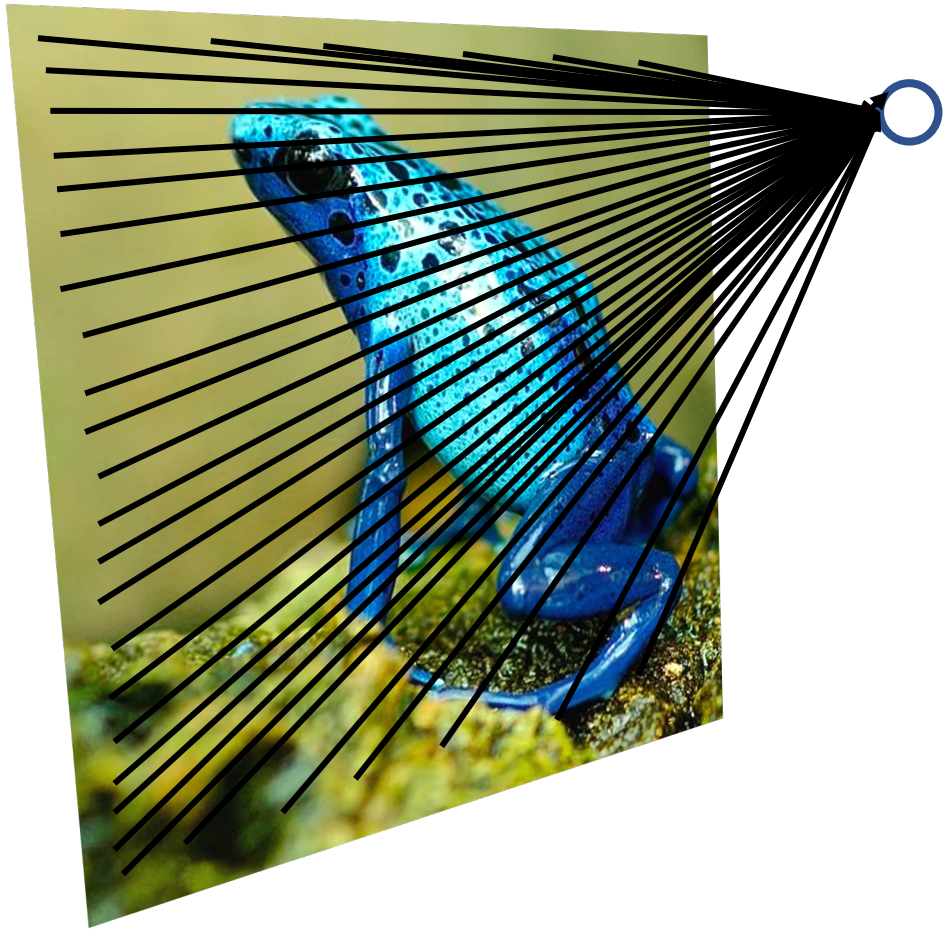
Problem 2: Too Many Parameters



- Want to use at least 200x200 px
- Fully connected network leads to too many parameters
- Can we do sparsely connected?
- But which connections to eliminate?
- How to efficiently train since we won't be able to effective vectorized computing?

Brad Quinton, Scott Chin

Problem 3: Pixels that form a visual feature are local



- Every unit is trying to make sense of the entire image.
- But spatial correlation is fairly local

Problem 3: Pixels forming a feature are local

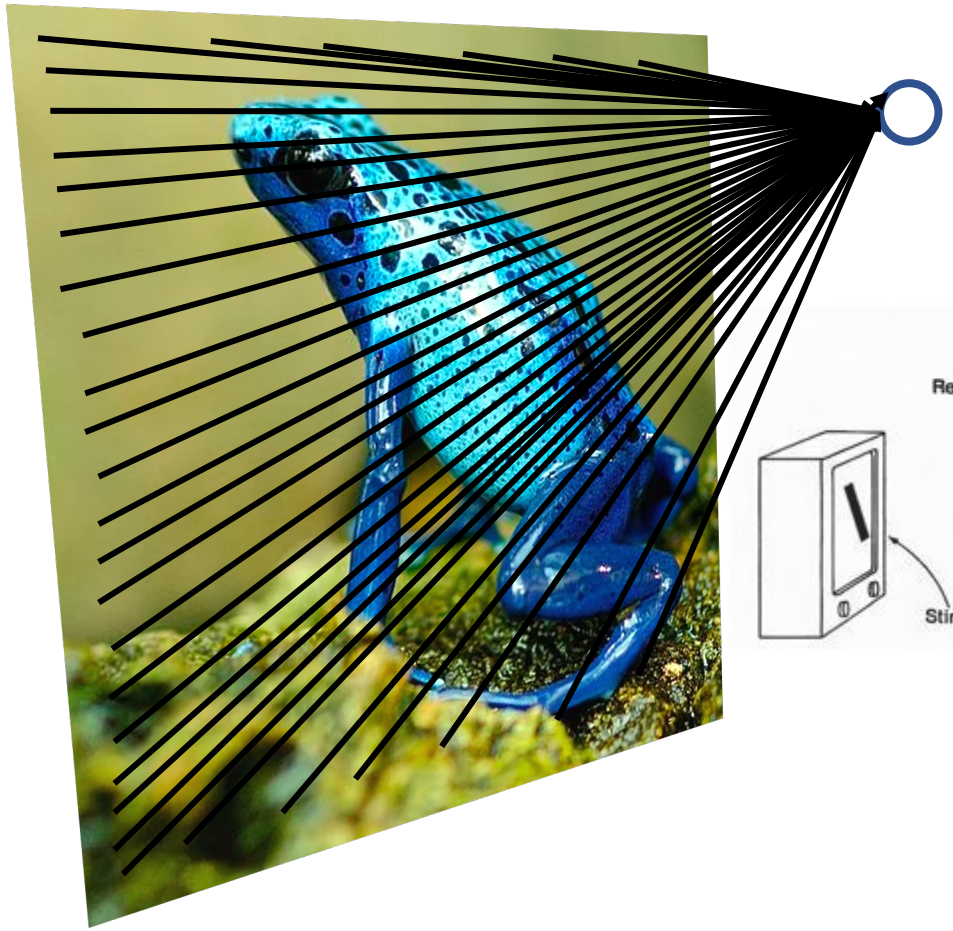
- Every unit is trying to make sense of the entire image.
- But spatial correlation is fairly local

For example:

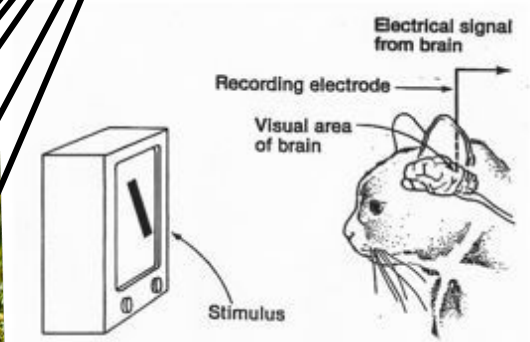
- Pixel of the eye has nothing to do with a pixel of the foreground. They are not really related.

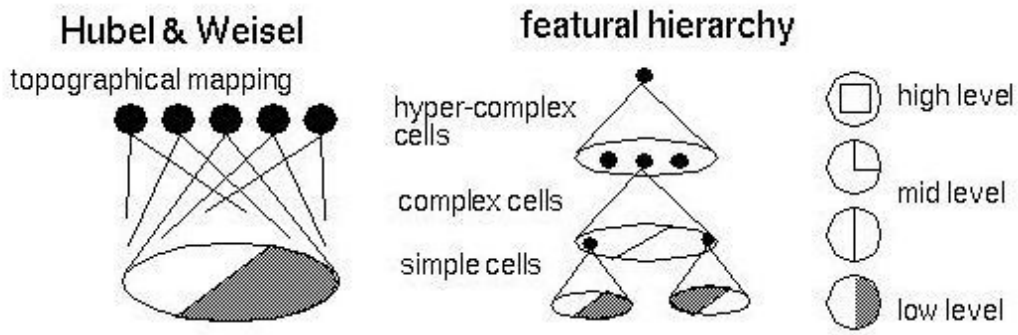
In theory, network can learn this. But you need a lot of data.

Problem 3: Pixels forming a feature are local



- Every unit is trying to make sense of the entire image.
- But spatial correlation is fairly local

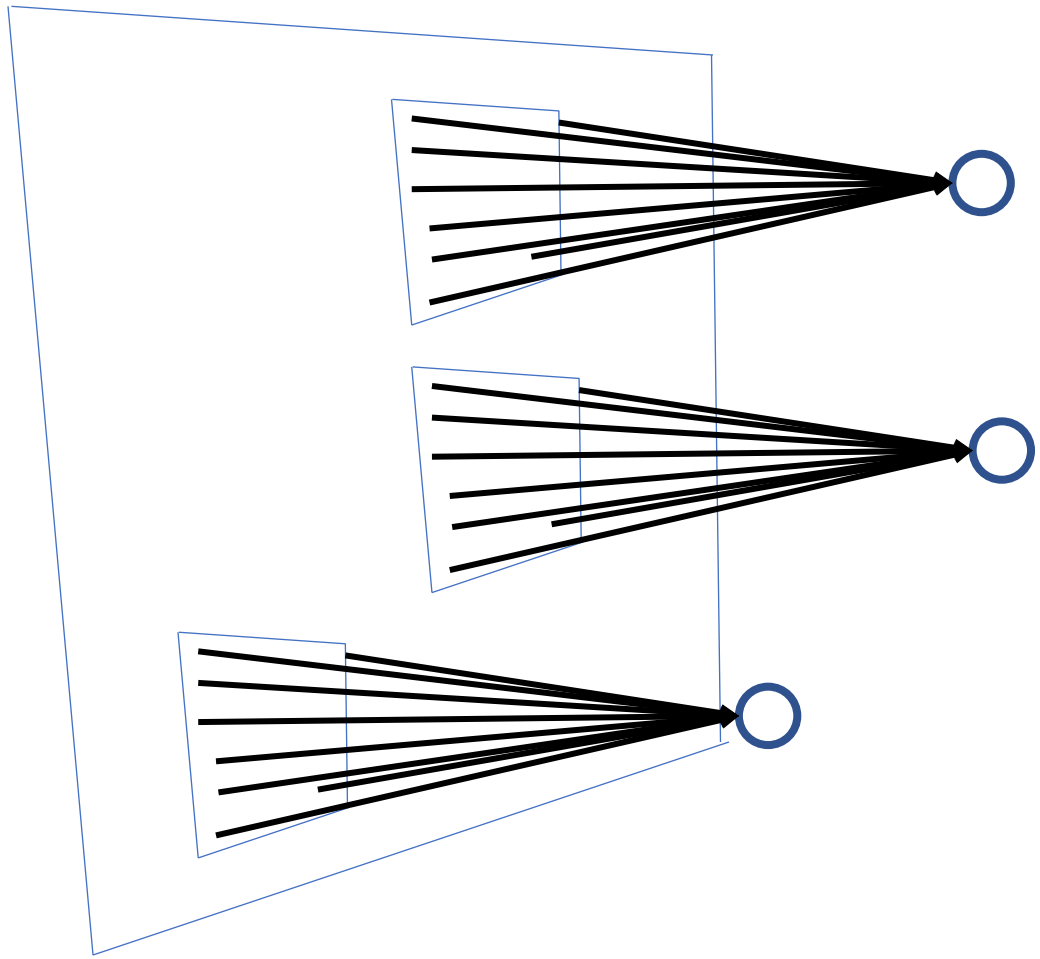




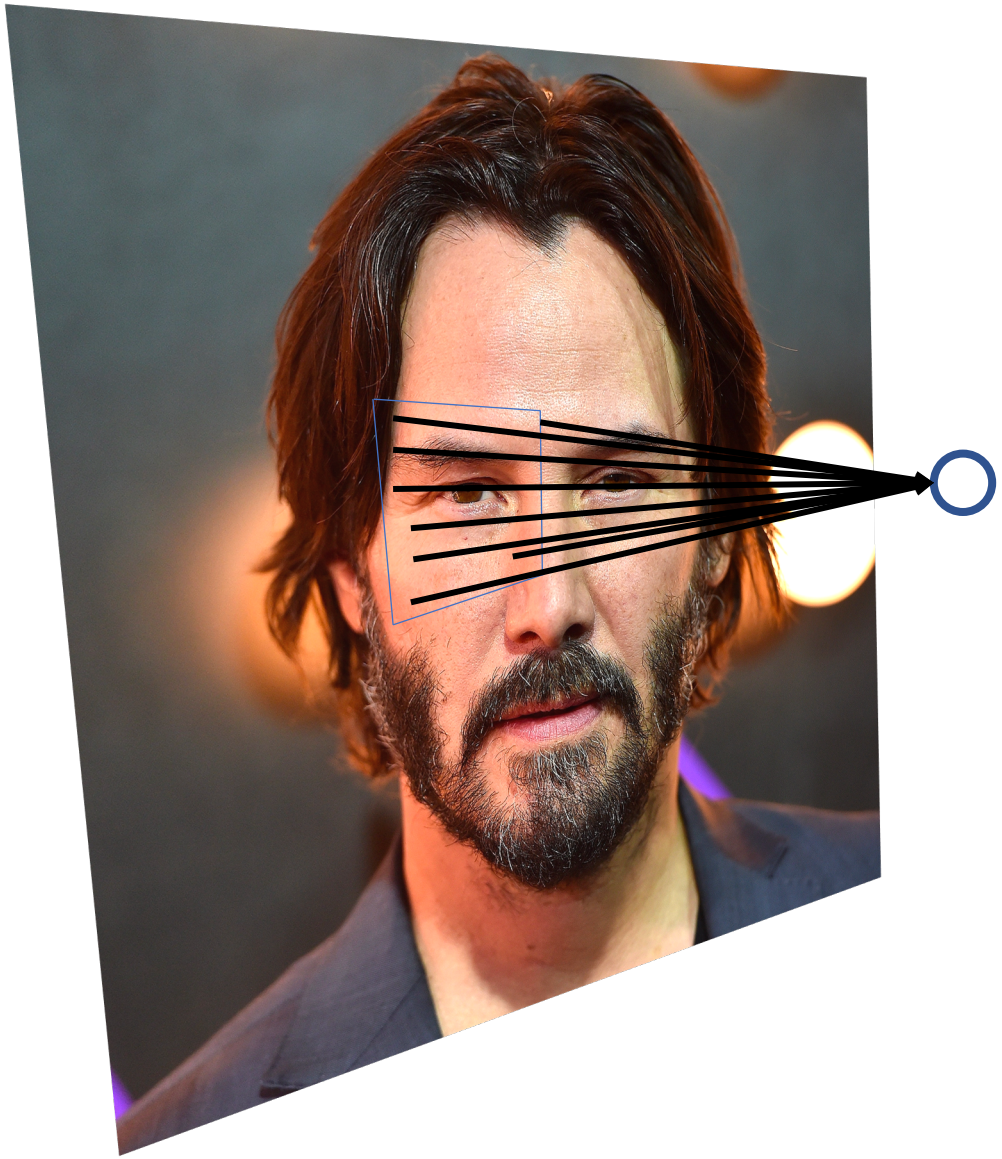
- Research in the 60's showed us that neuron's in the brain processes vision in a hierarchical manner.

