Convolutional Neural Networks 2

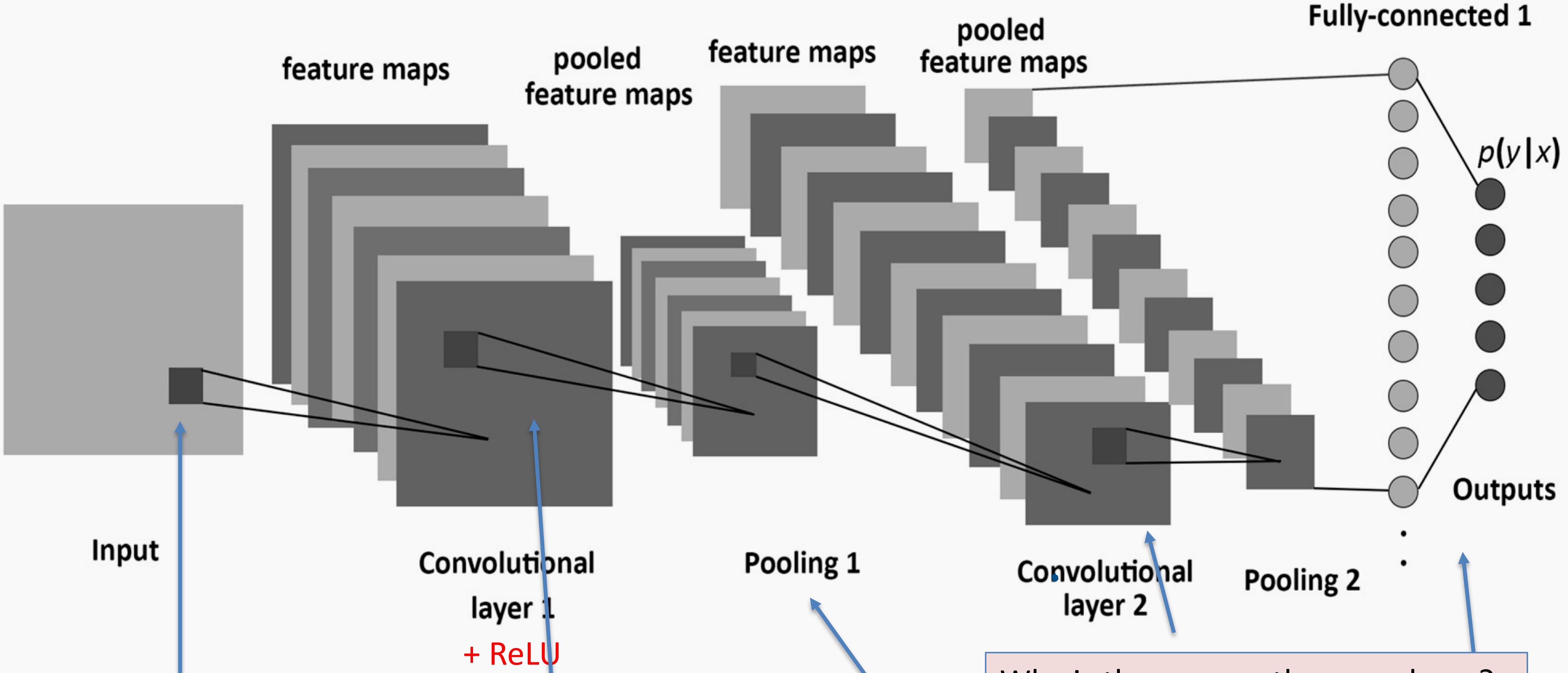
CS109B Data Science 2

Pavlos Protopapas, Mark Glickman, and Chris Tanner

Outline

- Review/Questions
- What are filters?
- What are the dimensions of filters and how we apply from one layer to the next?
- What is Pooling?
- What Activation Functions do we use?
- Why do we have a Dense Layer?

A Convolutional Network



What is this square?

What are feature maps and why are there multiple feature maps?

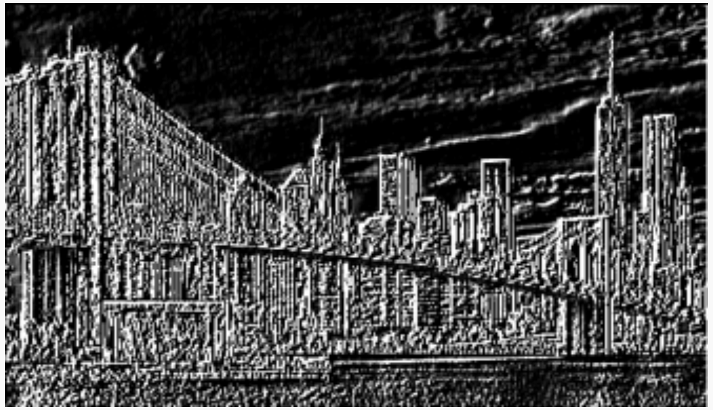
What is Pooling?

Why is there more than one layer?

What is going on here?

Grayscale

Edge



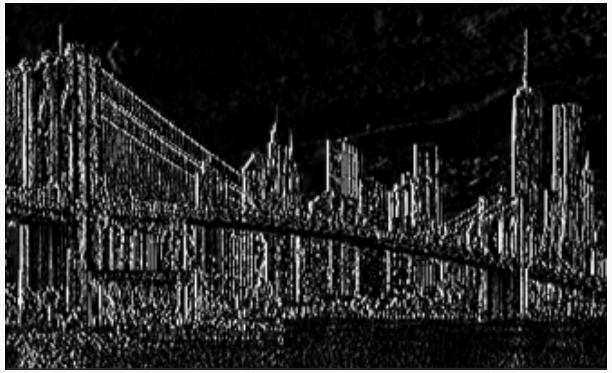
$*$

$$\begin{bmatrix} 0.0625 & 0.125 & 0.0625 \\ 0.125 & 0.25 & 0.125 \\ 0.0625 & 0.125 & 0.0625 \end{bmatrix} \quad \begin{bmatrix} 0 & -1 & 0 \\ -1 & 20 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -3 & 0 & 3 \\ -6 & 0 & 6 \\ -3 & 0 & 3 \end{bmatrix} \quad \begin{bmatrix} -3 & -6 & -3 \\ 0 & 0 & 0 \\ 3 & 6 & 3 \end{bmatrix}$$

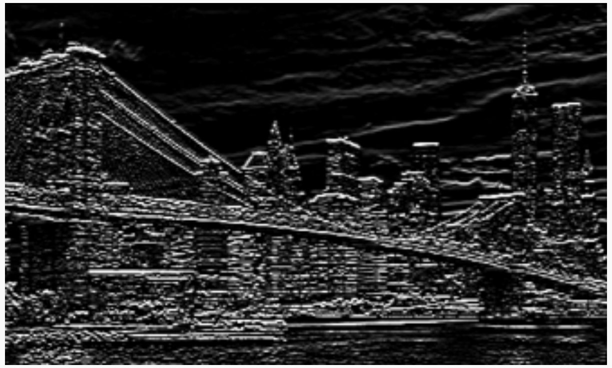
$=$

BLURRING

SHARPENING

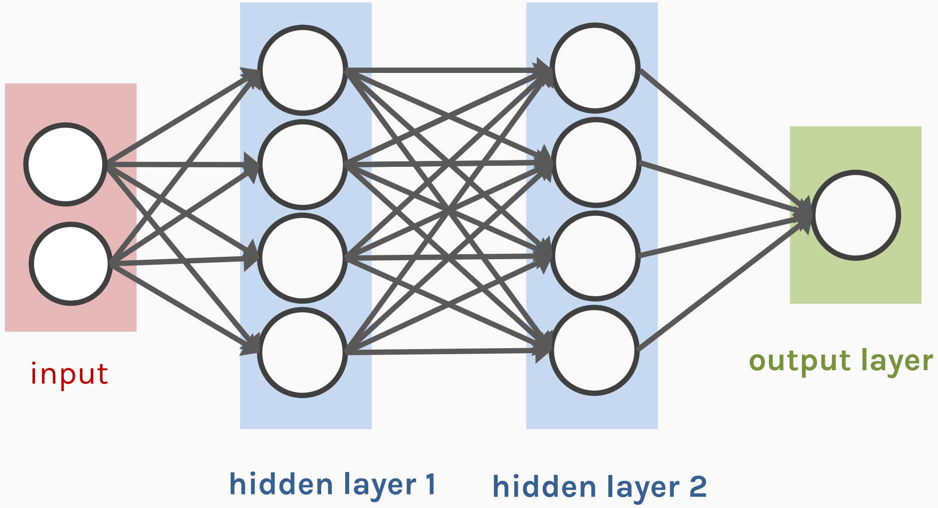


VERTICAL LINES

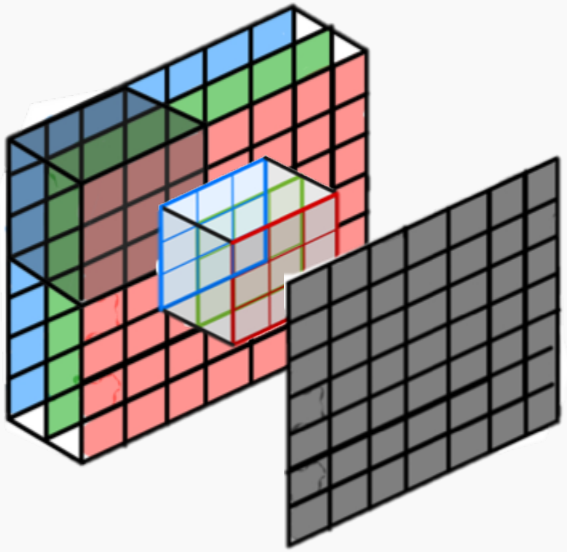


HORIZONTAL LINES

Basics of CNNs



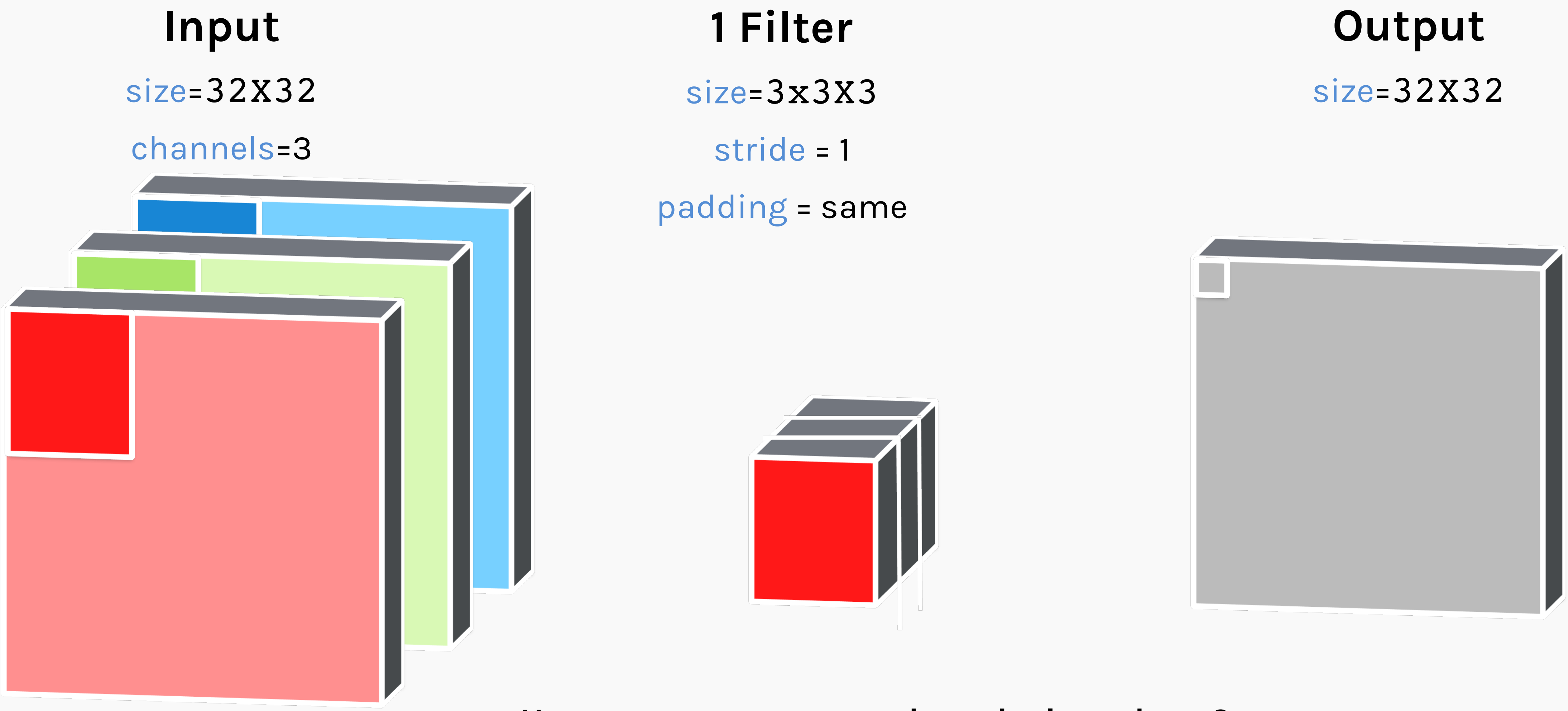
MLP



CNN

- CNNs are composed of layers, but those layers are not fully connected: they have filters, sets of cube-shaped weights, that are applied throughout the image.
- Each 2D slice of the filters are called kernels.
- These filters introduce translation invariance and parameter sharing.
- How are they applied? Convolution!