"Receptive Fields of Single Neurones in the Cat's Striate Cortex", Hubel, Wiesel, 1959, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1363130/

Solution: Locally Connected



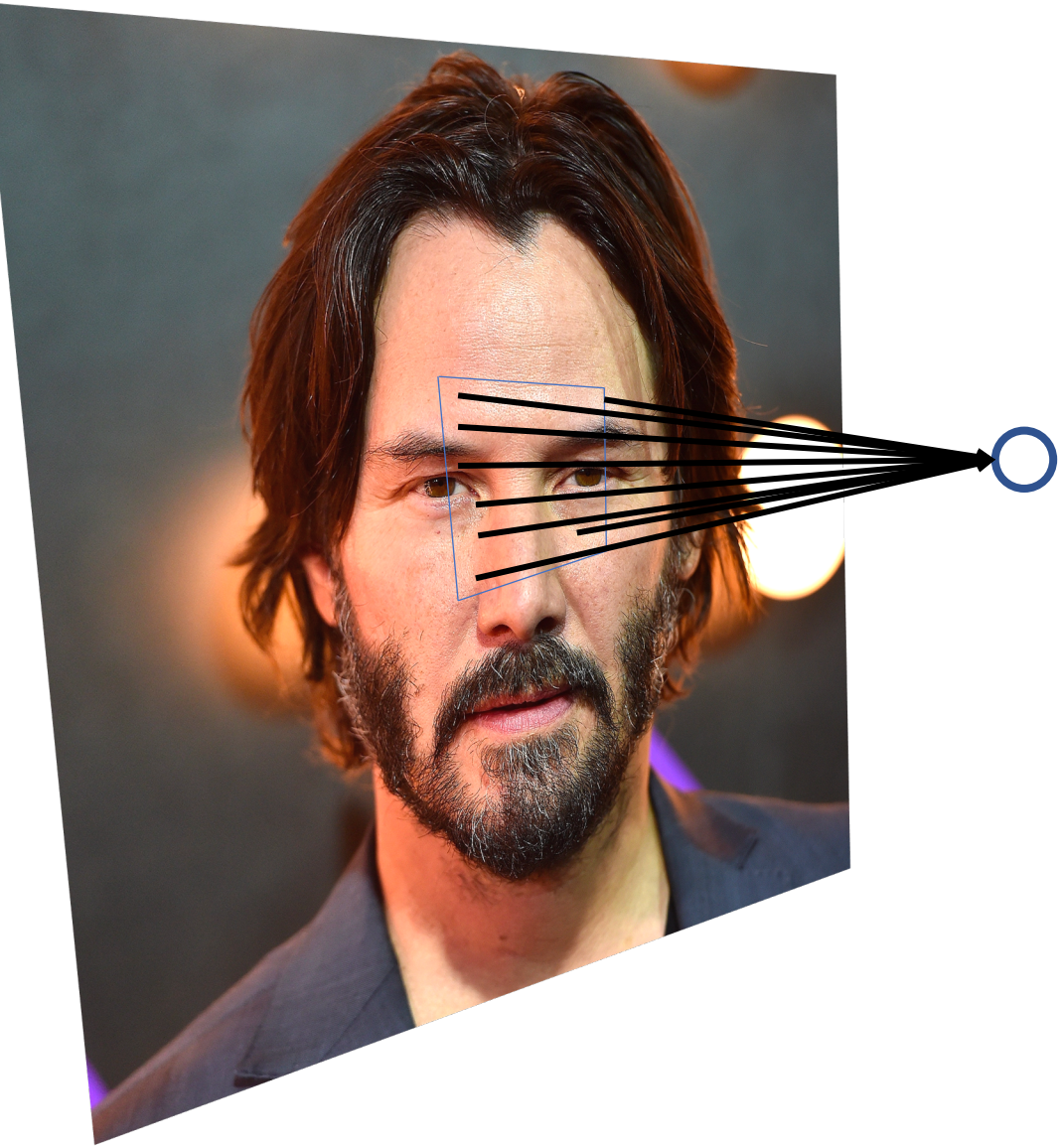
- Have each unit connect only to a smaller region of the image
  - For example, a 11px x 11px region
  - Sparsity!
  - Fewer parameters per unit!
  - Regions are contiguous so should be able to vectorize!
- Can work well on centered images such as face detection
- No tolerance to translation however



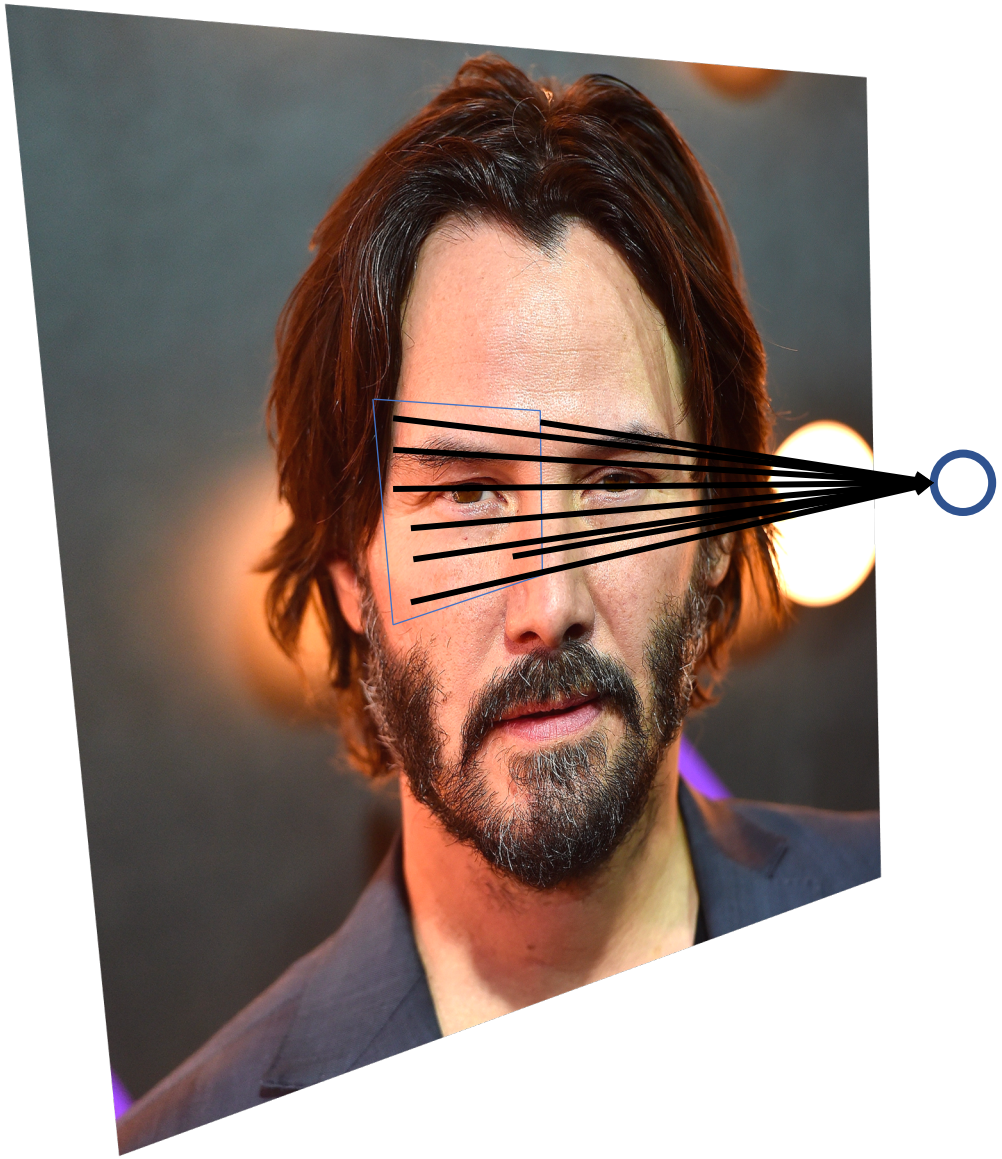
No tolerance to translation

- Say you train on roughly centered images of faces.
- This unit may ”learn” to activate strongly when the input looks like an eye
- But if you supply a non-centered face

No tolerance to translation



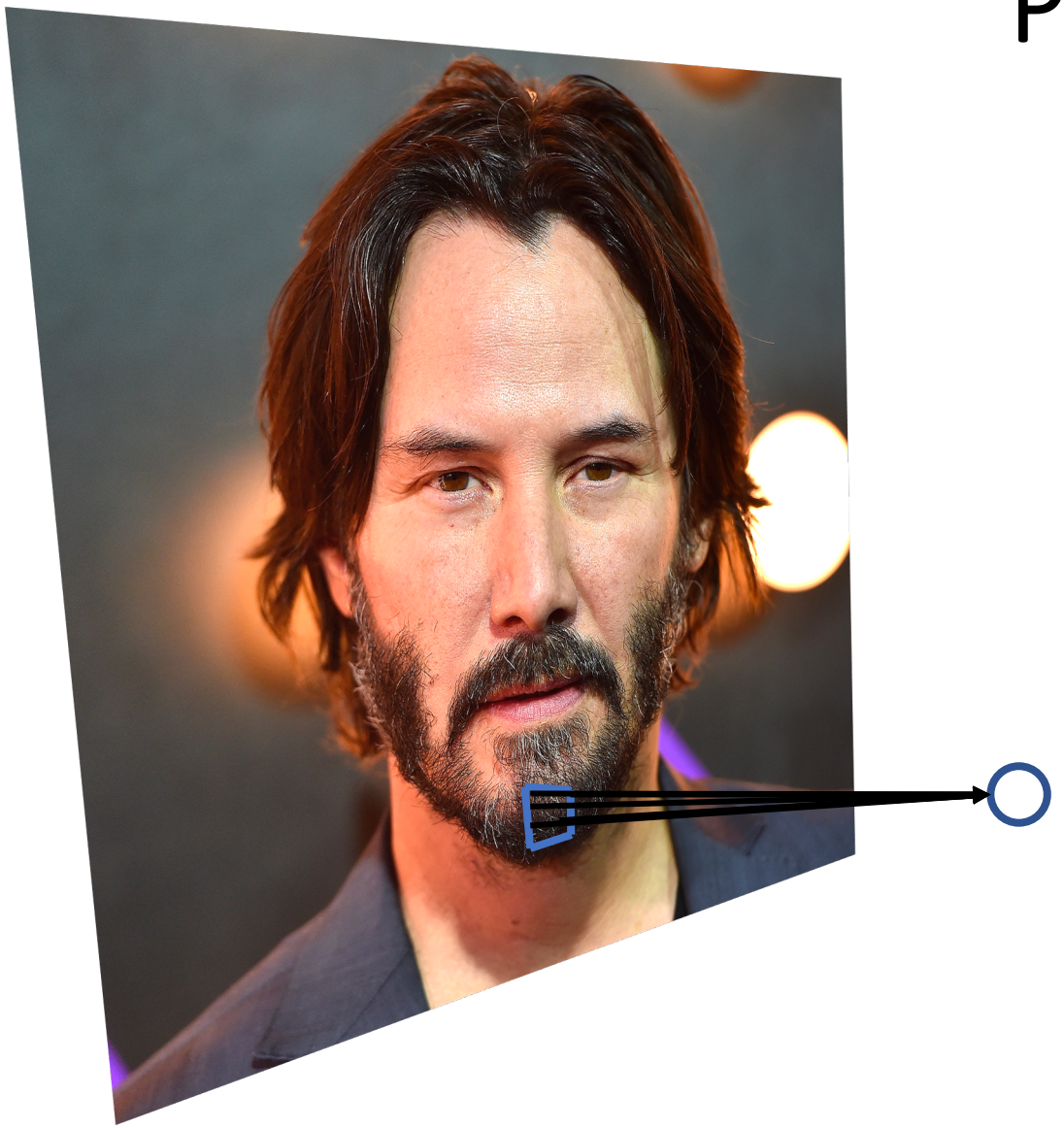
- Say you train on roughly centered images of faces.
- This unit may ”learn” to activate strongly when the input looks like an eye
- But if you supply a non-centered face, it won’t activate strongly
- And you wouldn’t expect the unit where the eye is now, to have learned to detect eyes since you trained on centered faces



Locally Connected

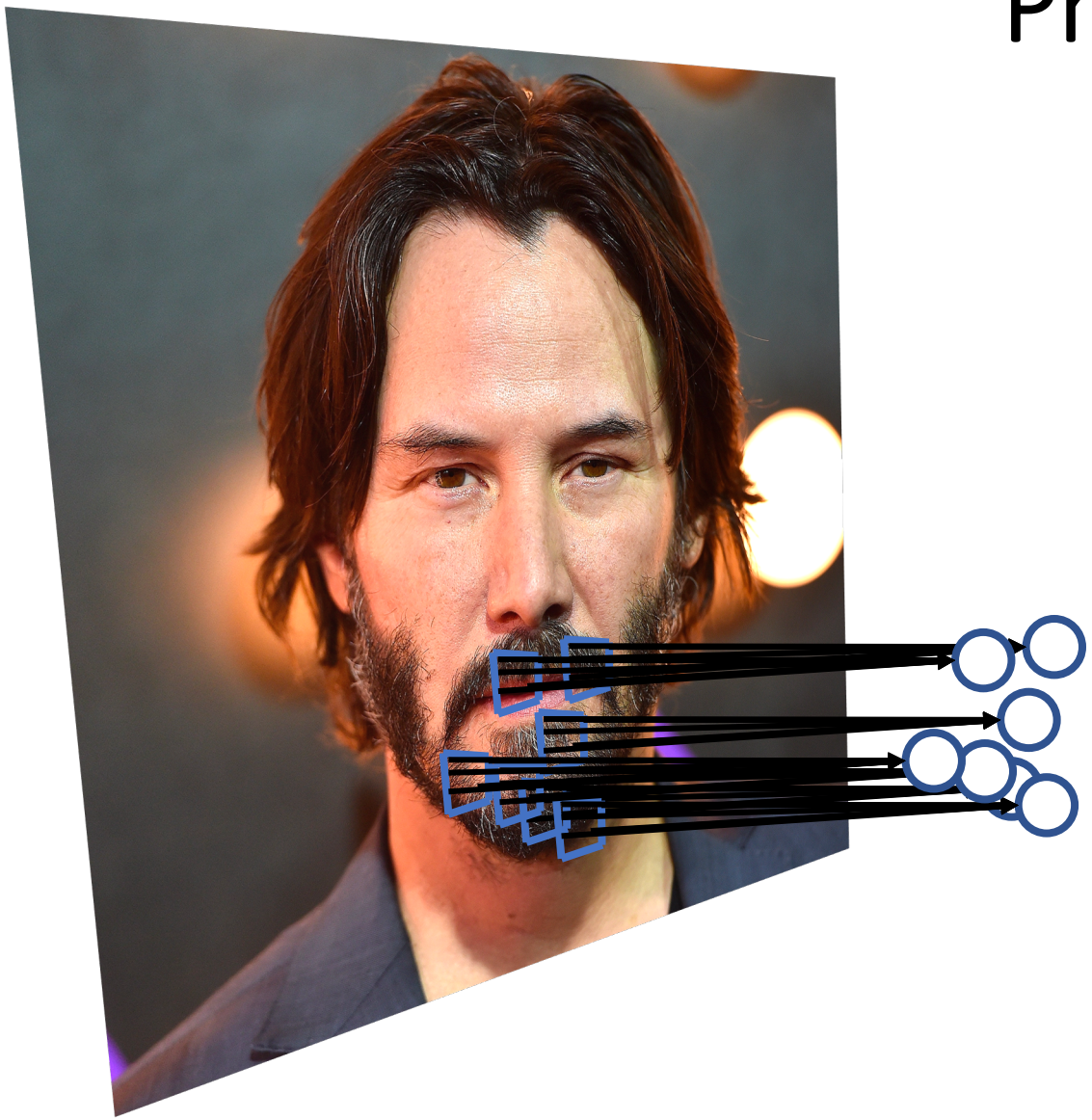
- Ok for some problems if you can expect centered data (i.e. visual features are in roughly the same spatial location):
  - Face detection
  - Skin cancer detection
  - Character recognition
- But what if that's not the case?
- Also, not taking advantage that image patterns at one location often repeat at other parts of the image.