Example: A convolutional layer with one 3x3 filter that takes an 32x32 RGB image as input.



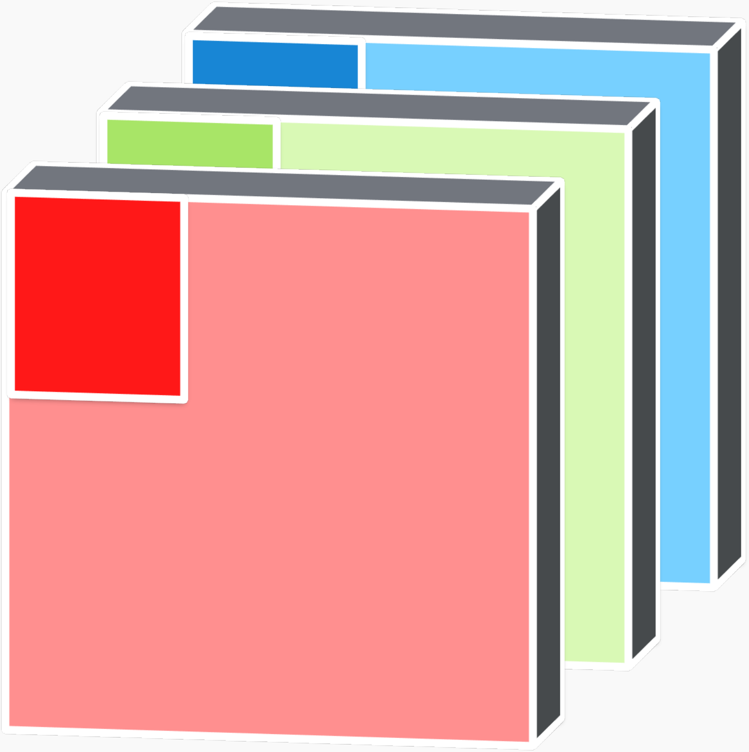
How many parameters does the layer have?

Example: A convolutional layer with one 3x3 filter that takes an 32x32 RGB image as input.

Input

size=32X32

channels=3

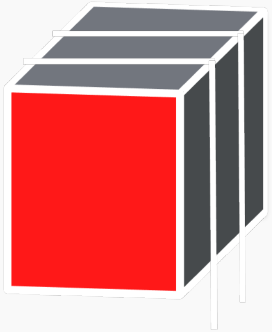


1 Filter

size=3x3X3

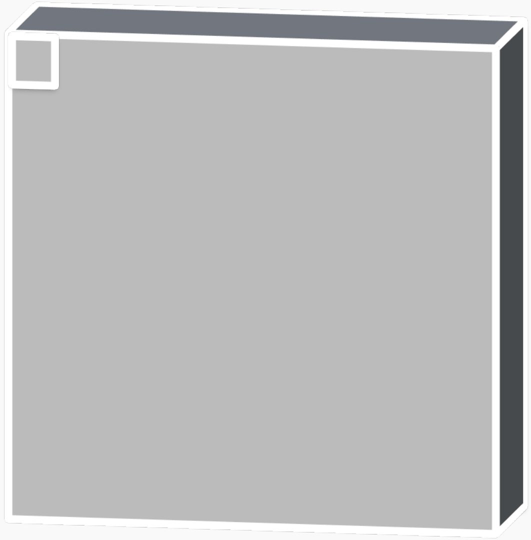
stride = 1

padding = same



Output

size=32X32



How many parameters does the layer have?

n_filters x filter_volume + biases = total number of params

1 x (3 x 3 x 3) + 1 = 28

Examples

A convolutional layer with `16 3x3` filters that takes `32x32` RGB image as input.

- How many parameters does the layer have?

Examples

A convolutional layer with `16 3x3` filters that takes `32x32` RGB image as input.

- How many parameters does the layer have?

$\boxed{16}$

Number of filters

Examples

A convolutional layer with `16 3x3` filters that takes `32x32` RGB image as input.

- How many parameters does the layer have?

$\boxed{16} \times \boxed{3 \times 3}$

Number of filters

Size of Filters

Examples

A convolutional layer with `16 3x3` filters that takes `32x32` RGB image as input.

- How many parameters does the layer have?

$$\underbrace{\boxed{16}}_{\text{Number of filters}} \times \underbrace{\boxed{3 \times 3}}_{\text{Size of Kernel}} \times \underbrace{\boxed{3}}_{\text{Number of channels of prev layer}}$$

Examples

A convolutional layer with `16 3x3` filters that takes `32x32` RGB image as input.

- How many parameters does the layer have?

$$\boxed{16} \times \boxed{3 \times 3} \times \boxed{3} + \boxed{16}$$

Number of filters

Size of Kernel

Number of channels of prev layer

Biases (one per filter)

Examples

A convolutional layer with `16 3x3` filters that takes `32x32` RGB image as input.

- How many parameters does the layer have?

$$\boxed{16} \times \boxed{3 \times 3} \times \boxed{3} + \boxed{16} = 448$$

Number of filters

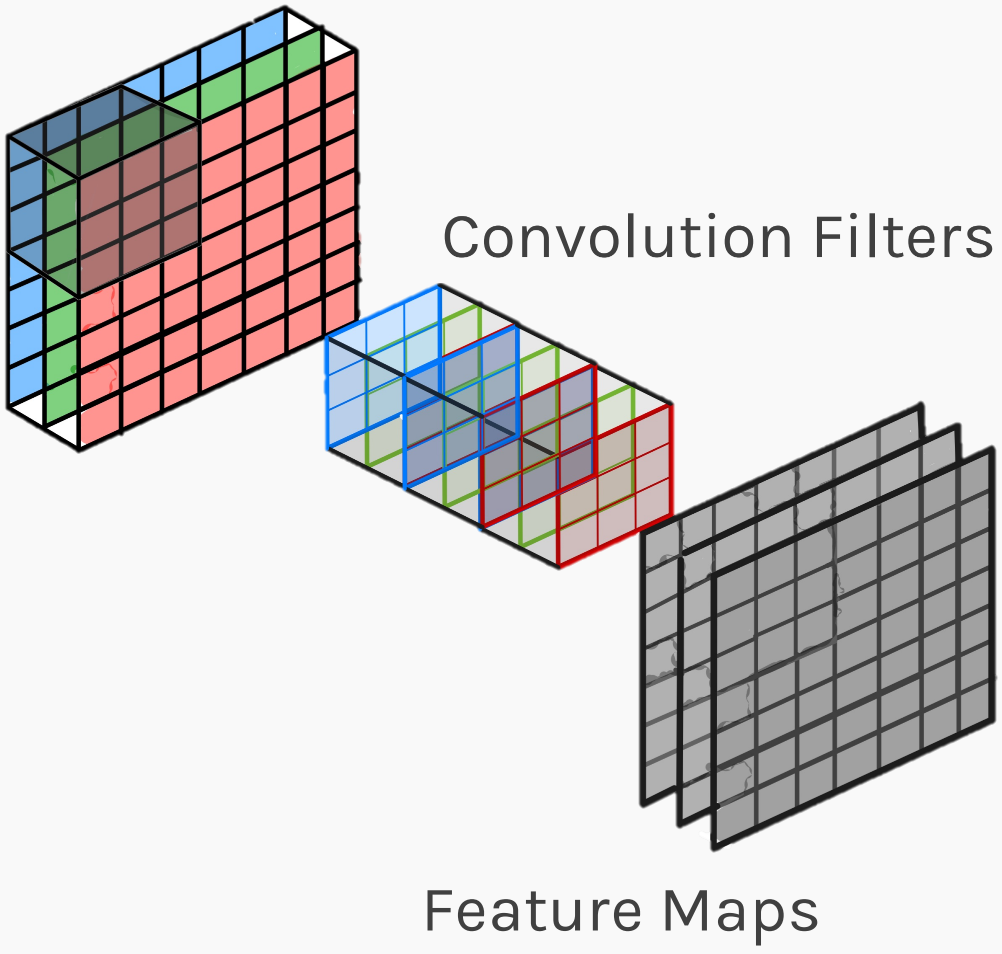
Size of Kernel

Number of channels of prev layer

Biases (one per filter)

Convolutional layers (cont)

- To be clear: each filter is convolved with the entirety of the 3D input cube but generates a 2D feature map.
- Because we have multiple filters, we end up with a 3D output: one 2D feature map per filter.
- The feature map dimension can change drastically from one conv layer to the next: we can enter a layer with a `32x32x16` input and exit with a `32x32x128` output if that layer has 128 filters.



Traning CNN

In a convolutional layer, we are basically applying multiple filters over the image to extract different features.

But most importantly, we are learning those filters!

Traning CNN

In a convolutional layer, we are basically applying multiple filters over the image to extract different features.

But most importantly, we are learning those filters!

HOW? We use BackProp and SGD as we did with FCNN

Traning CNN

In a convolutional layer, we are basically applying multiple filters over the image to extract different features.

But most importantly, we are learning those filters!

HOW? We use BackProp and SGD as we did with FCNN

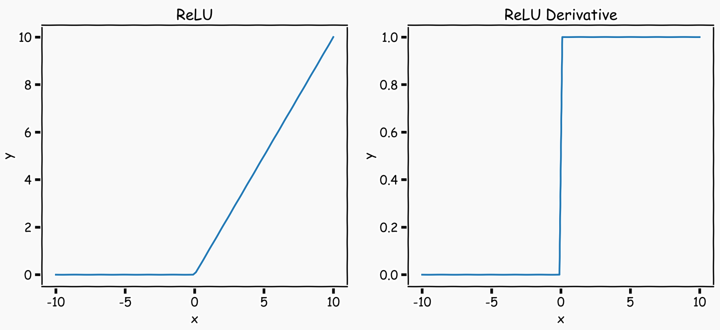
One thing we're missing: non-linearity.