Problem: Redundant Neurons



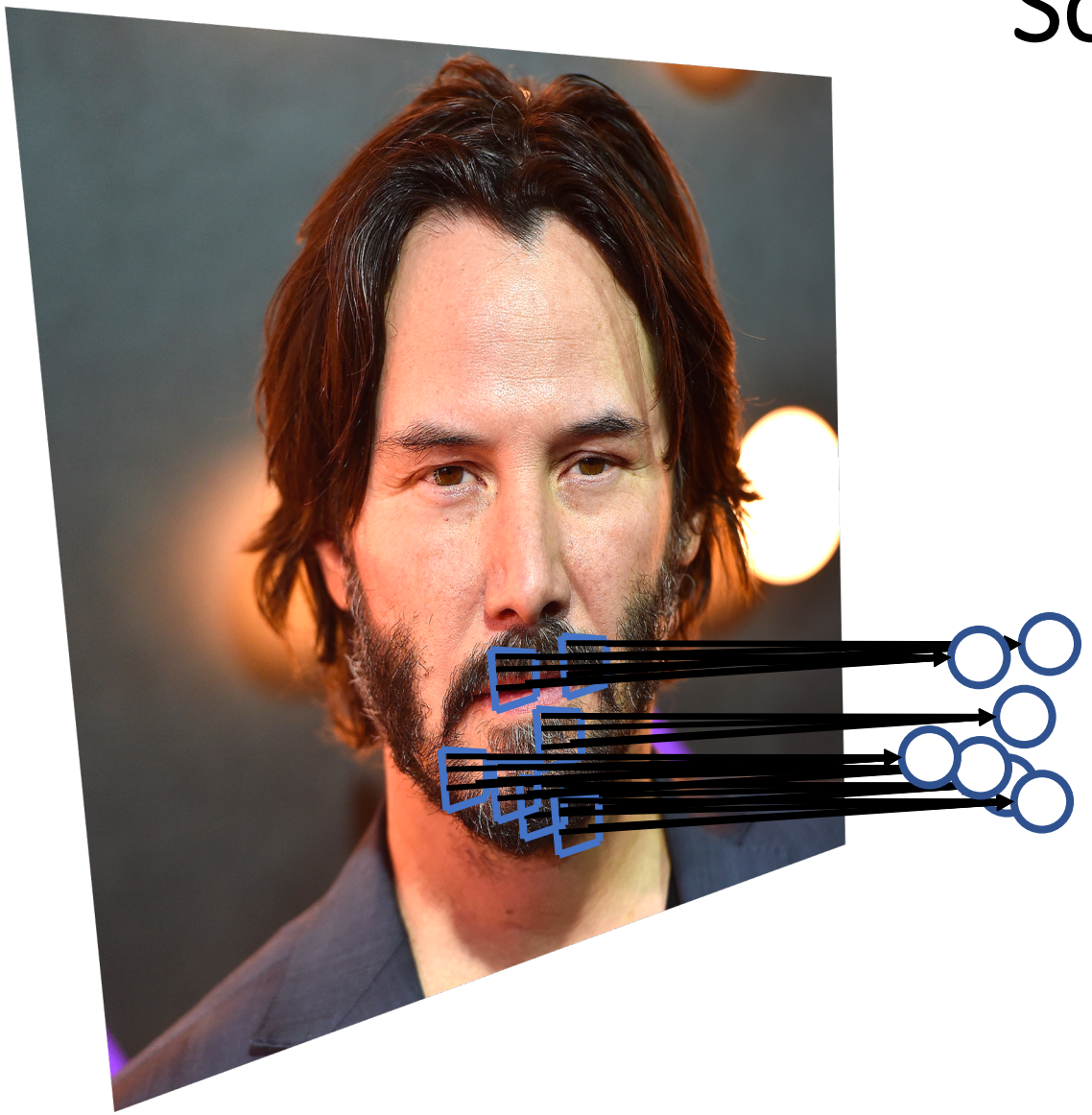
- For example, if a neuron learned to activate strongly to patterns that make up an 11x11 swatch of facial hair.

Problem: Redundant Neurons



- For example, if a neuron learned to activate strongly to patterns that make up an 11x11 swatch of facial hair.
- You'd expect this pattern to occur in many other places of the image.
- So now we have redundant neurons that are learning to do the same thing.

Solution: Shared parameters



- Ok let's have a bunch of the neurons at different locations share parameters
  - Further reduces number of parameters
- Simplify. Instead of multiple neurons sharing parameters, let's think of it as one neuron that "scans" looking for a specific feature → Filter aka Kernel
- Translation invariant (sort of)
- Use multiple filters. Each looks for a different feature.
- Convolutional Filters!

Convolutional Filters

Brad Quinton, Scott Chin

Motivation - Edge Detection

Given an image, how can we find the edges?

Example: Convolution Operation

Example 1

Brad Quinton, Scott Chin

Example 1: Convolution Operation

10	3	2	0	1	8
7	4	5	6	2	9
1	8	3	4	10	3
0	10	7	5	0	1
2	4	9	3	2	10
3	8	1	0	6	9

Input

- 6x6 pixel grayscale image

Brad Quinton, Scott Chin

Example 1: Convolution Operation

10	3	2	0	1	8
7	4	5	6	2	9
1	8	3	4	10	3
0	10	7	5	0	1
2	4	9	3	2	10
3	8	1	0	6	9

Input

- Say we have a 3x3 filter

1	0	-1
1	0	-1
1	0	-1

Brad Quinton, Scott Chin

Example 1: Convolution Operation

10	3	2	0	1	8
7	4	5	6	2	9
1	8	3	4	10	3
0	10	7	5	0	1
2	4	9	3	2	10
3	8	1	0	6	9

Input

conv

1	0	-1
1	0	-1
1	0	-1

→

8	5	-3	-10
-7	7	3	2
-16	10	7	-2
-12	14	9	-12

Element-wise multiply and sum

Brad Quinton, Scott Chin

Example 1: Convolution Operation

10	3	2	0	1	8
7	4	5	6	2	9
1	8	3	4	10	3
0	10	7	5	0	1
2	4	9	3	2	10
3	8	1	0	6	9

Input

conv

1	0	-1
1	0	-1
1	0	-1

→

8	5	-3	-10
-7	7	3	2
-16	10	7	-2
-12	14	9	-12

Element-wise multiply and sum

Brad Quinton, Scott Chin

Example 1: Convolution Operation

10	3	2	0	1	8
7	4	5	6	2	9
1	8	3	4	10	3
0	10	7	5	0	1
2	4	9	3	2	10
3	8	1	0	6	9

Input

conv

1	0	-1
1	0	-1
1	0	-1

→

8	5	-3	-10
-7	7	3	2
-16	10	7	-2
-12	14	9	-12

Element-wise multiply and sum

Brad Quinton, Scott Chin

Example 1: Convolution Operation

10	3	2	0	1	8
7	4	5	6	2	9
1	8	3	4	10	3
0	10	7	5	0	1
2	4	9	3	2	10
3	8	1	0	6	9

Input

conv

1	0	-1
1	0	-1
1	0	-1

→

8	5	-3	-10
-7	7	3	2
-16	10	7	-2
-12	14	9	-12

Element-wise multiply and sum

Brad Quinton, Scott Chin

Example 1: Convolution Operation

10	3	2	0	1	8
7	4	5	6	2	9
1	8	3	4	10	3
0	10	7	5	0	1
2	4	9	3	2	10
3	8	1	0	6	9

Input

conv

1	0	-1
1	0	-1
1	0	-1

→

8	5	-3	-10
-7	7	3	2
-16	10	7	-2
-12	14	9	-12

Element-wise multiply and sum

Brad Quinton, Scott Chin

Example 1: Convolution Operation

10	3	2	0	1	8
7	4	5	6	2	9
1	8	3	4	10	3
0	10	7	5	0	1
2	4	9	3	2	10
3	8	1	0	6	9

Input

conv

1	0	-1
1	0	-1
1	0	-1

→

8	5	-3	-10
-7	7	3	2
-16	10	7	-2
-12	14	9	-12

Element-wise multiply and sum

Brad Quinton, Scott Chin

Example 1: Convolution Operation

10	3	2	0	1	8
7	4	5	6	2	9
1	8	3	4	10	3
0	10	7	5	0	1
2	4	9	3	2	10
3	8	1	0	6	9

Input

conv

1	0	-1
1	0	-1
1	0	-1

→

8	5	-3	-10
-7	7	3	2
-16	10	7	-2
-12	14	9	-12

Element-wise multiply and sum

Brad Quinton, Scott Chin

Example 1: Convolution Operation

10	3	2	0	1	8
7	4	5	6	2	9
1	8	3	4	10	3
0	10	7	5	0	1
2	4	9	3	2	10
3	8	1	0	6	9

Input

conv

1	0	-1
1	0	-1
1	0	-1

→

8	5	-3	-10
-7	7	3	2
-16	10	7	-2
-12	14	9	-12

Element-wise multiply and sum

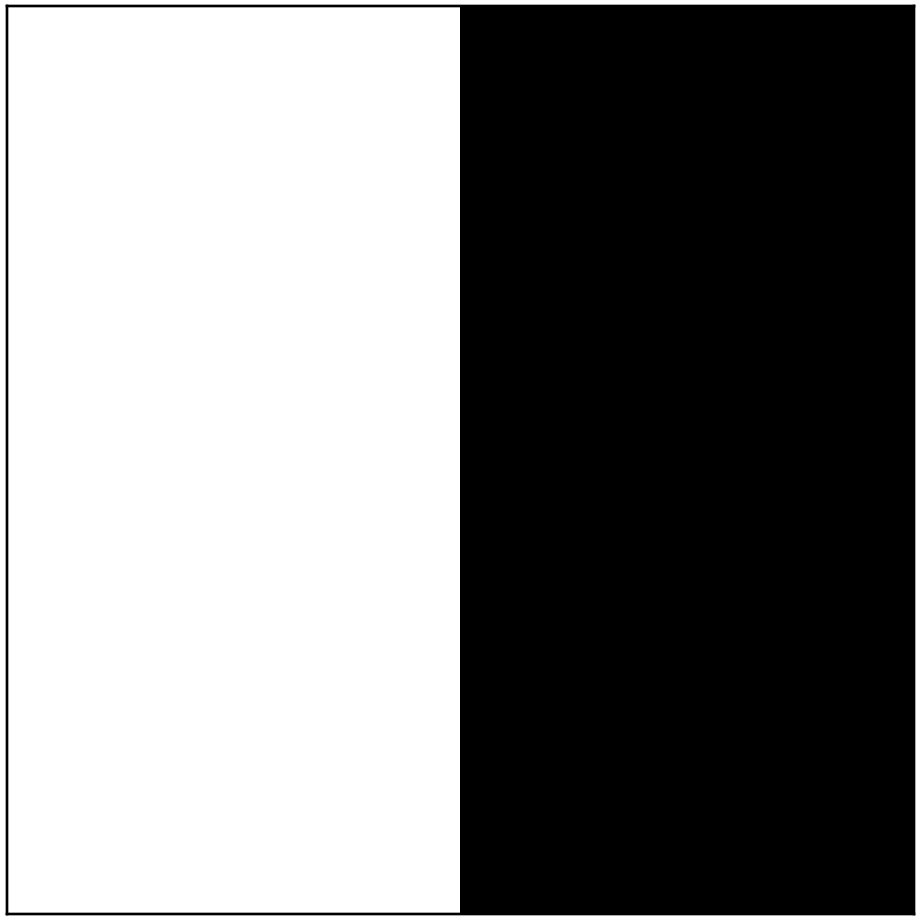
Brad Quinton, Scott Chin

Example: Vertical Edge Detection

Example 2

Brad Quinton, Scott Chin

Example 2: Vertical Edge Detection



conv

1	0	-1
1	0	-1
1	0	-1

→

Brad Quinton, Scott Chin

Example 2: Vertical Edge Detection

255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0

conv

1	0	-1
1	0	-1
1	0	-1

→

Brad Quinton, Scott Chin

Example 2: Vertical Edge Detection

255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0

Output is intensity with which vertical edge occurs at the corresponding input location

conv

1	0	-1
1	0	-1
1	0	-1

→

0	765	765	0
0	765	765	0
0	765	765	0
0	765	765	0

Brad Quinton, Scott Chin

Example 3: Weak Vertical Edge

1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0

Output is intensity is now lower

conv

1	0	-1
1	0	-1
1	0	-1

→

0	3	3	0
0	3	3	0
0	3	3	0
0	3	3	0

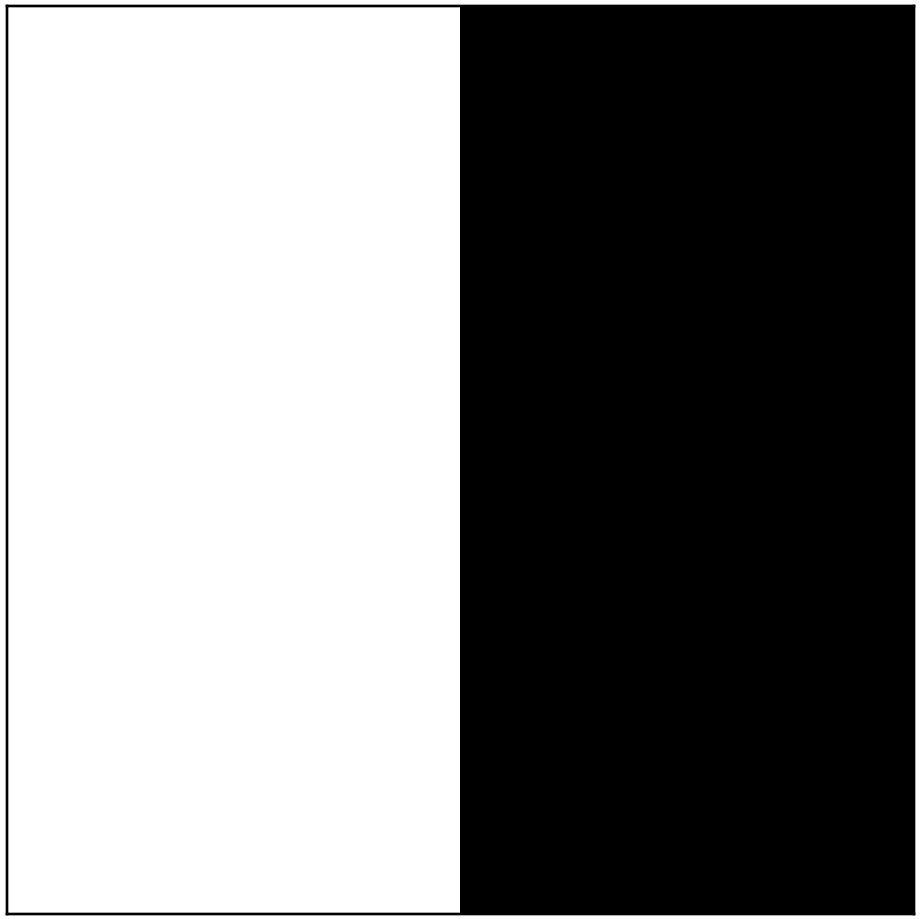
Brad Quinton, Scott Chin

Example: Vertical Edge Detection

Example 2

Brad Quinton, Scott Chin

Example 2: Vertical Edge Detection



conv

1	0	-1
1	0	-1
1	0	-1

→

Brad Quinton, Scott Chin

Example 2: Vertical Edge Detection

255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0

conv

1	0	-1
1	0	-1
1	0	-1

→

Brad Quinton, Scott Chin

Example 2: Vertical Edge Detection

255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0

Output is intensity with which vertical edge occurs at the corresponding input location