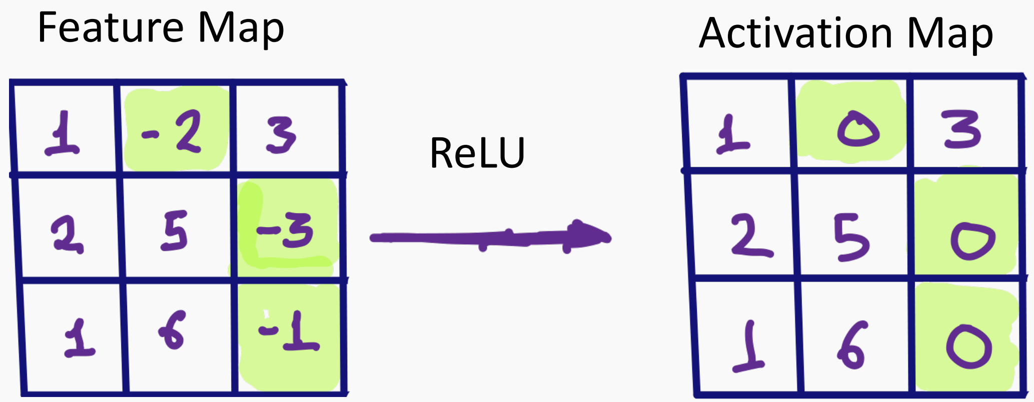
ReLU

We apply non-linear activation **after** convolution as we did for FCNN.

The most successful non-linear activation function for CNNs is the Rectified Non-Linear unit (ReLU):

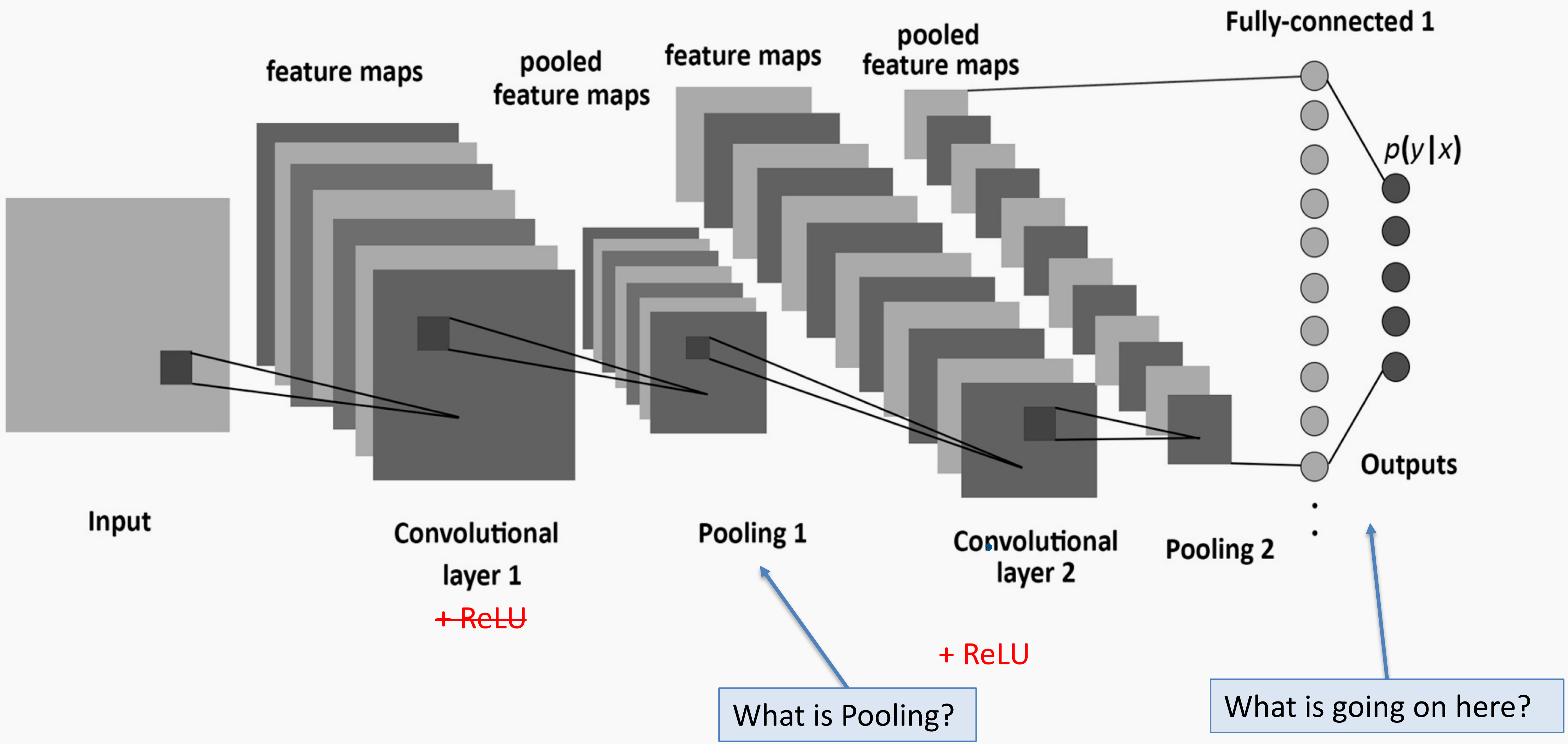
$$y = \max(0, x)$$





This combats the vanishing gradient problem occurring in sigmoid, it is easier to compute, and generates sparsity.

A Convolutional Network



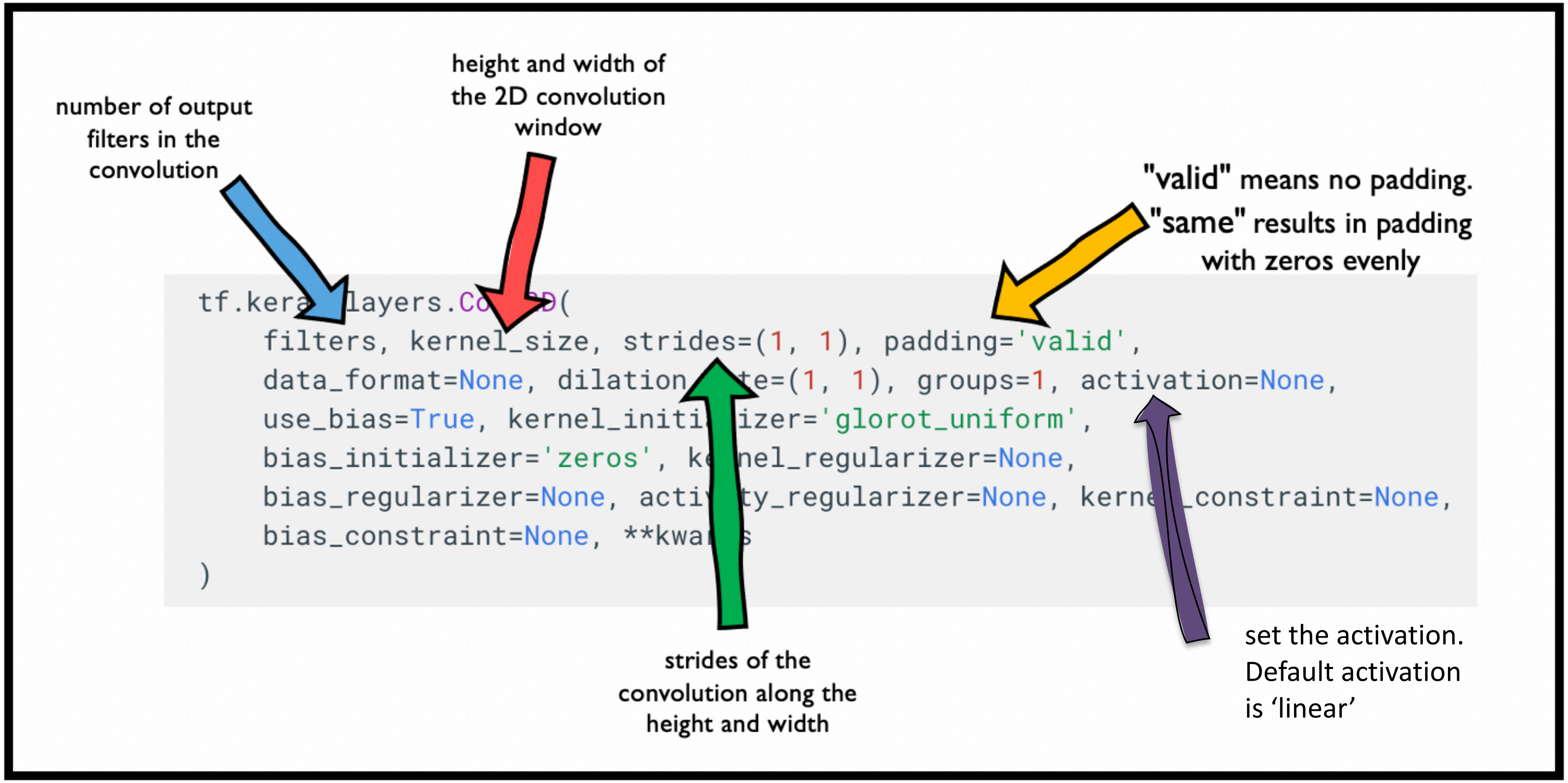
Convolutional layers so far

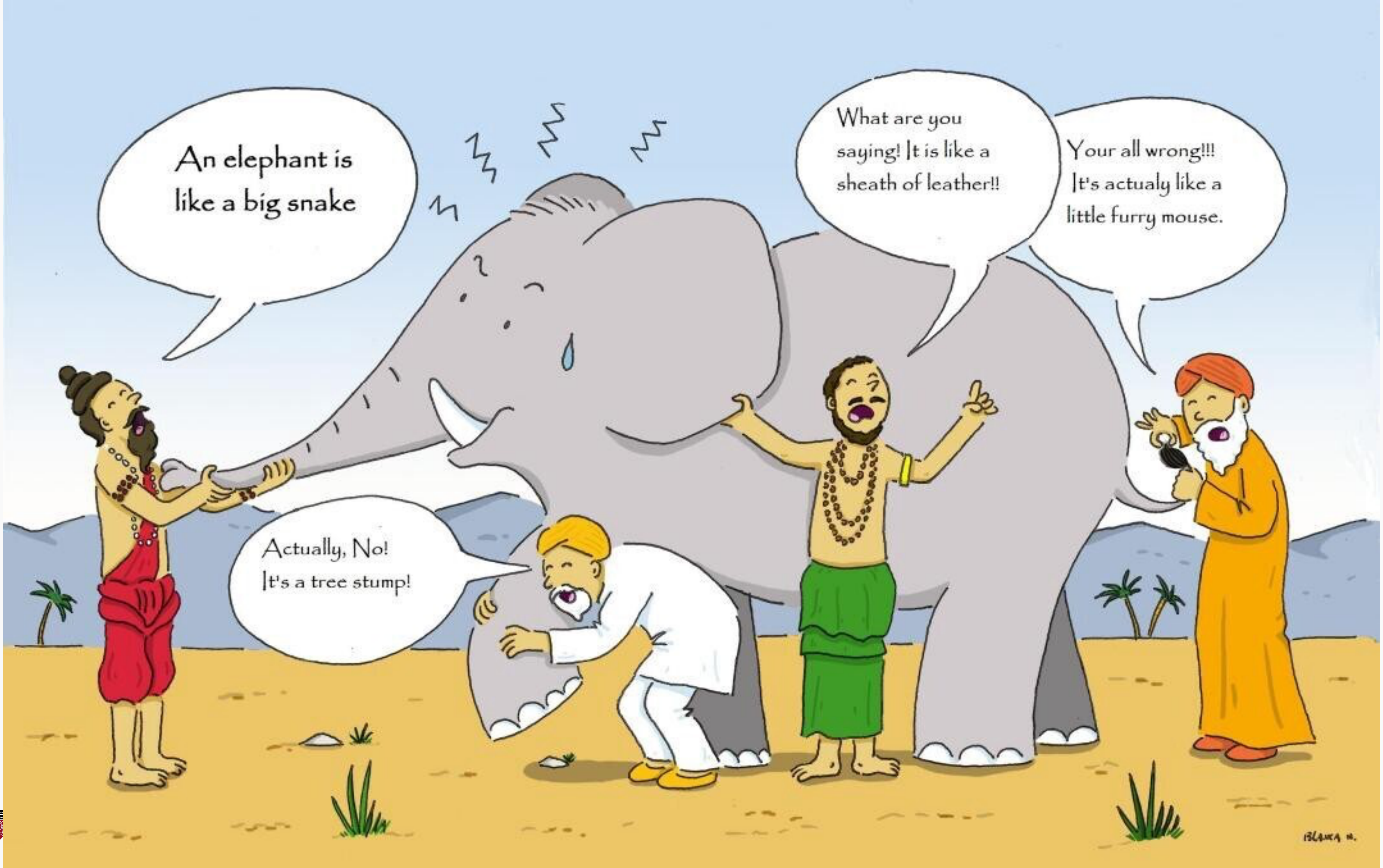
Multiple parameters to define:

- number of filters
- size of kernels
- stride
- padding
- activation function to use

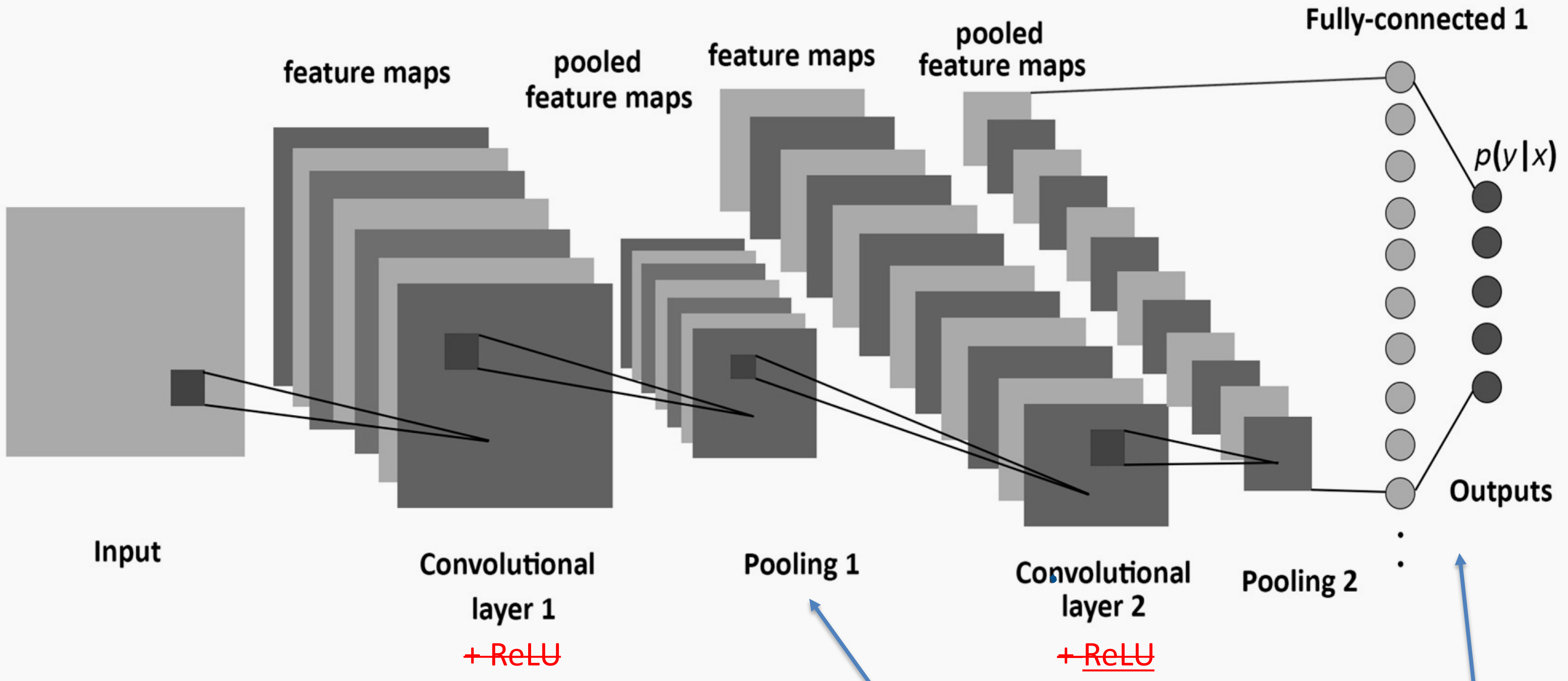
Convolutional layers so far

Multiple parameters to define:




An elephant is like a big snake
What are you saying! It is like a sheath of leather!!
Your all wrong!!! It's actualy like a little furry mouse.
Actually, No! It's a tree stump!

A Convolutional Network



What is Pooling?

What is going on here?

Pooling

A **pooling** layer is a new layer added after the convolutional layer. Specifically, it is added after a nonlinearity (e.g. ReLU) has been applied to the feature maps*.

The pooling layer operates upon each activation map separately to create a new set of the same number of pooled feature maps.

* Maxpooling could be applied before ReLU.

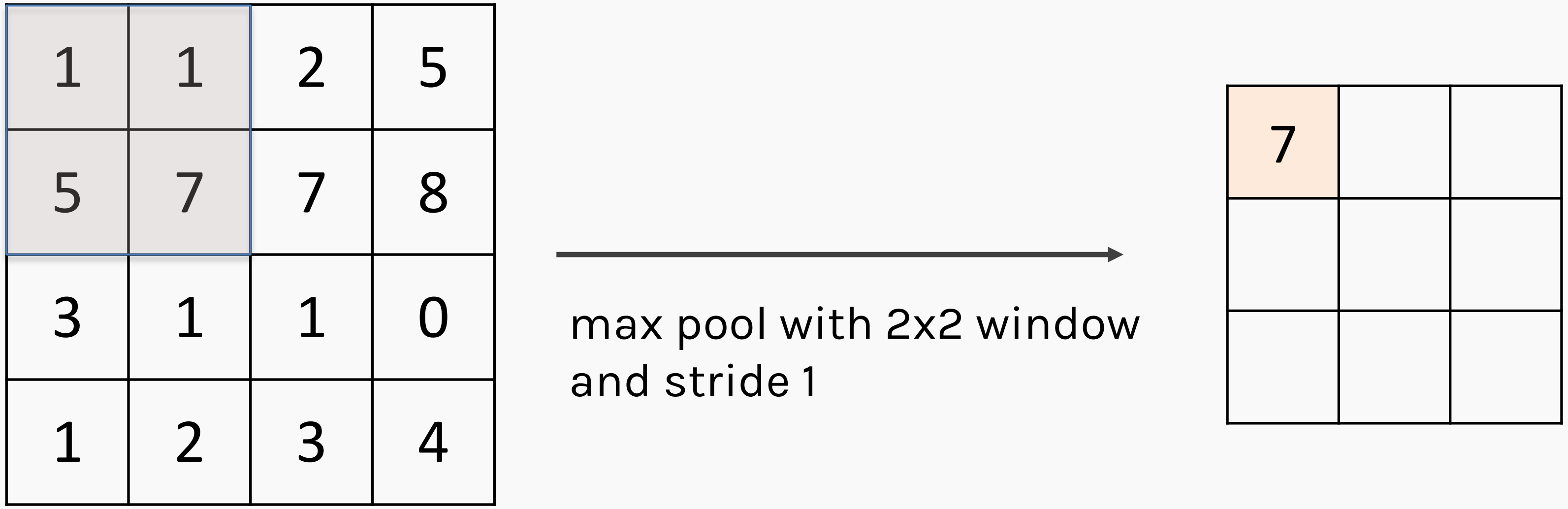
25

Pooling

A **pooling** layer is a new layer added after the convolutional layer. Specifically, it is added after a nonlinearity (e.g. ReLU) has been applied to the feature maps.

The pooling layer operates upon each activation map separately to create a new set of the same number of pooled feature maps.

Example:



26

Pooling

A **pooling** layer is a new layer added after the convolutional layer. Specifically, it is added after a nonlinearity (e.g. ReLU) has been applied to the feature maps.

The pooling layer operates upon each activation map separately to create a new set of the same number of pooled feature maps.

Example:

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

max pool with 2x2 window and stride 1

7	7	

27

Pooling

A **pooling** layer is a new layer added after the convolutional layer. Specifically, it is added after a nonlinearity (e.g. ReLU) has been applied to the feature maps.

The pooling layer operates upon each activation map separately to create a new set of the same number of pooled feature maps.

Example:

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

max pool with 2x2 window and stride 1

7	7	8

28

Pooling

A **pooling** layer is a new layer added after the convolutional layer. Specifically, it is added after a nonlinearity (e.g. ReLU) has been applied to the feature maps.

The pooling layer operates upon each feature map separately to create a new set of the same number of pooled feature maps.

Pooling involves selecting:

- A pooling operation, much like a filter, to be applied to feature maps: e.g. max, mean, median.
- The size of the pooling operation.
- The stride.

29

Pooling

<u>Pooling involves selecting:</u>

- A pooling operation, much like a filter, to be applied to feature maps: max, mean, median.
- The size of the pooling operation.
- The stride.

The size of the pooling operator must be smaller than the size of the feature map; specifically, it is <u>almost always</u> 2×2 applied with a stride of 2 using **max** pooling.

30

Pooling

<u>Pooling involves selecting:</u>

- A pooling operation, much like a filter to be applied to feature maps: max, mean, median.
- The size of the pooling operation.
- The stride.

The size of the pooling operator must be smaller than the size of the feature map; specifically, it is <u>almost always</u> 2×2 applied with a stride of 2 using **max** pooling.

Invariant to small, "local transitions"

Face detection: enough to check the presence of eyes, not their precise location

Reduces input size of the final fully connected layers (more later)

No learnable parameters

31

Pooling: example with stride 2x2

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

max pool with 2x2 window and stride 2x2

7	8
3	4

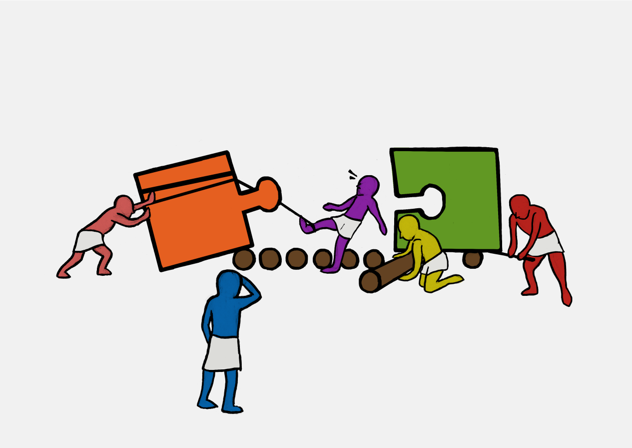
1	1	2	5
5	7	7	8
3	1	1	0
1	2	3	4

mean pool with 2x2 window and stride 2x2

3.5	5.5
1.75	2

32

Exercise: Pooling mechanics



The aim of this exercise is to understand the tf.keras implementation of average and max pooling:

- implement Max Pooling by building a model with a single MaxPooling2D layer
- Next, implement Average Pooling by building a model with a single AvgPooling2D layer
- Use the helper code to visualize the output
- Use the hint we provide