conv

1	0	-1
1	0	-1
1	0	-1

→

0	765	765	0
0	765	765	0
0	765	765	0
0	765	765	0

Brad Quinton, Scott Chin

Example 3: Weak Vertical Edge

1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0

Output is intensity is now lower

conv

1	0	-1
1	0	-1
1	0	-1

→

0	3	3	0
0	3	3	0
0	3	3	0
0	3	3	0

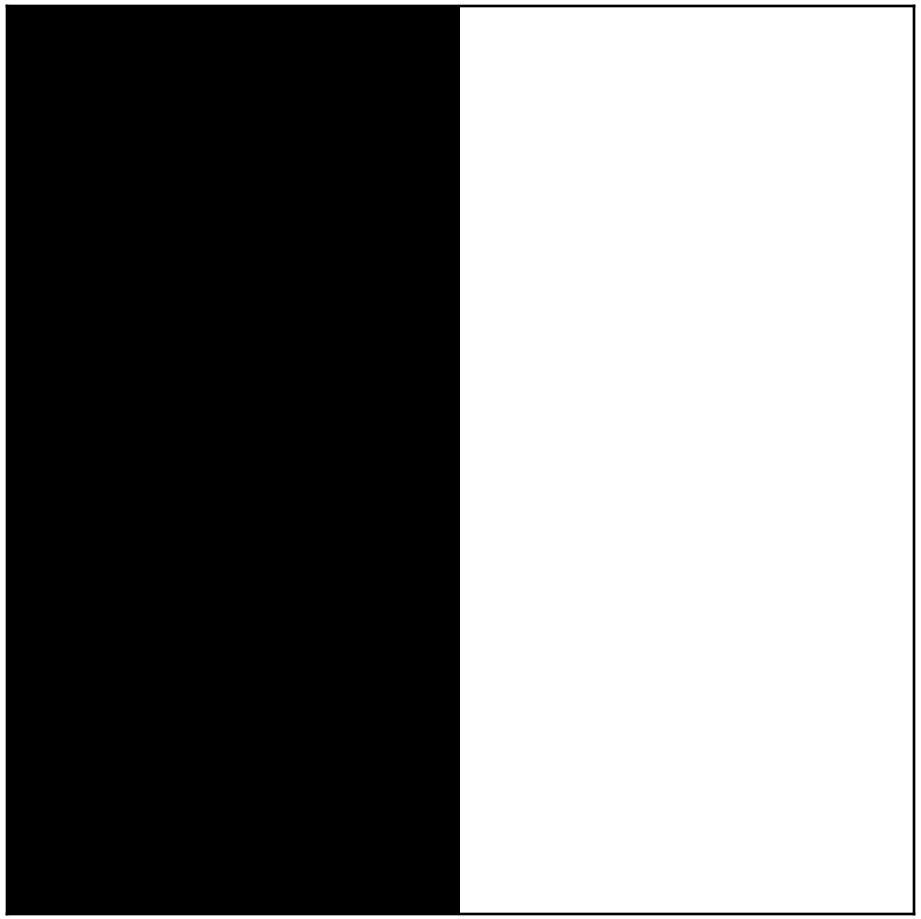
Brad Quinton, Scott Chin

Example:
Dark to Light Vertical Edge

Example 4

Brad Quinton, Scott Chin

Example 4: Dark to light



conv

1	0	-1
1	0	-1
1	0	-1

→

Brad Quinton, Scott Chin

Example 4: Dark to light

0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255

Sign is different. So this filter is explicitly looking for a light-to-dark edge

conv

1	0	-1
1	0	-1
1	0	-1

→

0	-765	-765	0
0	-765	-765	0
0	-765	-765	0
0	-765	-765	0

Brad Quinton, Scott Chin

Example 4: Dark to light

0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255

Change signs on the filter to explicitly look for dark to light edges

conv

-1	0	1
-1	0	1
-1	0	1

→

0	765	765	0
0	765	765	0
0	765	765	0
0	765	765	0

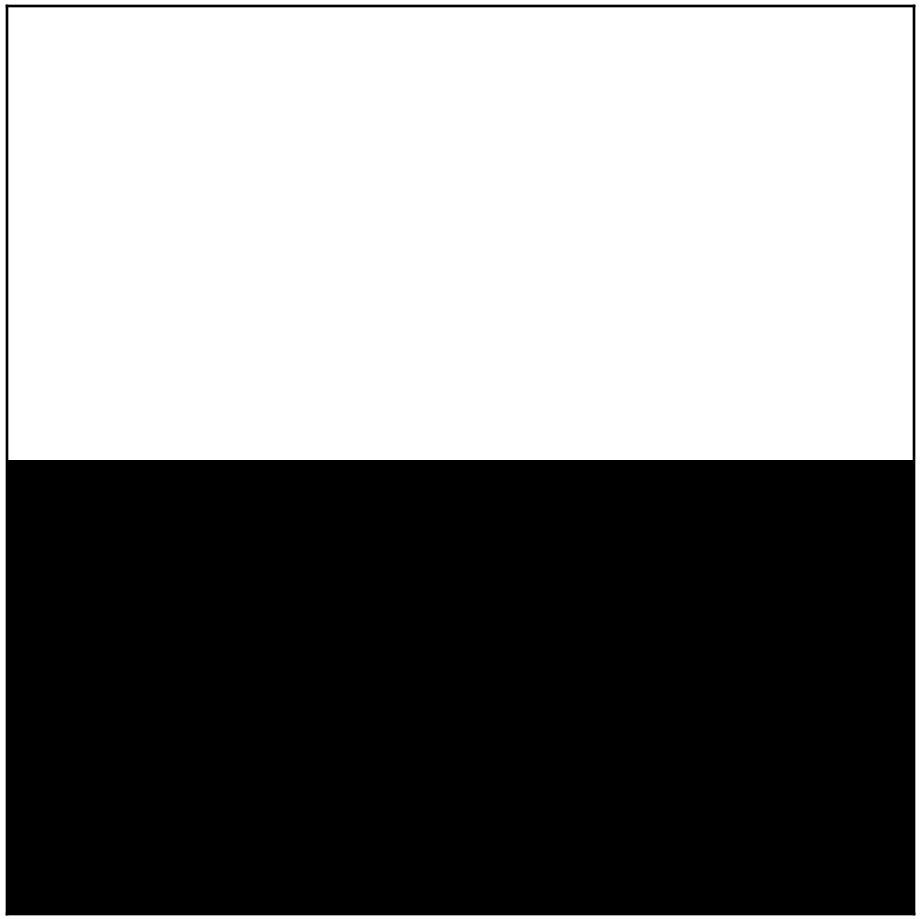
Brad Quinton, Scott Chin

Example:
Absence of Vertical Edge

Example 5

Brad Quinton, Scott Chin

Example 5: Vertical Edge Detection (Absence)



conv

1	0	-1
1	0	-1
1	0	-1

→

Brad Quinton, Scott Chin

Example 5: Vertical Edge Detection (Absence)

255	255	255	255	255	255
255	255	255	255	255	255
255	255	255	255	255	255
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

conv

1	0	-1
1	0	-1
1	0	-1

→

0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Brad Quinton, Scott Chin

Horizontal Edge Filter

1	0	-1
1	0	-1
1	0	-1

Vertical Edge Filter

?	?	?
?	?	?
?	?	?

Horizontal Edge Filter

Horizontal Edge Filter

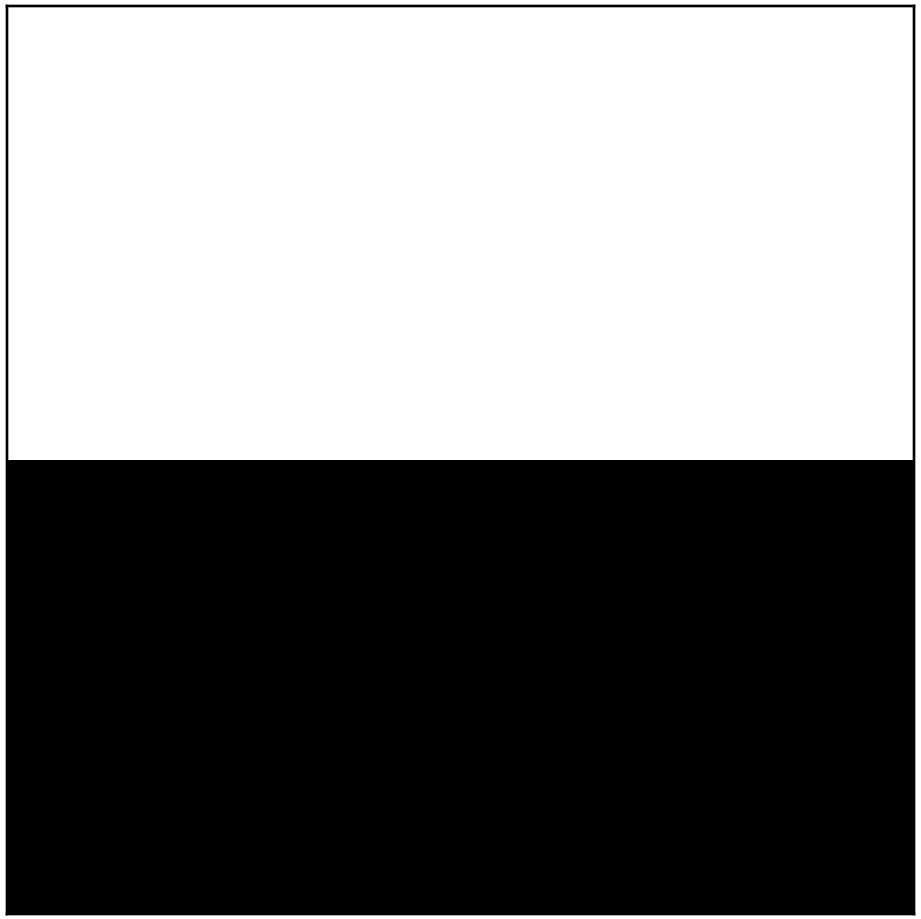
1	0	-1
1	0	-1
1	0	-1

Vertical Edge Filter

1	1	1
0	0	0
-1	-1	-1

Horizontal Edge Filter

Example 6: Horizontal Edge Detection



conv

1	1	1
0	0	0
-1	-1	-1

→

Brad Quinton, Scott Chin

Example 6: Horizontal Edge Detection

255	255	255	255	255	255
255	255	255	255	255	255
255	255	255	255	255	255
0	0	0	0	0	0
0	0	0	0	0	0
0	0	0	0	0	0

conv

1	1	1
0	0	0
-1	-1	-1

→

0	0	0	0
765	765	765	765
765	765	765	765
0	0	0	0

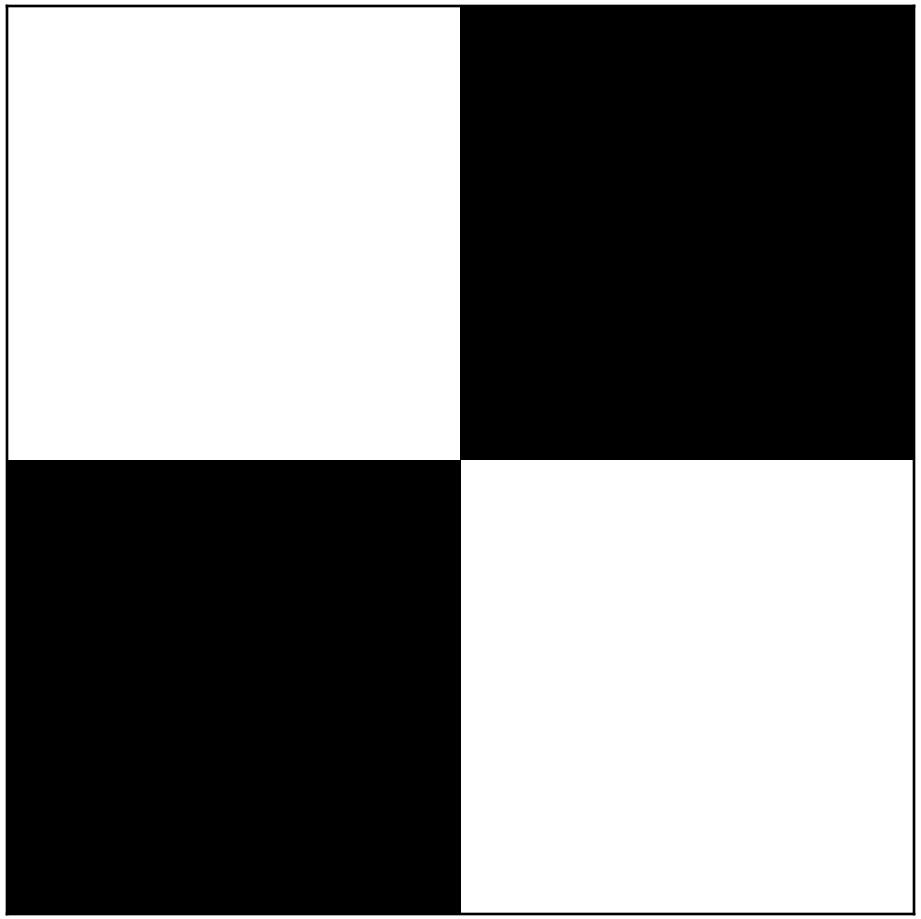
Brad Quinton, Scott Chin

Example:
Horizontal and Vertical Edge

Example 7

Brad Quinton, Scott Chin

Example 7: Horizontal and Vertical Edges



*

1	1	1
0	0	0
-1	-1	-1

→

Brad Quinton, Scott Chin

Example 7: Horizontal and Vertical Edges

255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255

*

1	1	1
0	0	0
-1	-1	-1

→

0	0	0	0
765	255	-255	765
765	255	-255	765
0	0	0	0

Brad Quinton, Scott Chin

Example 7: Horizontal and Vertical Edges

255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255

No edges here

conv

1	1	1
0	0	0
-1	-1	-1

→

0	0	0	0
765	255	-255	765
765	255	-255	765
0	0	0	0

Brad Quinton, Scott Chin

Example 7: Horizontal and Vertical Edges

255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255

Strong Horizontal Edge

conv

1	1	1
0	0	0
-1	-1	-1

→

0	0	0	0
765	255	-255	765
765	255	-255	765
0	0	0	0

Brad Quinton, Scott Chin

Example 7: Horizontal and Vertical Edges

255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255

conv

1	1	1
0	0	0
-1	-1	-1

→

0	0	0	0
765	255	-255	765
765	255	-255	765
0	0	0	0

No Horizontal Edge (but vertical edge here)

Brad Quinton, Scott Chin

Example 7: Horizontal and Vertical Edges

255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
0	0	0	255	255	255
0	0	0	255	255	255
0	0	0	255	255	255

partial horizontal edge

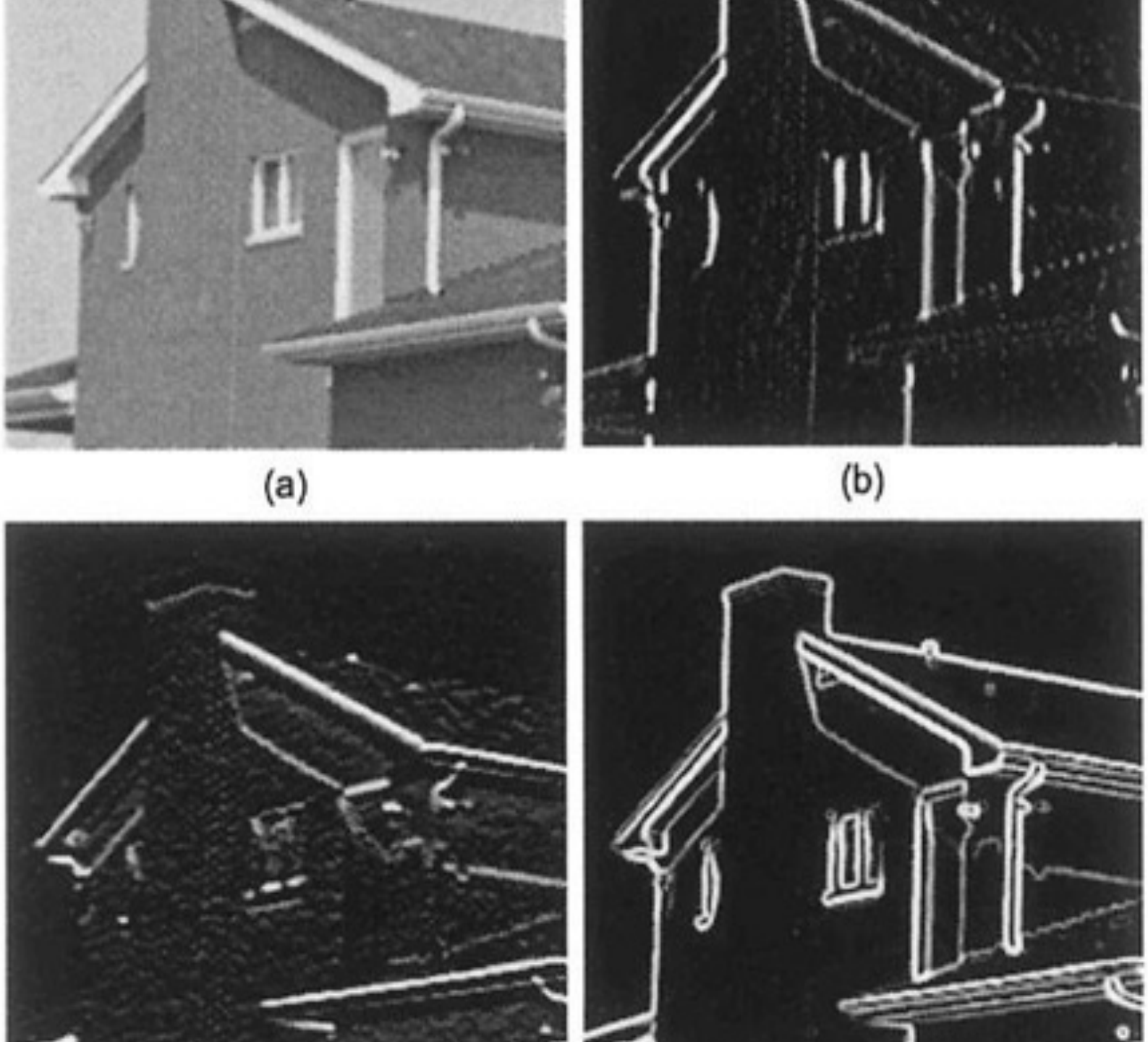
conv

1	1	1
0	0	0
-1	-1	-1

→

0	0	0	0
765	255	-255	765
765	255	-255	765
0	0	0	0

Brad Quinton, Scott Chin


(a)
(b)

Interpretation of Convolution Output

- Output of convolution is a Feature Map
- Describes the intensity and location where a feature is present in the input image

255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0
255	255	255	0	0	0

conv

1	0	-1
1	0	-1
1	0	-1

=

0	765	765	0
0	765	765	0
0	765	765	0
0	765	765	0

Brad Quinton, Scott Chin

How to Choose Filter Values?

1	0	-1
1	0	-1
1	0	-1

Vertical Edge Filter

How to Choose Filter Values?

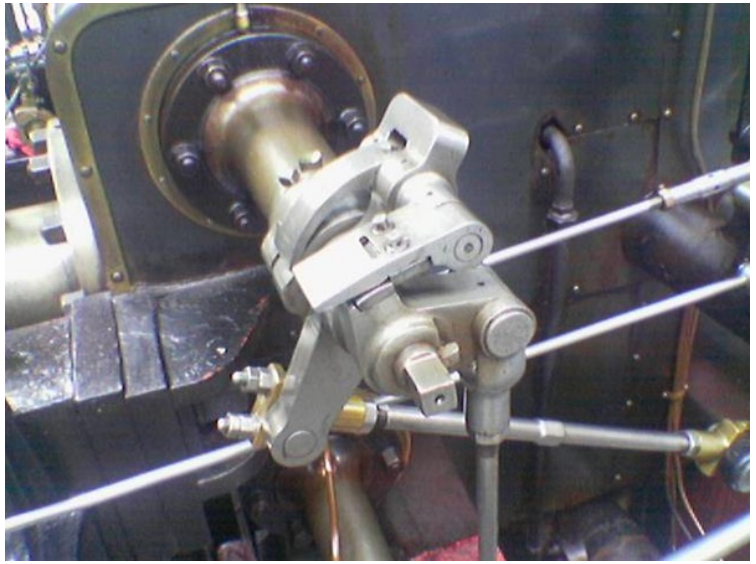
- Much research in traditional computer vision on best values to use

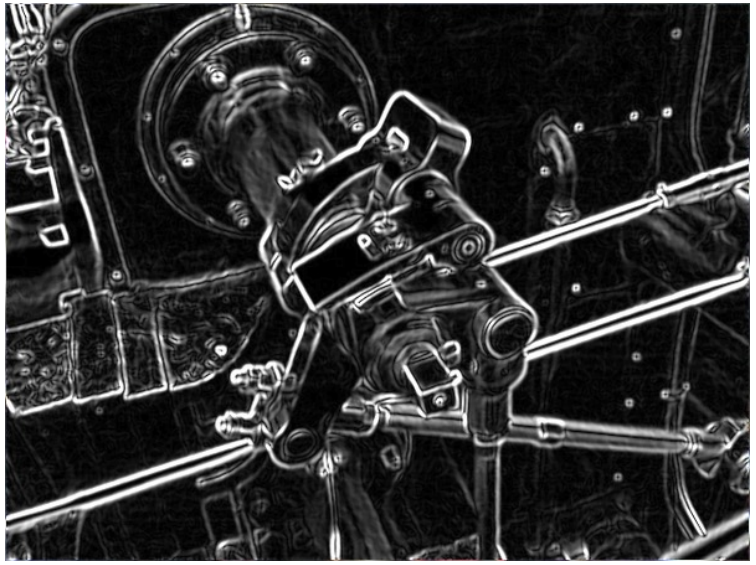
1	0	-1
1	0	-1
1	0	-1

Vertical Edge Filter

1	0	-1
2	0	-2
1	0	-1

Sobel Filter





How to Choose Filter Values?

- Much research in traditional computer vision on best values to use

1	0	-1
1	0	-1
1	0	-1

Vertical Edge Filter

1	0	-1
2	0	-2
1	0	-1

Sobel Filter

3	0	-3
10	0	-10
3	0	-3

Scharr Filter

Let the deep learning model decide!

- Treat the filter values as learnable parameters
- Supply data and let the model learn the best values for the data

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

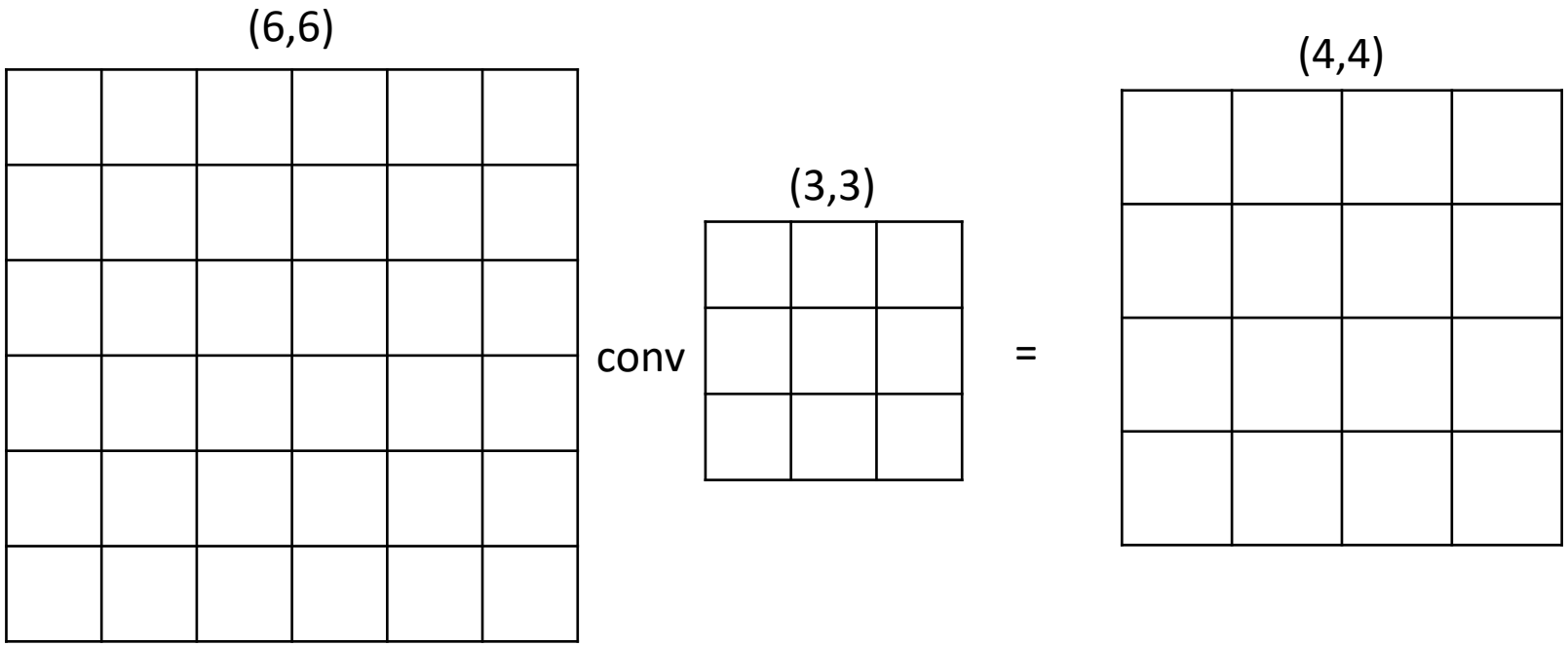
Brad Quinton, Scott Chin

Treat Filter Values as Learned Parameters

- You can do backpropagation through the convolution operation
- Convolution is just a simple linear operation (multiply accumulate)
- Use techniques we already know to train on a set of images and "learn" the set of filter values to best minimize some loss.
- This could result in the same values such as Sobel or Scharr filters, but most likely some other values that best work for the training data.
- The concept of automatically learning the filter values was a massive shift from traditional computer vision approach of developing hand-crafted filters

Generalizing the shapes of filters and outputs

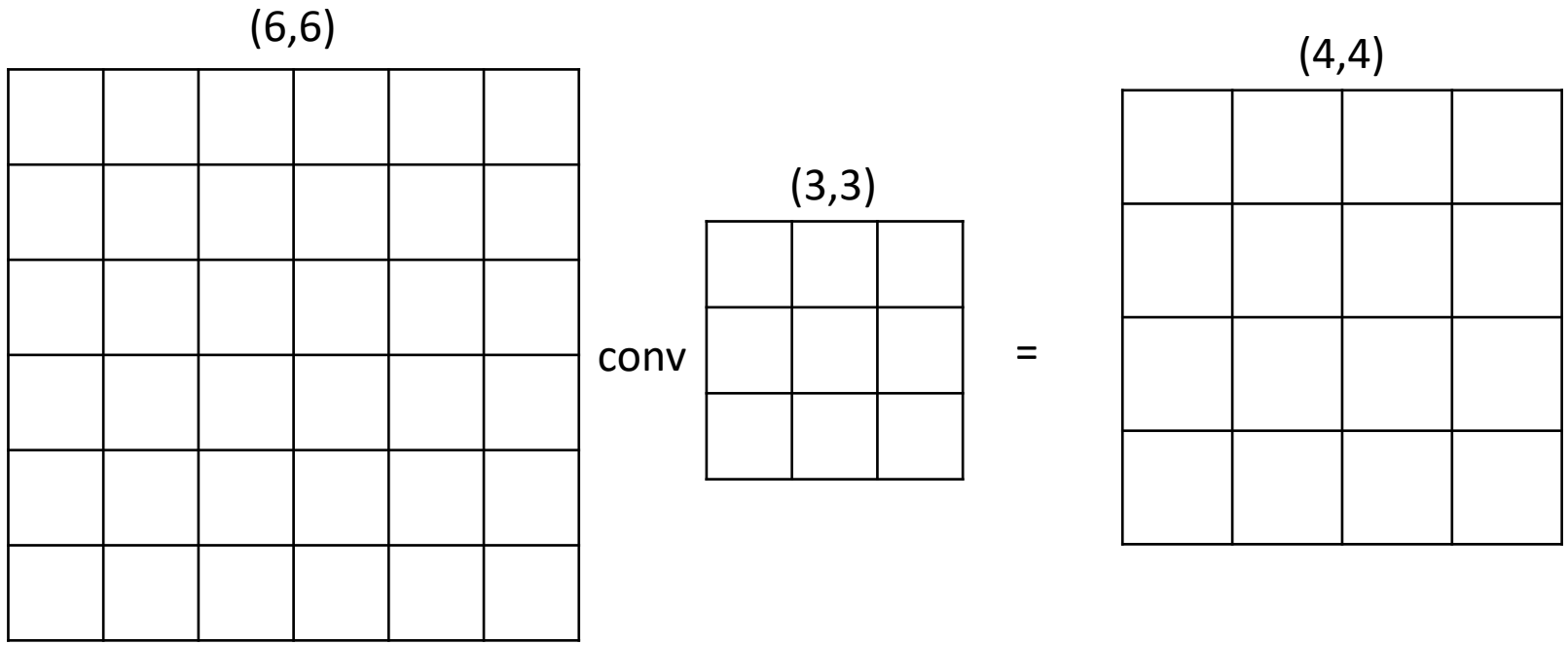
- More generally, filter is shape (f, f)
- Input image is shape (h, w)
- Output Shape: $(?, ?)$



Brad Quinton, Scott Chin

Generalizing the shapes of filters and outputs

- More generally, filter is shape (f, f)
- Input image is shape (h, w)
- Output Shape: $(h - f + 1, w - f + 1)$



Summary So far for Convolutions:

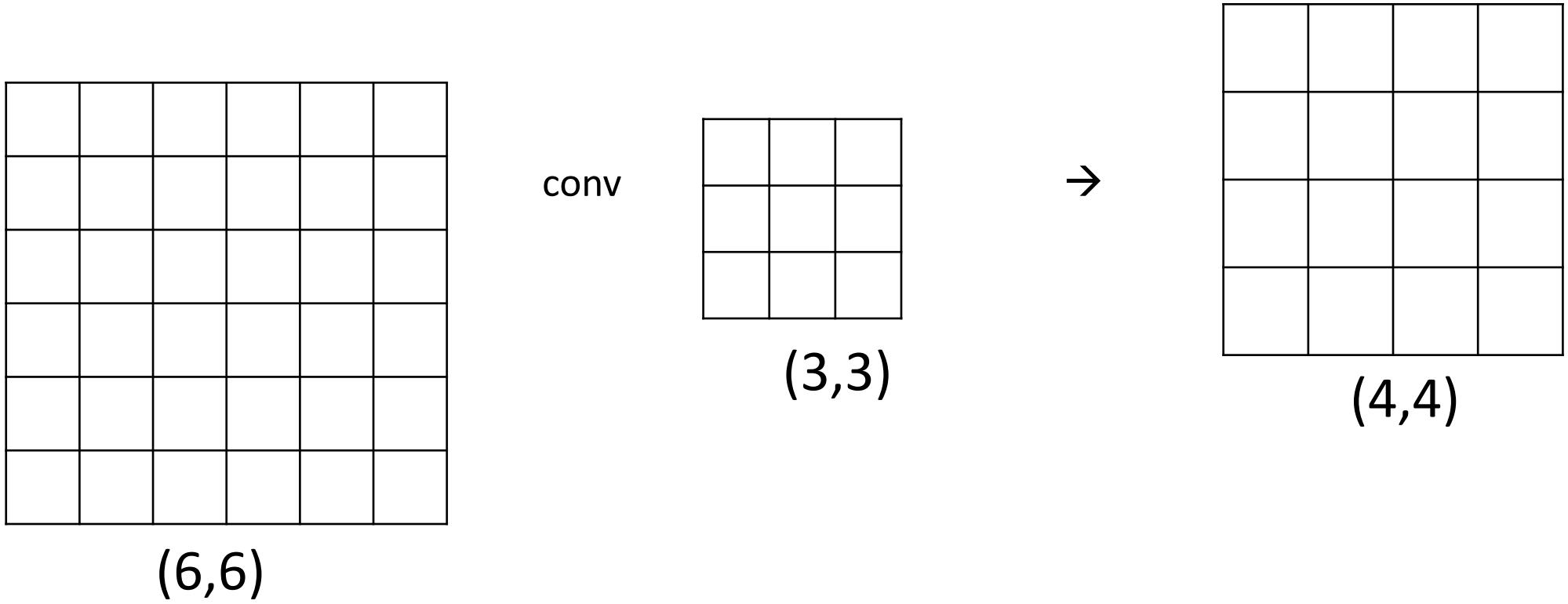
- Edges are just one kind of feature
- A filter can look for any arbitrary visual feature depending on its values
- A filter has shape (f, f) and square by convention
- Output of convolution is a Feature Map
- Feature Map is the intensity and location with which a feature is present in the input image
- Feature map shape: $(h - f + 1, w - f + 1)$ where input shape is (h, w)
- In Deep Learning, filters use learned parameters trained with the same techniques that we've talked about throughout the course.