How far the pooling window moves for each pooling step

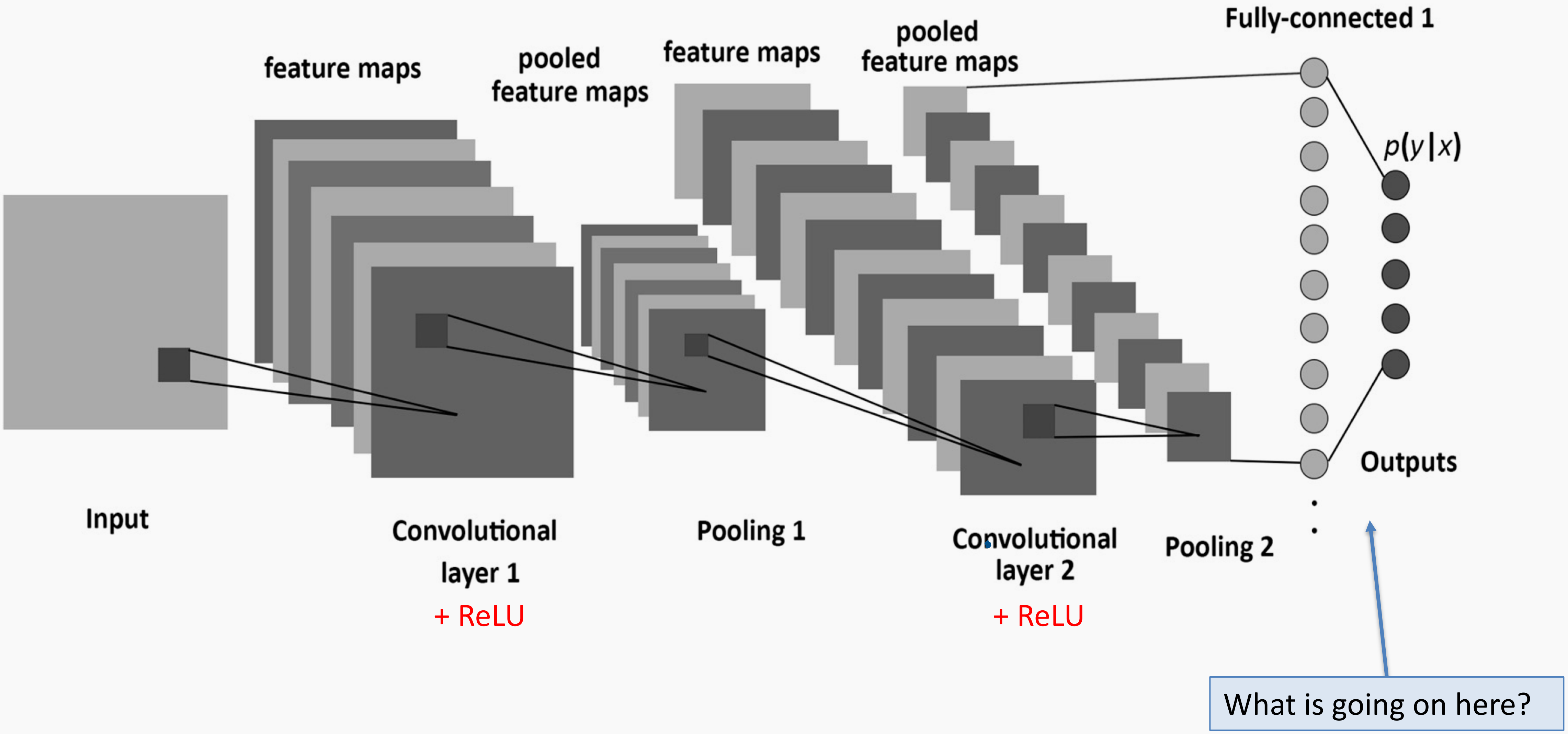
Channels first or channels last

```
tf.keras.layers.MaxPool2D(
    pool_size=(2, 2), strides=None, padding='valid', data_format=None,
    **kwargs
)
```

Window-size over which to take the maximum

"valid" means no padding, "same" results in padding

A Convolutional Network



Building a CNN

A convolutional neural network is built by stacking layers, typically of 3 types:

Convolutional Layers	Pooling Layers	Fully connected Layers

Building a CNN

Convolutional Layers

Action

- Apply filters to extract features
- Filters are composed of small kernels, learned
- One bias per filter
- Apply activation function on every value of feature map

Parameters

- Number of filters
- Size of kernels (W and H only, D is defined by input cube)
- Activation function
- Stride
- Padding

I/O

- Input: 3D cube, previous set of feature maps
- Output: 3D cube, one 2D map per filter

Building a CNN

A convolutional neural network is built by stacking layers, typically of 3 types:

Convolutional Layers	Pooling Layers	Fully connected Layers

Building a CNN

Pooling Layers

Action

- Reduce dimensionality
- Extract maximum or average of a region
- Sliding window approach

Parameters

- Stride
- Size of window

I/O

- Input: 3D cube, previous set of feature maps
- Output: 3D cube, one 2D map per filter, reduced spatial dimensions

Building a CNN

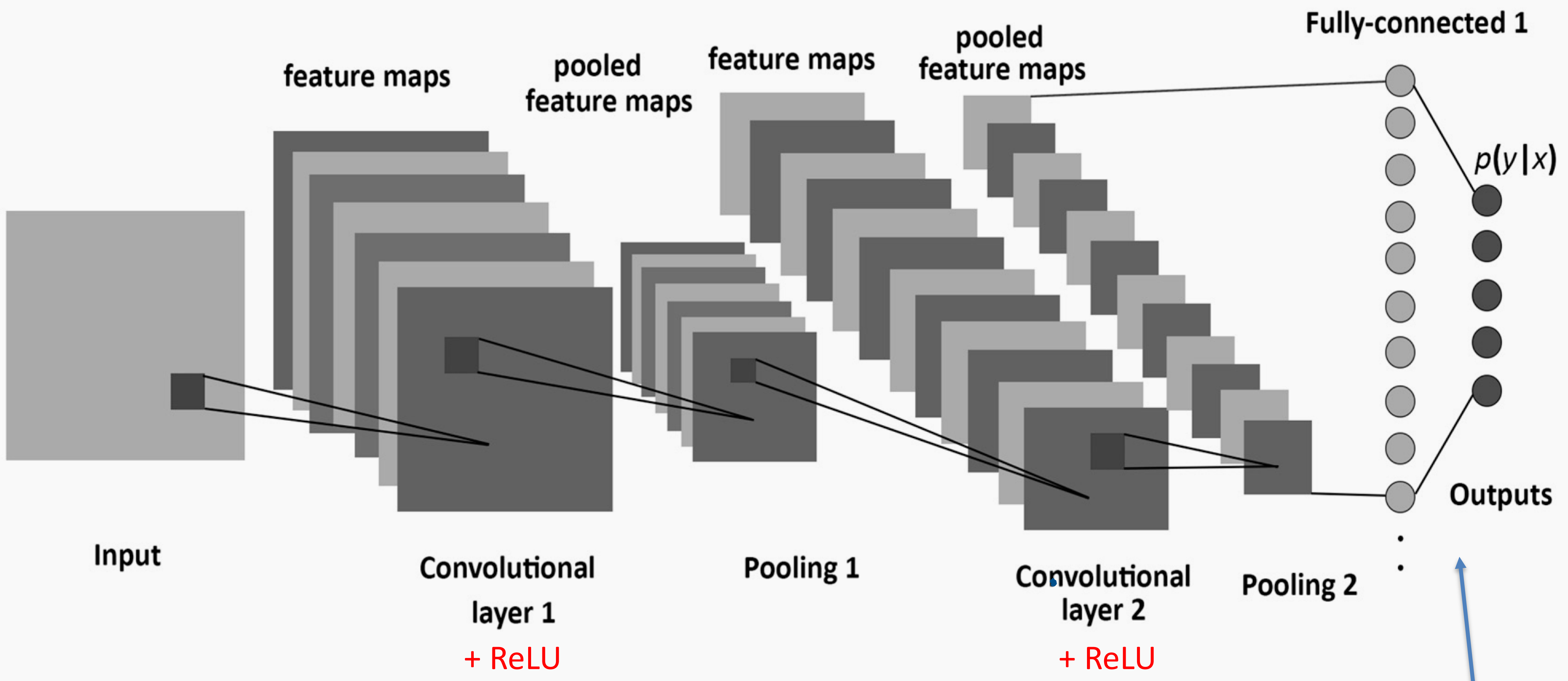
A convolutional neural network is built by stacking layers, typically of 3 types:

Convolutional Layers

Pooling Layers

Fully connected Layers

A Convolutional Network



What is going on here?

Building a CNN

Fully connected Layers

Action

- Aggregate information from final feature maps
- Generate final classification (or regression)

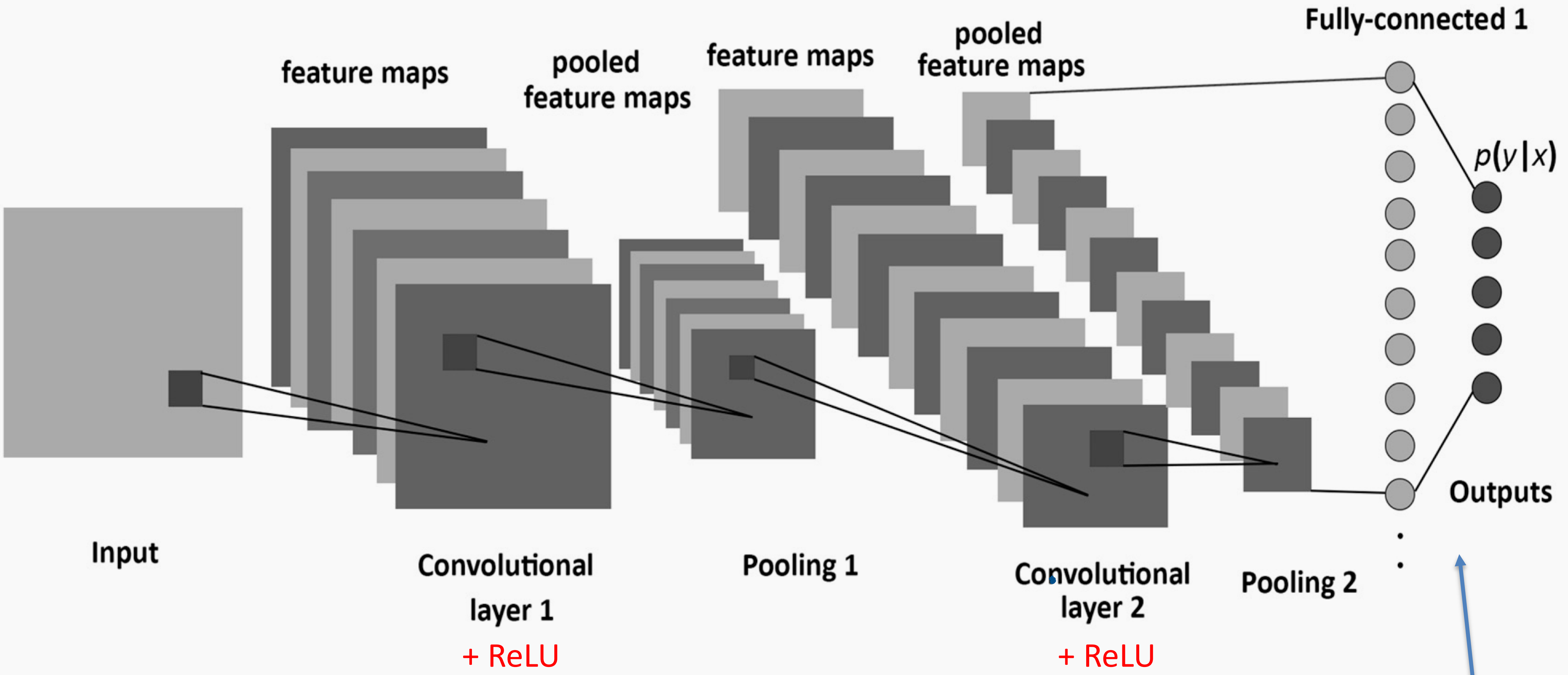
Parameters

- Number of nodes
- Activation function: usually changes depending on role of the layer. If aggregating info, use ReLU. If producing final classification, use Softmax. If regression use linear