Brad Quinton, Scott Chin

Convolution over 3D Data (Volumes)

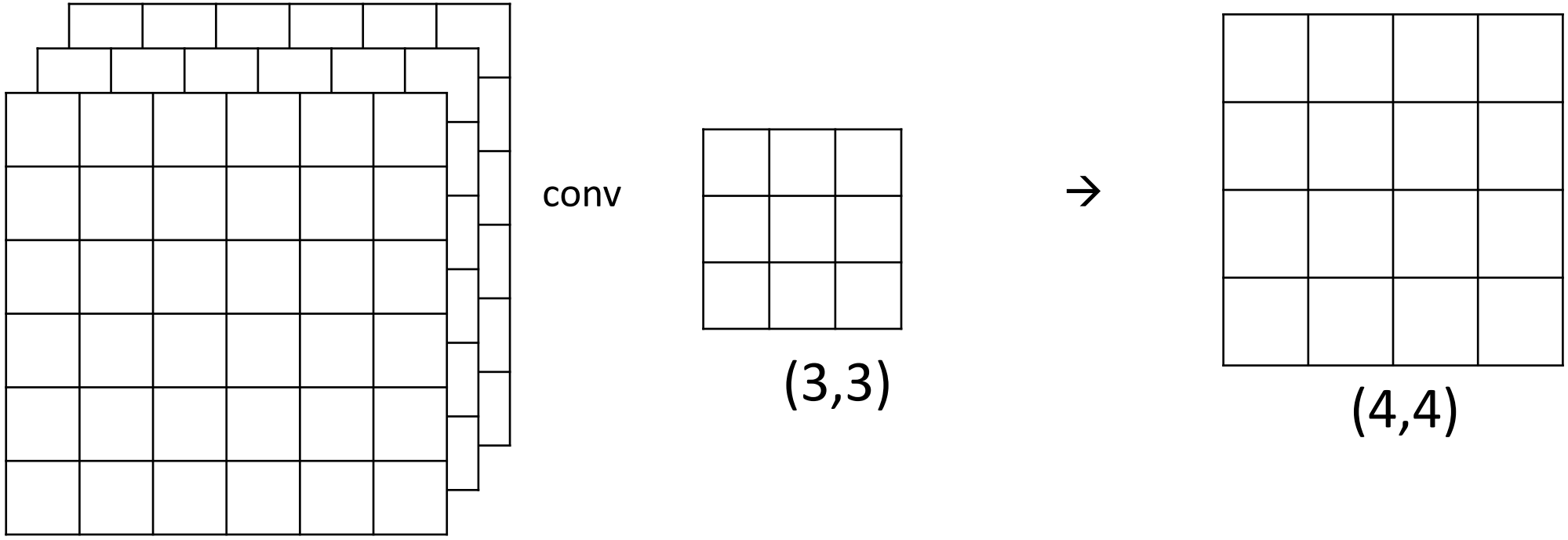
Brad Quinton, Scott Chin

Convolution Over 2D Data



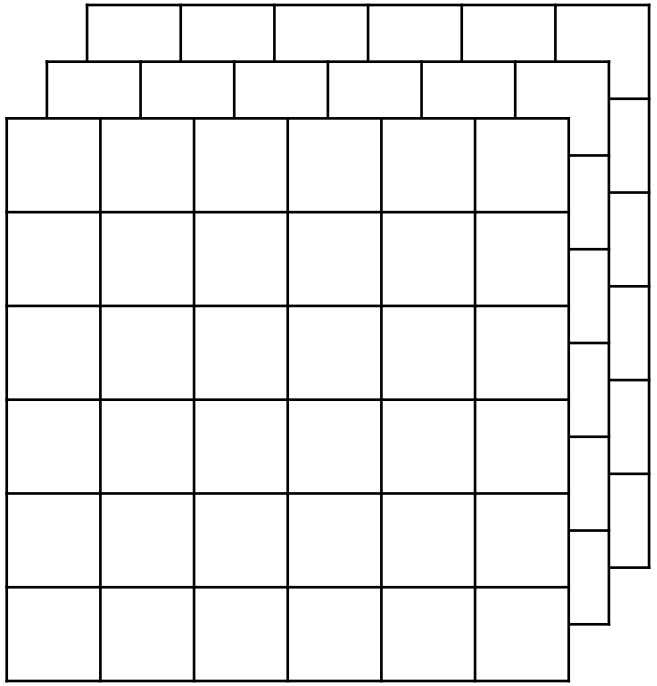
Brad Quinton, Scott Chin

Convolution Over Volumes



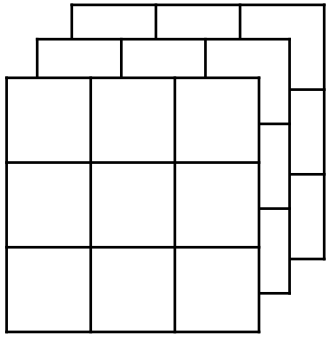
Call this dimension Channels

Convolution Over Volumes



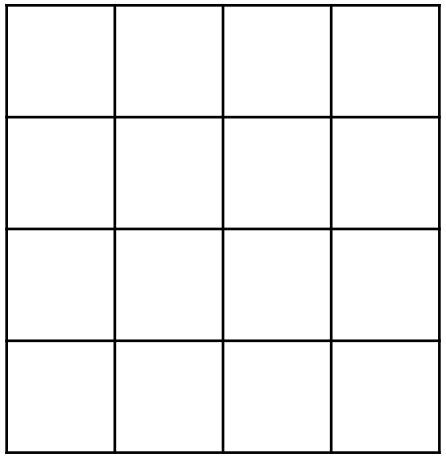
(6,6,3)

conv



(3,3,3)

→



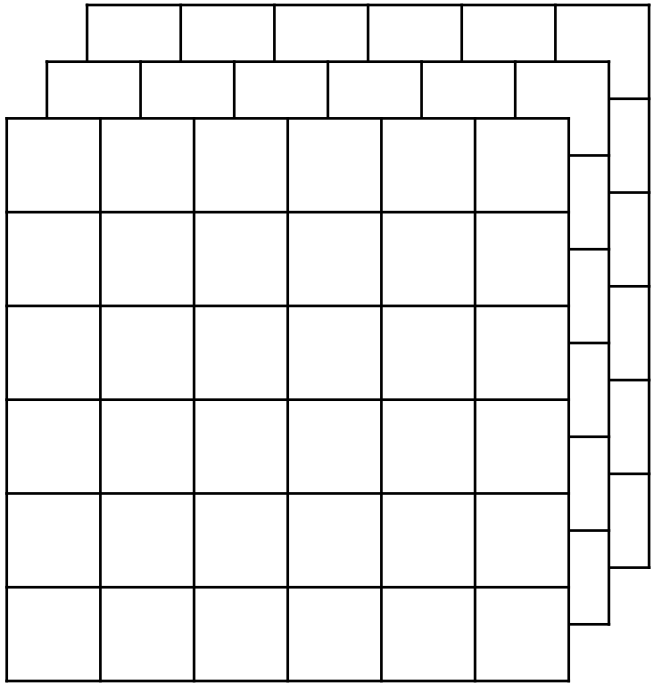
(4,4)

Call this dimension Channels

- Filters **must** also have the same number of channels as the input data
- Still doing elementwise multiply and sum
- Output shape remains the same

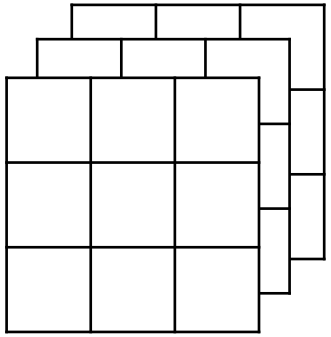
Brad Quinton, Scott Chin

Convolution Over Volumes



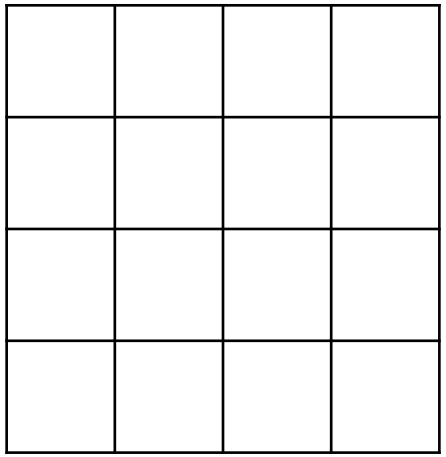
(h, w, c)

conv



(f, f, c)

→



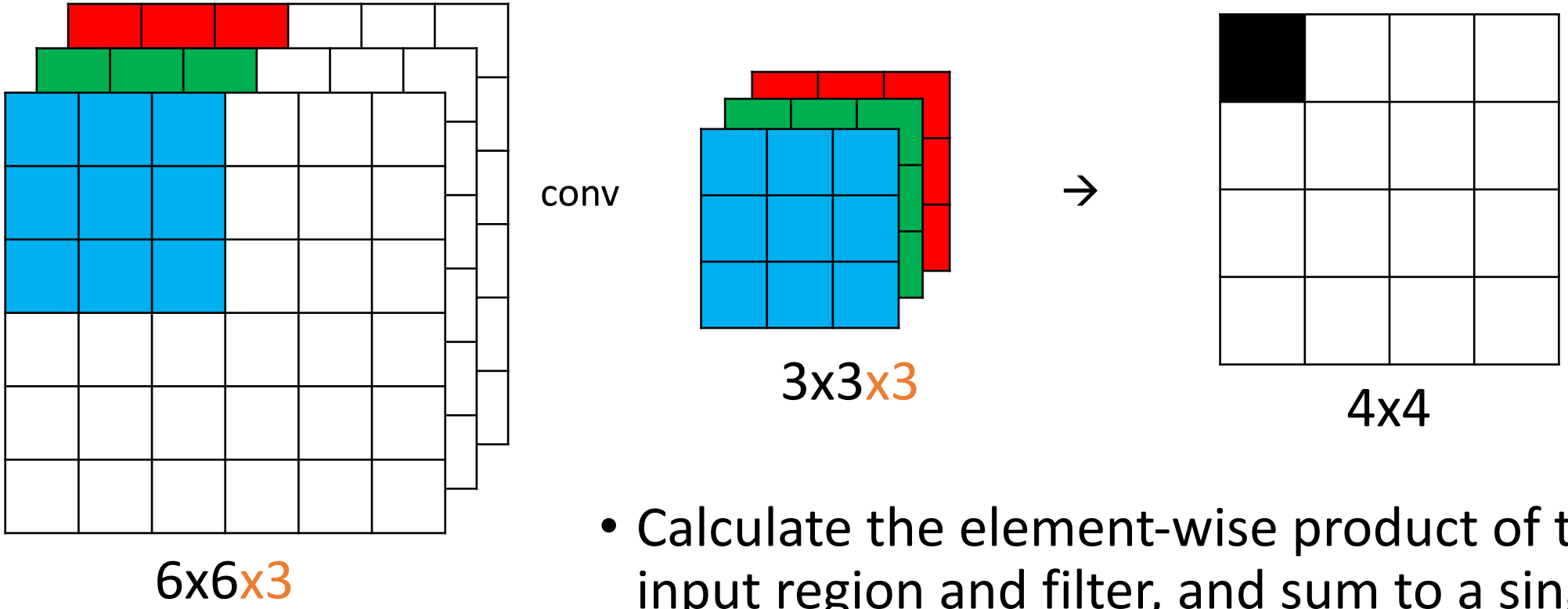
$(h - f + 1, w - f + 1)$

Call this dimension Channels

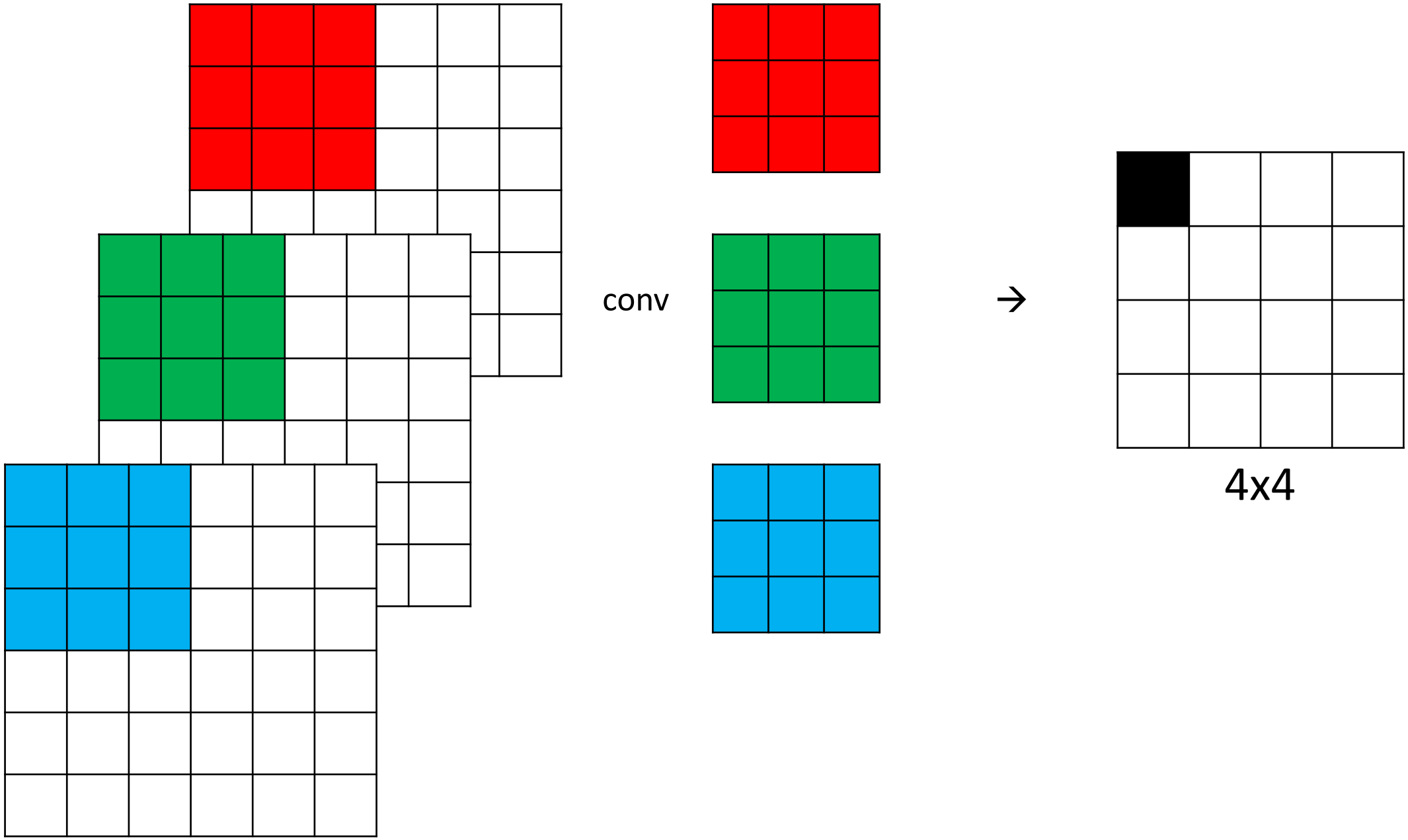
- Filters **must** also have the same number of channels as the input data
- Still doing elementwise multiply and sum
- Output shape remains the same