I/O

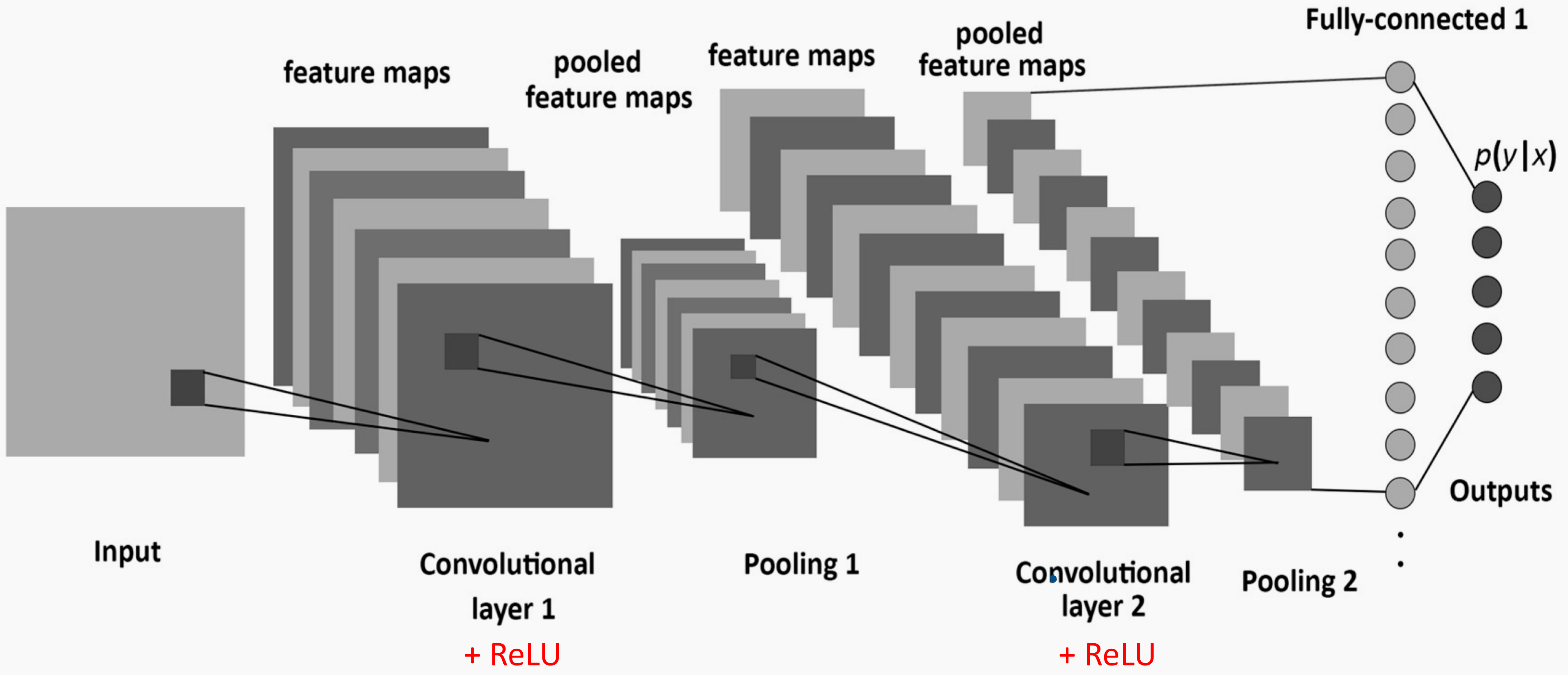
- Input: **FLATTENED** 3D cube, previous set of feature maps
- Output: Probabilities for each class or simply prediction for regression $\hat{y}$

A Convolutional Network



~~What is going on here?~~

A Convolutional Network



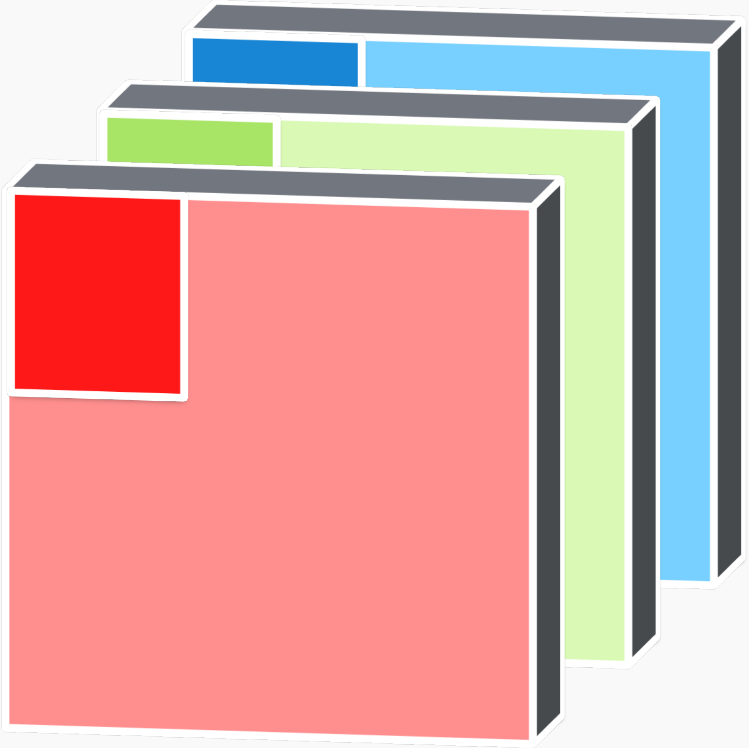
Examples

- Let **C** be a CNN with the following disposition:
  - Input: 32x32x3 images
  - Conv1: 8 3x3 filters, stride 1, padding=same
  - Conv2: 16 5x5 filters, stride 2, padding=same
  - Flatten layer (explained in the next few slides)
  - Dense1: 512 nodes
  - Dense2: 4 nodes
- How many parameters does this network have?

Input

size=32X32

channels=3

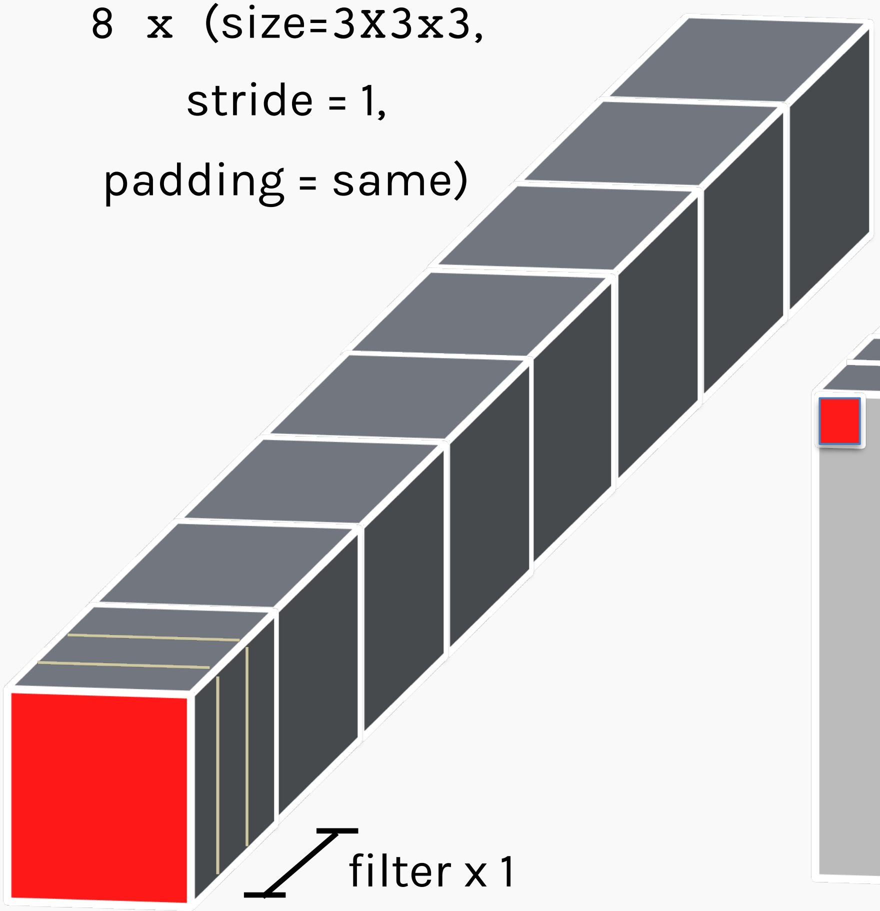


Filter

8 x (size=3X3x3,

stride = 1,

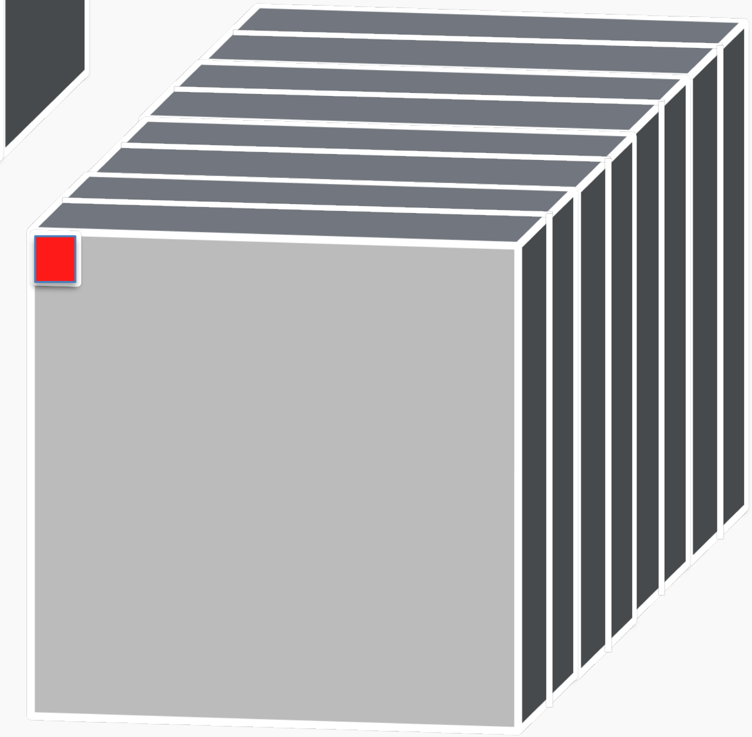
padding = same)



Output

(size=32X32,

channels = 8)



How many parameters does the layer have if I want to use 8 filters?

n_filters x filter_volume + biases = total number of params

$$8 \times (3 \times 3 \times 3) + 8 = 224$$

Input

(size=32X32,

channels=8)

Filter

16 x (size=5X5X8,

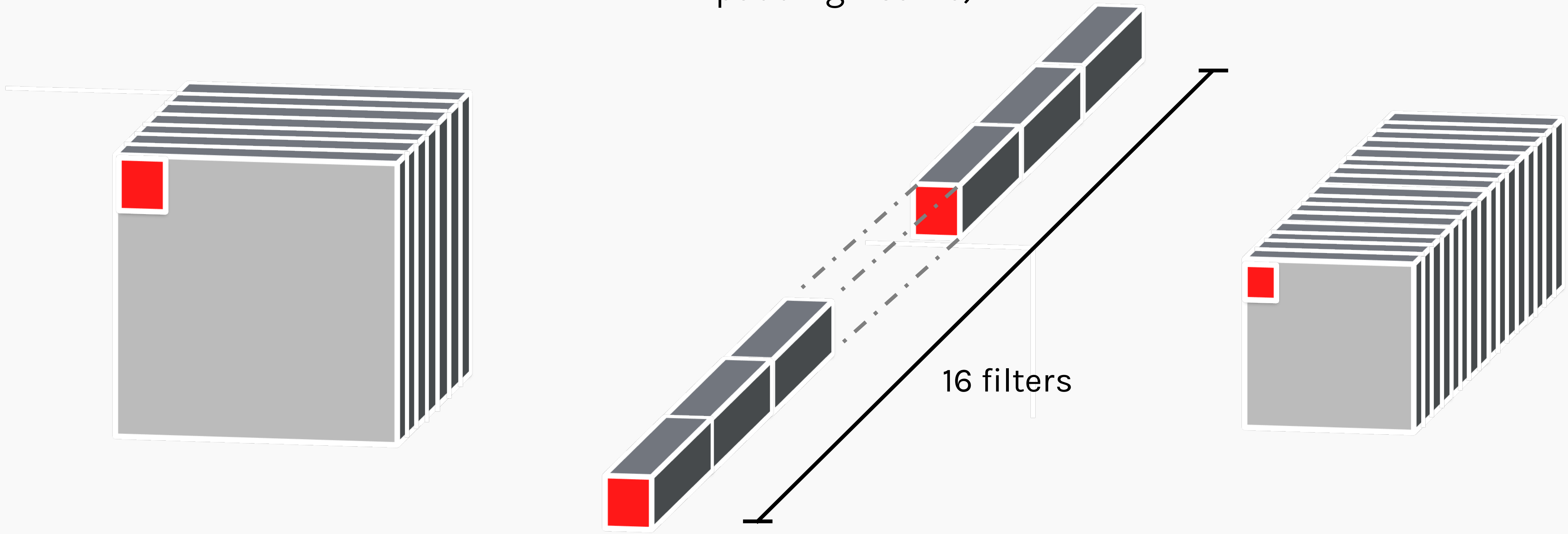
stride = 2,

padding = same)

Output

(size=16X16,

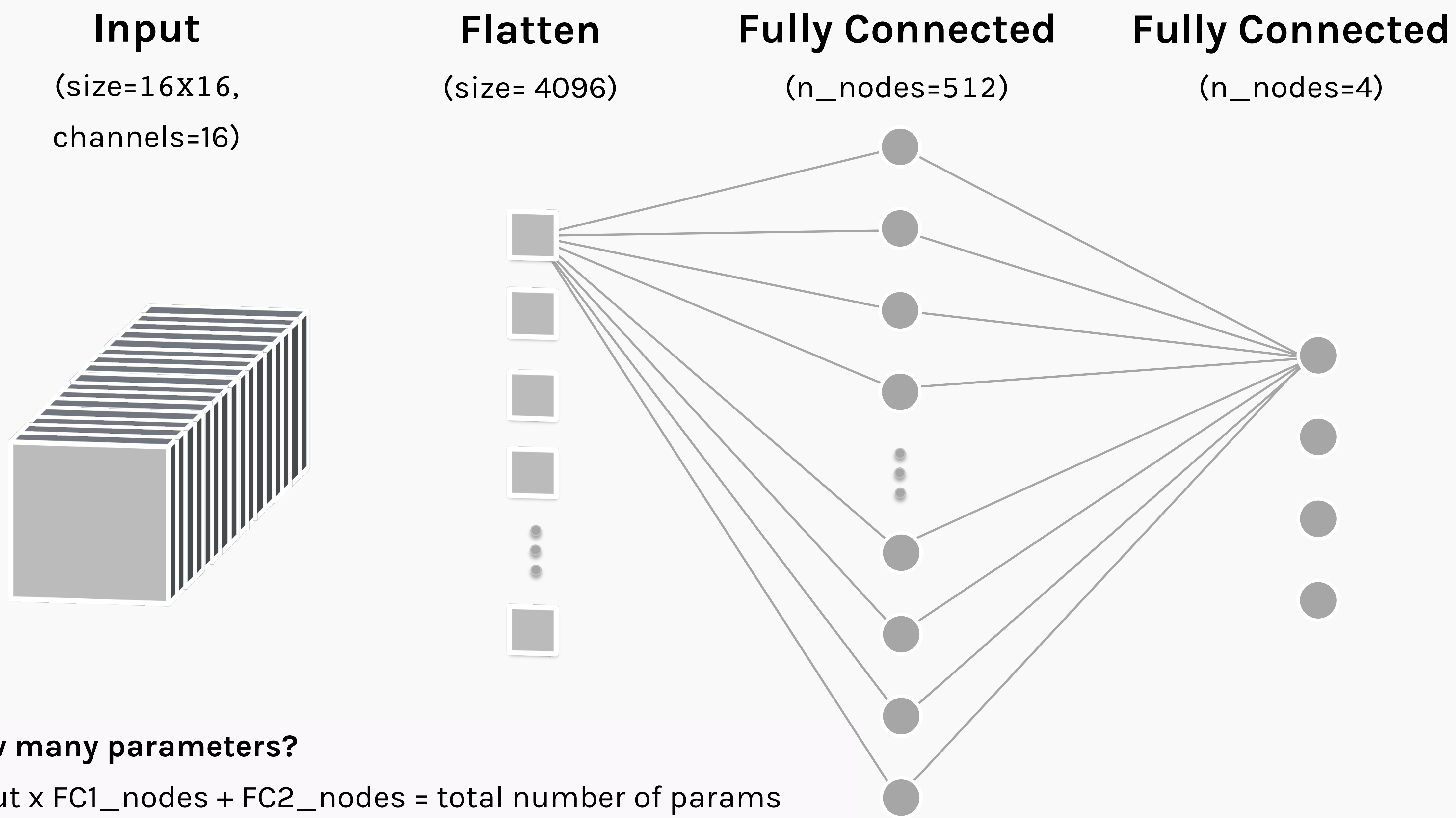
channels=16)



How many parameters does the layer have if I want to use 16 filters?

n_filters x filter_volume + biases = total number of params

16 x (5 x 5 x 8) + 16 = 3216



How many parameters?

input x FC1_nodes + FC2_nodes = total number of params

`(16x16x16) x 512 + 512 + 512 x 4 + 4 = 2,099,716`

Examples

- Let **C** be a CNN with the following disposition:
  - Input: 32x32x3 images
  - Conv1: 8 3x3 filters, stride 1, padding=same
  - Conv2: 16 5x5 filters, stride 2, padding=same
  - Flatten layer
  - Dense1: 512 nodes
  - Dense2: 4 nodes
- How many parameters does this network have?

$$\underbrace{(8 \times 3 \times 3 \times 3 + 8)}_{\text{Conv1}} + \underbrace{(16 \times 5 \times 5 \times 8 + 16)}_{\text{Conv2}} + \underbrace{(16 \times 16 \times 16 \times 512 + 512)}_{\text{Dense1}} + \underbrace{(512 \times 4 + 4)}_{\text{Dense2}}$$

What do CNN layers learn?

- Each CNN layer learns features of increasing complexity.

What do CNN layers learn?

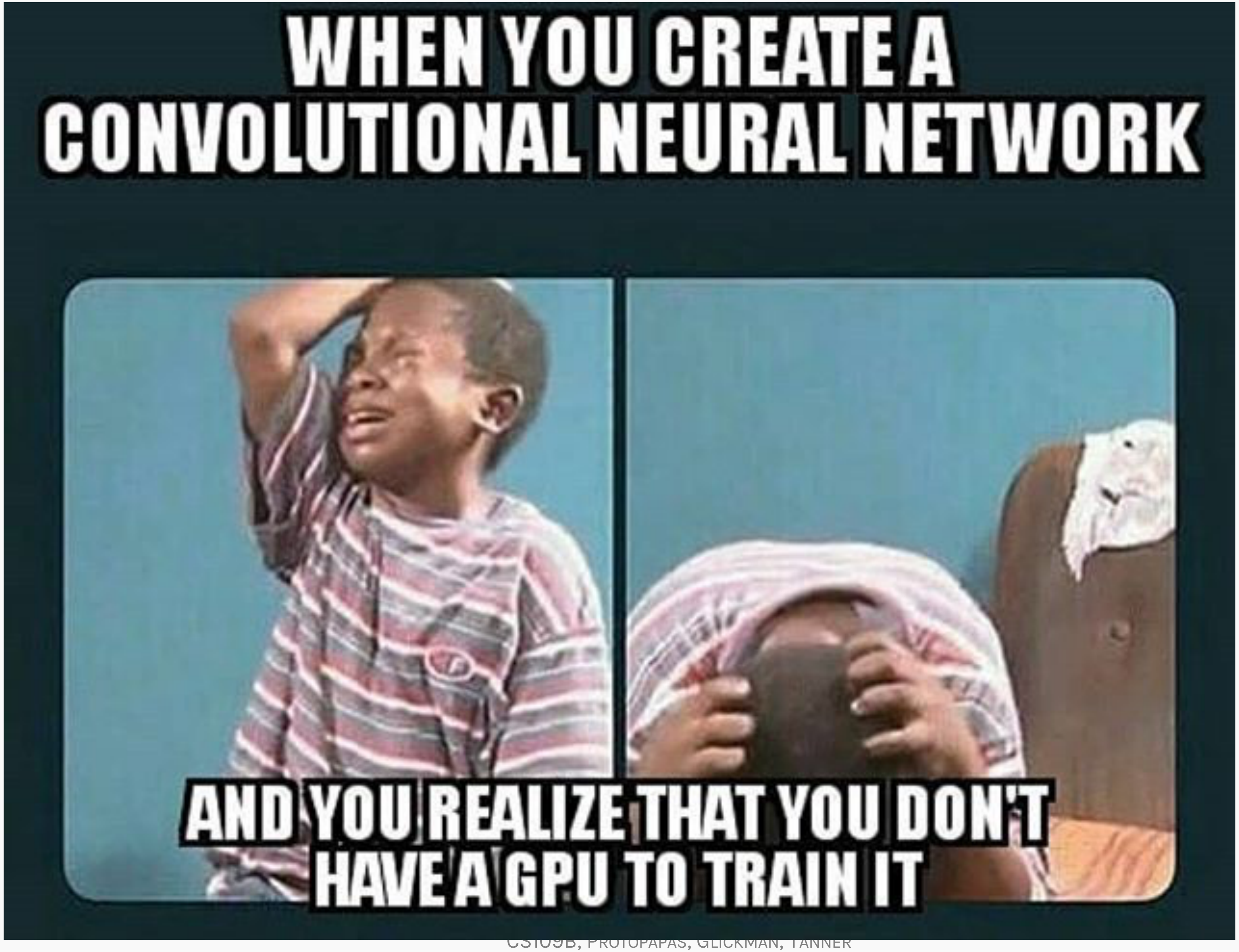
- Each CNN layer learns features of increasing complexity.
- The first layers learn basic feature detection filters: edges, corners, etc.