Brad Quinton, Scott Chin

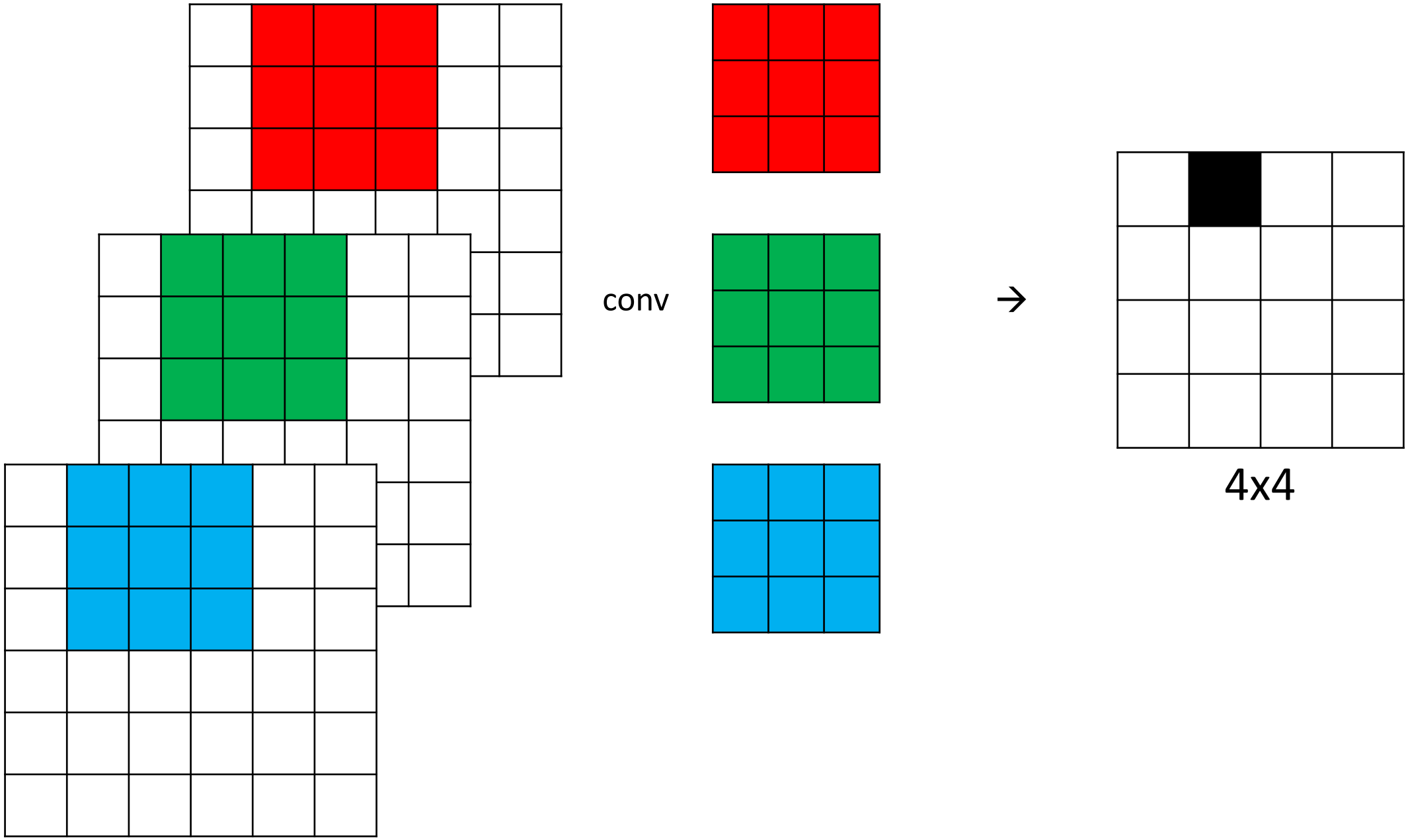
Example: Convolution Over Volumes



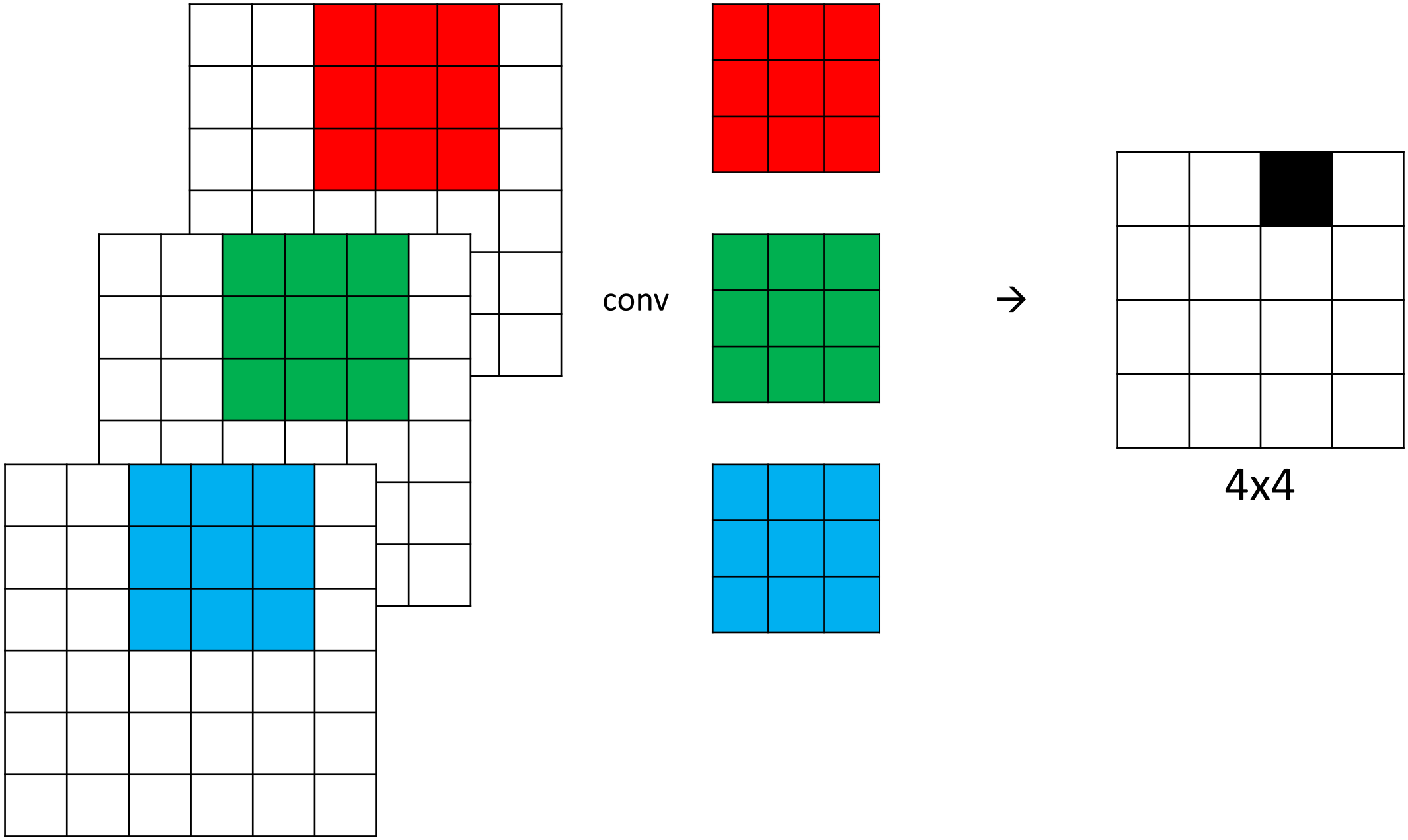
- Calculate the element-wise product of the input region and filter, and sum to a single number → similar to 2D

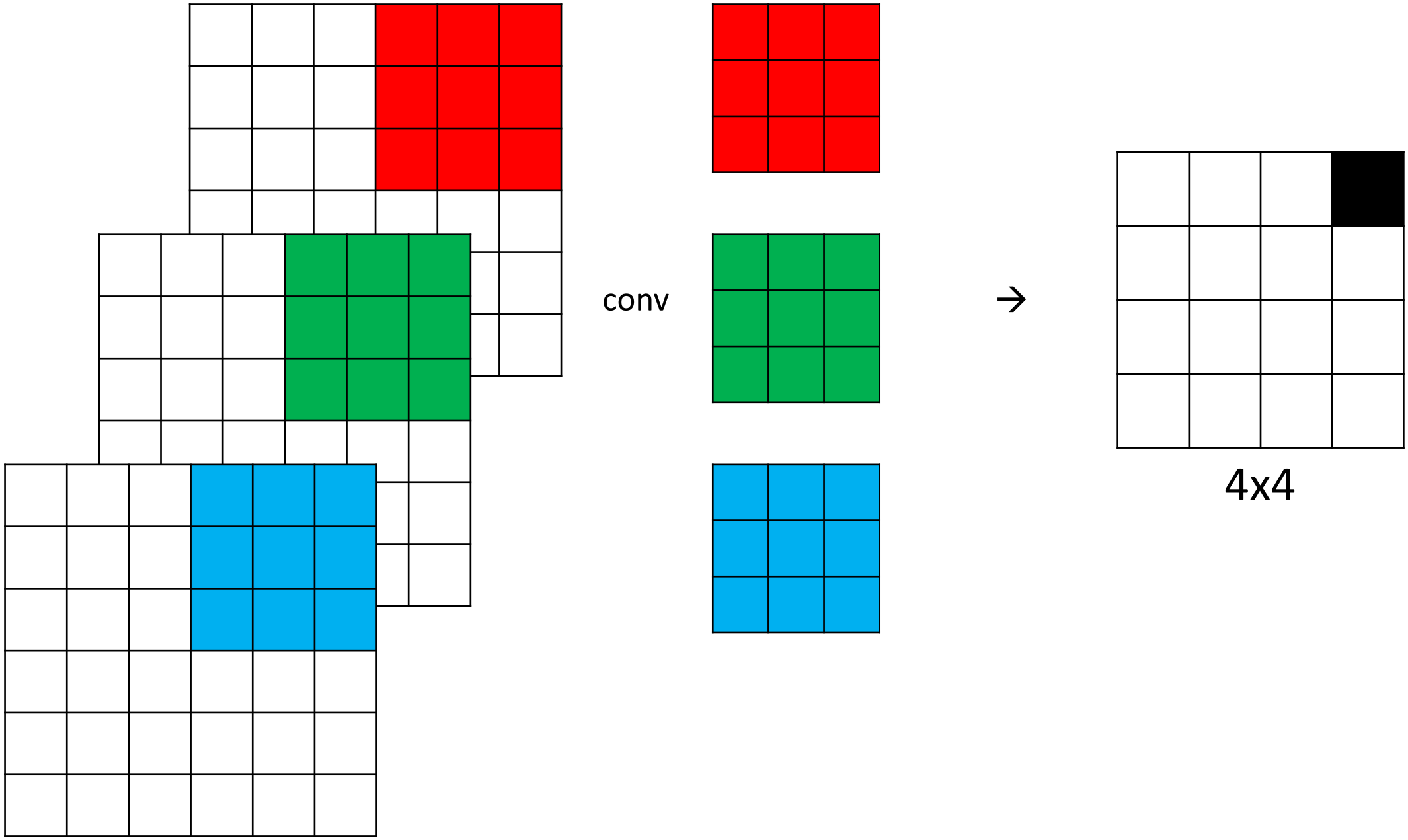


Brad Quinton, Scott Chin

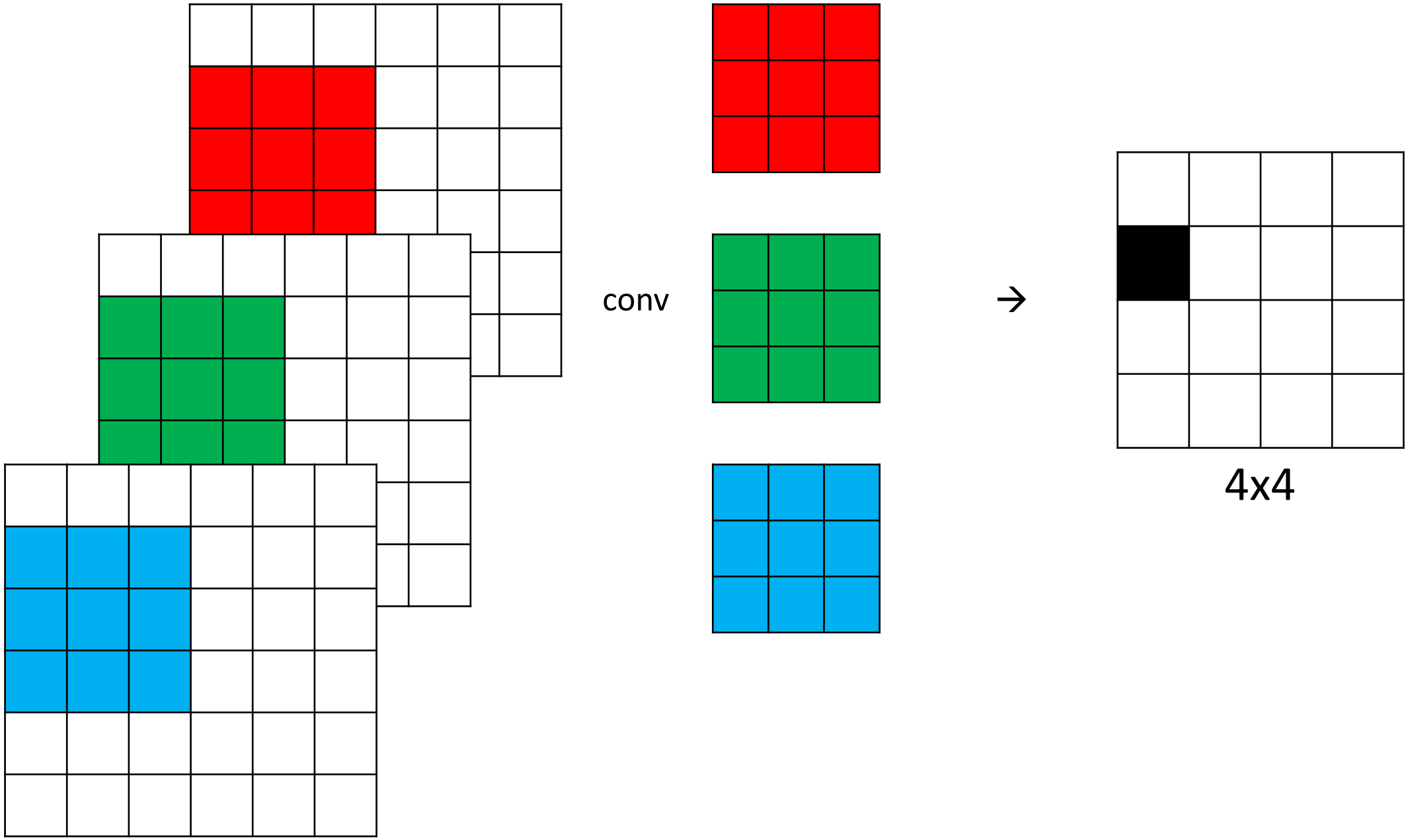


Brad Quinton, Scott Chin

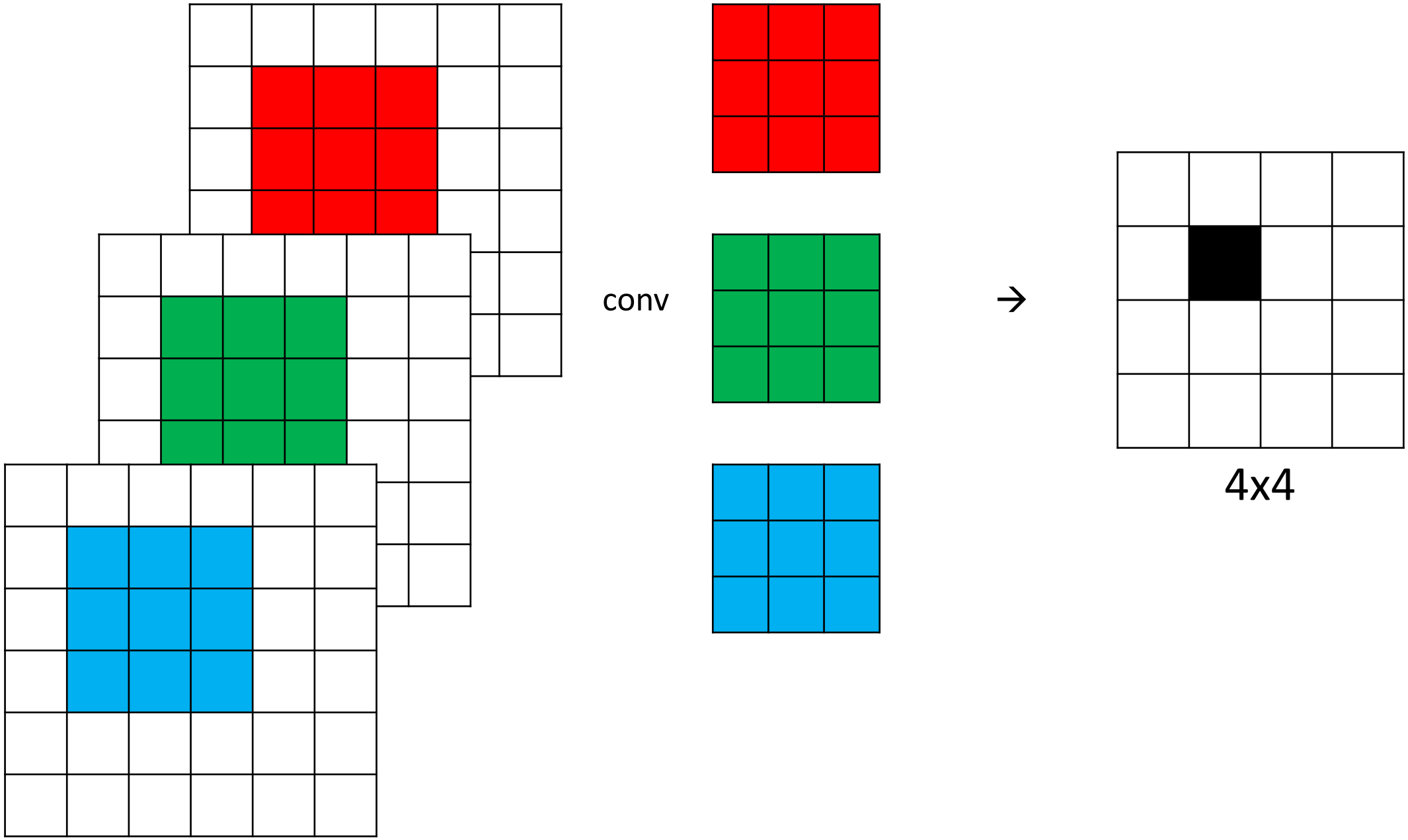

conv
→
4x4

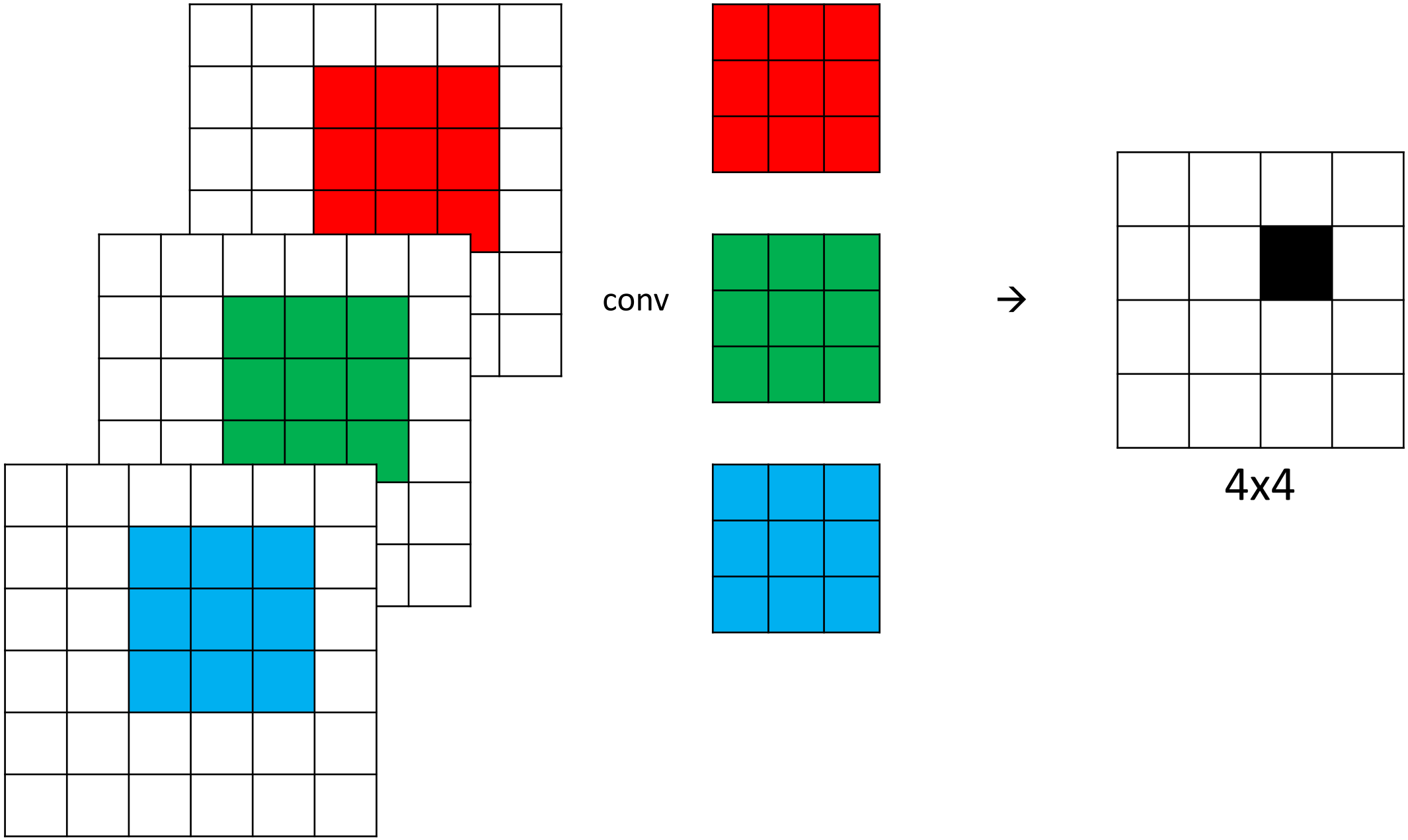


Brad Quinton, Scott Chin

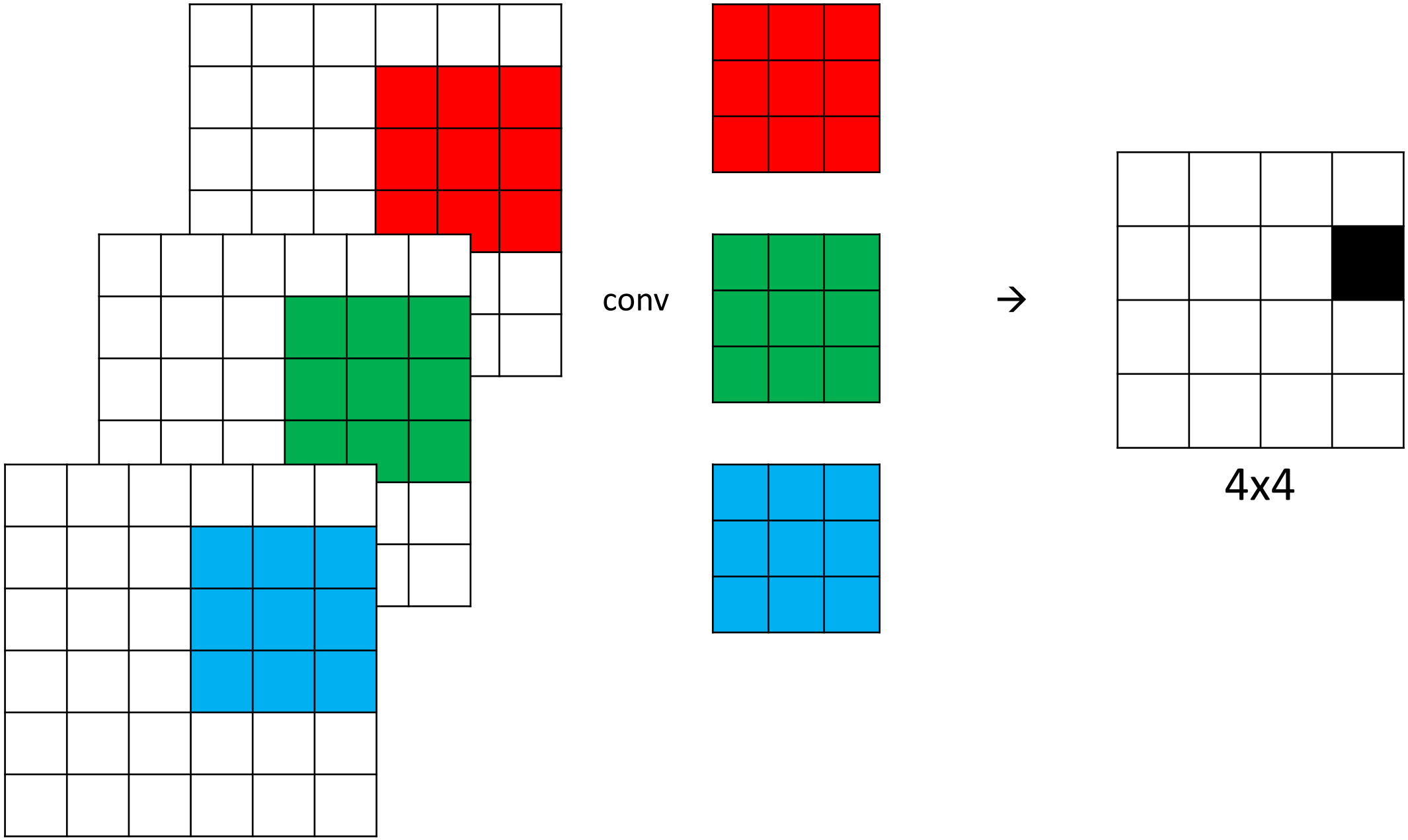


Brad Quinton, Scott Chin

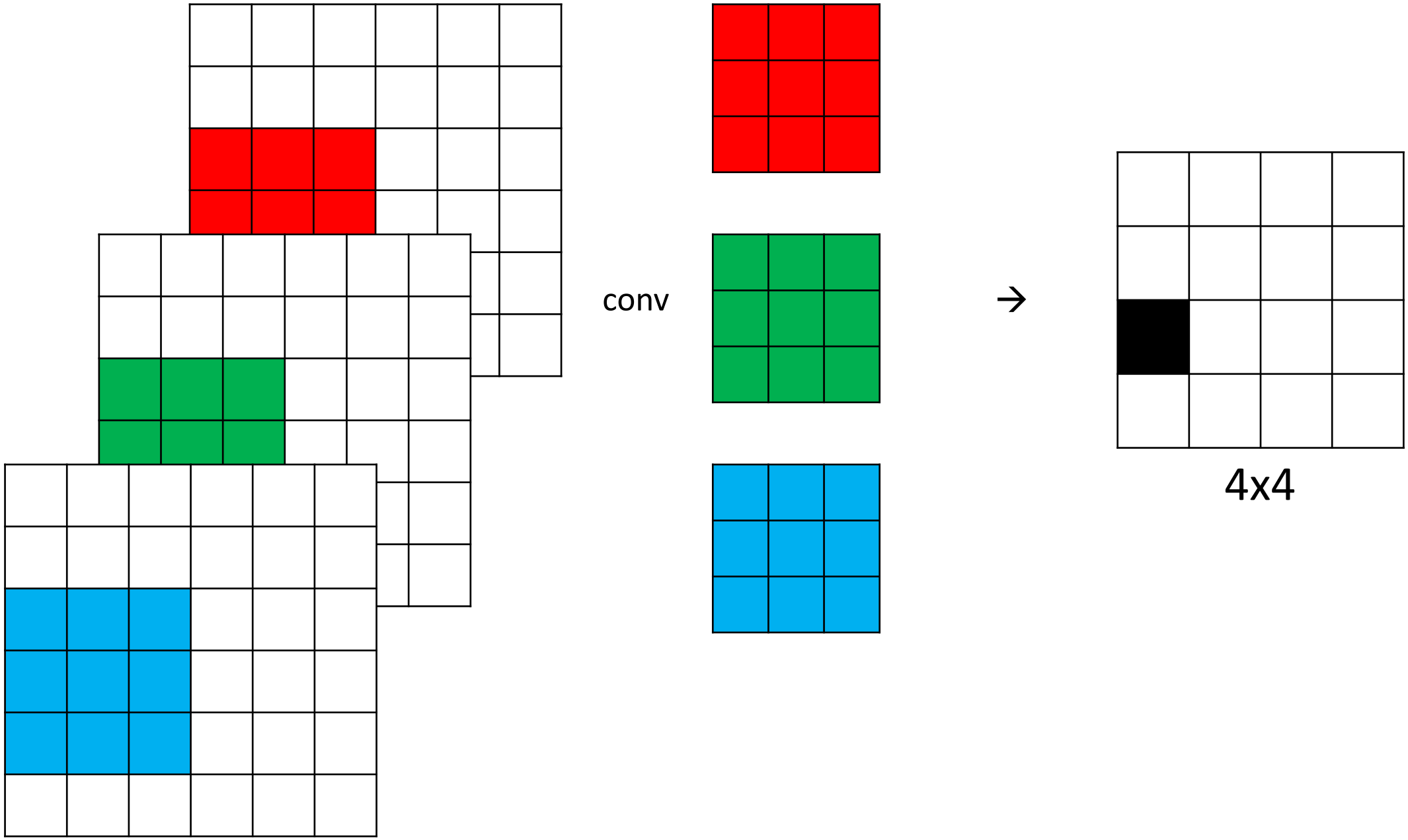

conv
→
4x4



Brad Quinton, Scott Chin



Brad Quinton, Scott Chin



Brad Quinton, Scott Chin

Summary – Convolution Over Volumes

- By convention, refer to filter shape as (f, f), but there is an implicit understanding that it has the same number of channels as the input.
- True shape of filter tensor is (f, f, c)
- Intuition is that, for color image, all three color channels are describing the same spatial objects except it is decomposed into three different color (channels). Therefore, we want a filter that looks through all channels.
- One output is the element-wise product and sum of filter with a region of the input
- Output is still a map with shape $(h - f + 1, w - f + 1)$

Learning Objectives

- Introduction to Convolutional Neural Networks and their applications
- Working towards an Image Classification Model
- Discuss inefficiencies of using a fully-connected neural network for image applications
- Become familiar with the Convolution operation and Convolutional Filters for 2D and 3D data
- Understand edge detection using manually chosen filter values

Brad Quinton, Scott Chin