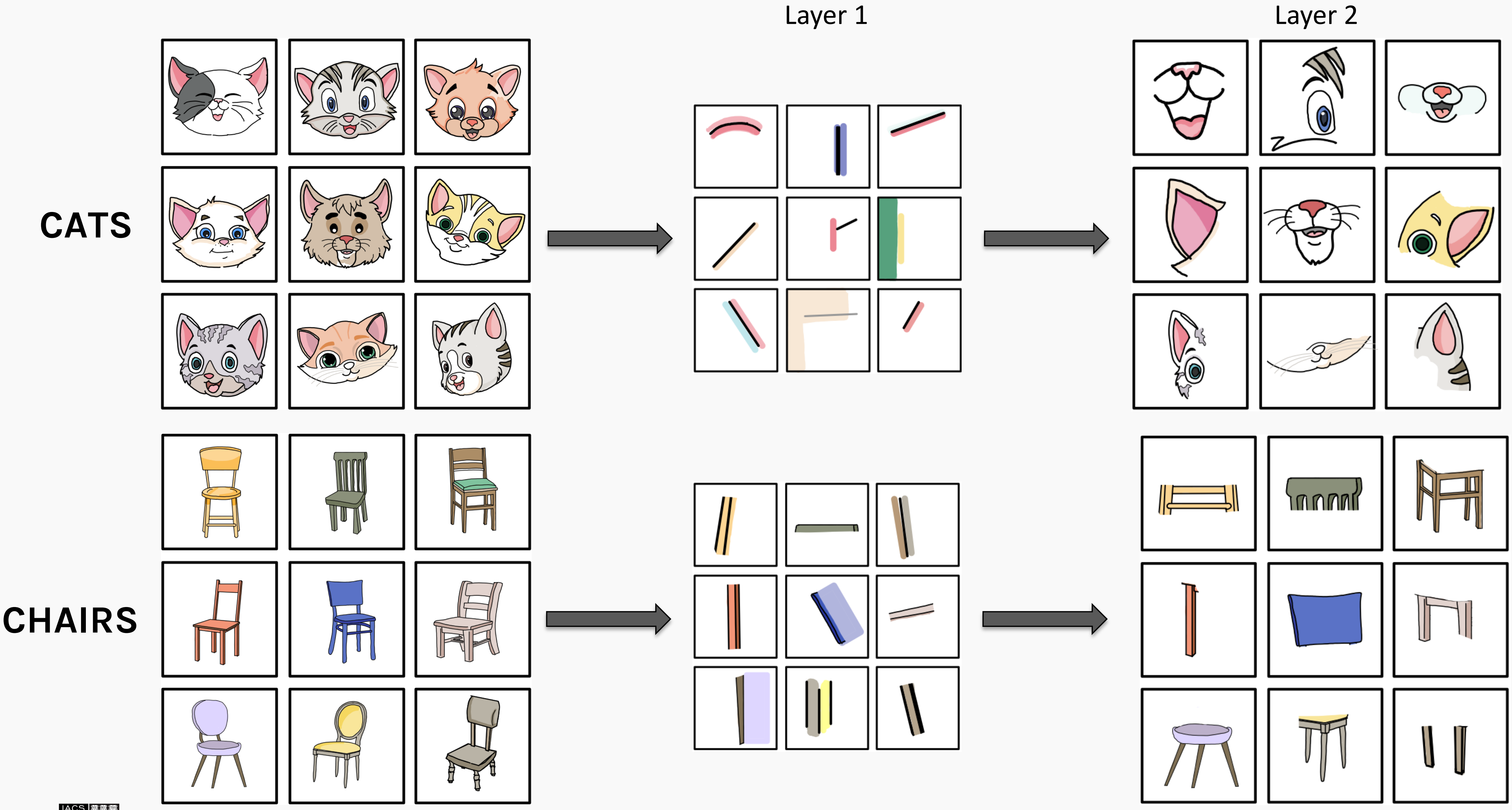
What do CNN layers learn?

- Each CNN layer learns features of increasing complexity.
- The first layers learn basic feature detection filters: edges, corners, etc.
- The middle layers learn filters that detect parts of objects. For faces, they might learn to respond to eyes, noses, etc.

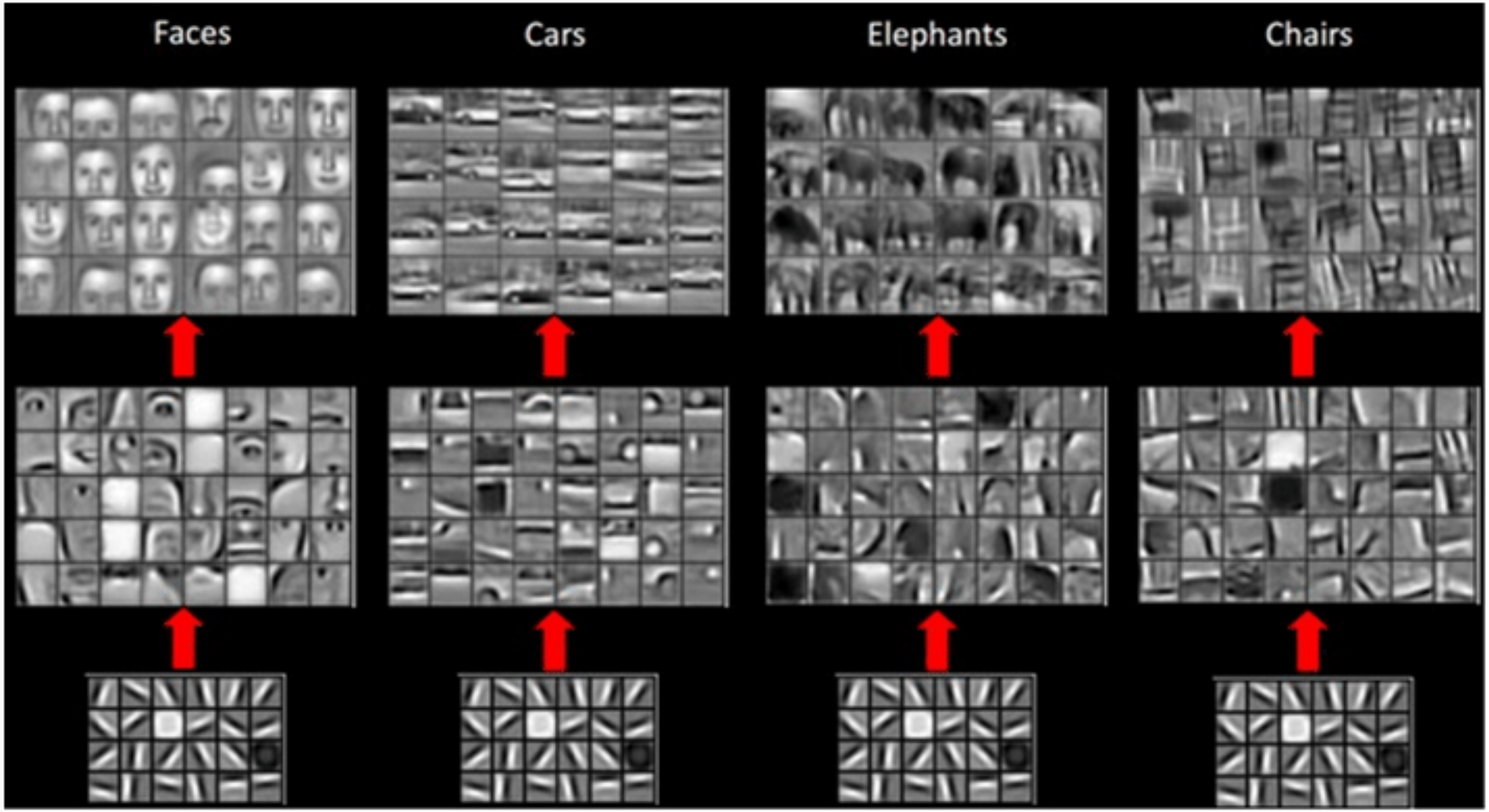
What do CNN layers learn?

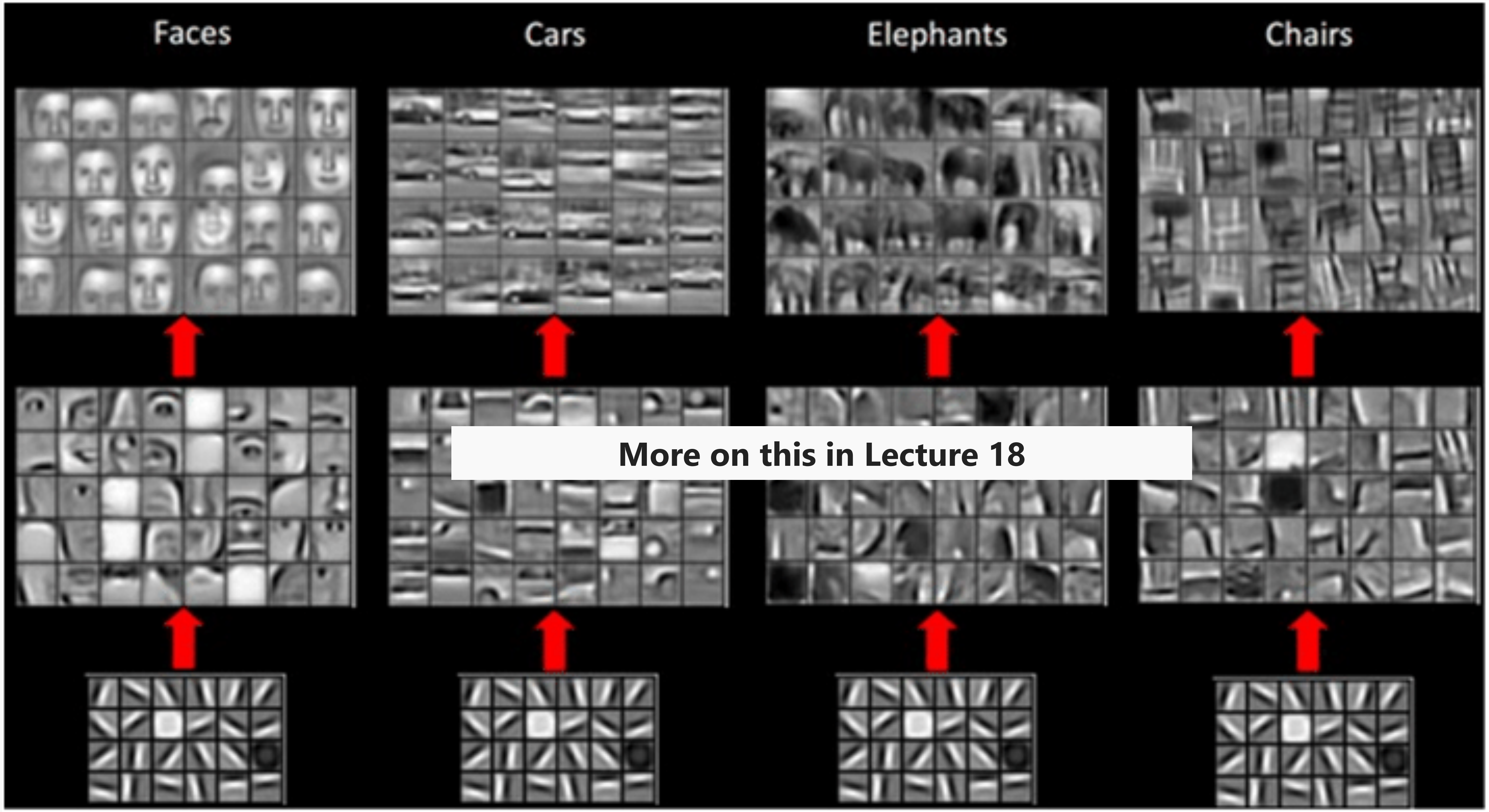
- Each CNN layer learns features of increasing complexity.
- The first layers learn basic feature detection filters: edges, corners, etc.
- The middle layers learn filters that detect parts of objects. For faces, they might learn to respond to eyes, noses, etc.
- The last layers have higher representations: they learn to recognize full objects, in different shapes and positions.


WHEN YOU CREATE A CONVOLUTIONAL NEURAL NETWORK
AND YOU REALIZE THAT YOU DON'T HAVE A GPU TO TRAIN IT



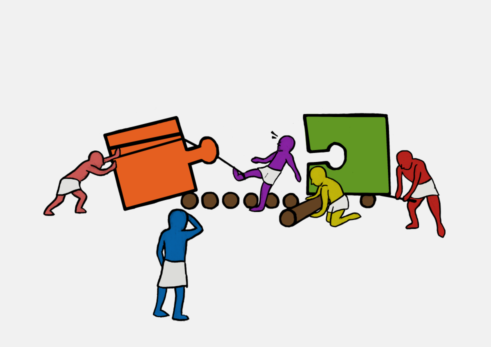
Layer 1
Layer 2
CATS
CHAIRS


Faces
Cars
Elephants
Chairs



More on this in Lecture 18

Exercise: Performance measure



- The aim of this exercise is to compare average and max pooling by measuring accuracy and number of parameters for the classification of MNIST digits

- Build three MNIST classification models, one with no pooling, one with average pooling, and one with max pooling, and train them with similar hyper-parameters

- Compute the number of parameters and the accuracy on the test set for each model

Model Type	Test Accuracy	Test Loss	Number of Parameters
Without pooling	0.8884	0.4061	303314
With avg pooling	0.8996	0.3618	29138
With max pooling	0.9116	0.2936	29138