Machine Learning

Prof. Adil Khan

Today’s Objectives

- A quick recap of the last lecture
- Understanding image convolution and its applications
- Why do we need convolutional neural networks (CNNs)?
- Padding, Stride, Multiple Channels, and Multiple Filters
- Types of Layers in a CNN
- See a simple example of CNN step by step

Recap (1)

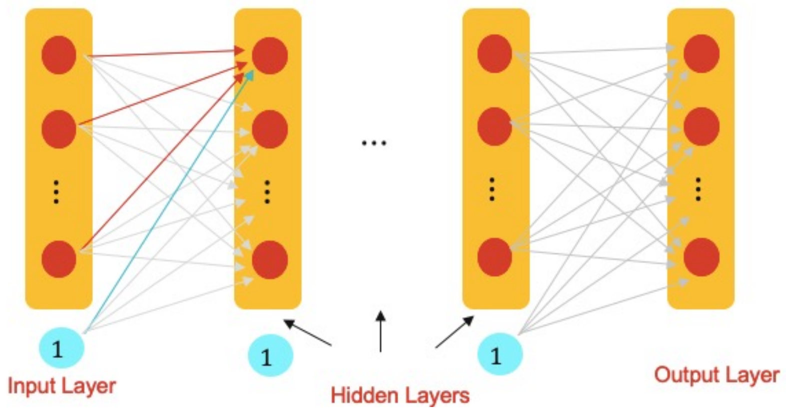
DNNs

1. **Feature learning** is better than using hand-crafted features
2. Feature learning should aim for learning **hierarchical features**
3. Higher level features **should not be simple linear combinations** of low-leve features

Deep Learning enables us to achieve these goals!

Feedforward Networks

$$a_j^l = \sigma\left(\sum_k w_{jk}^l a_k^{l-1} + b_j^l\right)$$



Activation Functions

1. Sigmoid
2. Hyperbolic Tangent
3. Rectified Linear Unit
4. Leaky ReLU
5. Exponential ReLU

Recap (2)

Neural Learning

- Using Gradient Descent find the optimum weights and the biases for the DNN that minimze the error or the cost

Backpropogation

- At the heart of backpropagation is an expression for the partial derivative of the cost function C with respect to every weight w (or bias b) in the network.

$$\boxed{\delta_j^L = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L).}$$

$$\boxed{\frac{\partial C}{\partial b_j^l} = \delta_j^l}$$

$$\boxed{\delta_j^l = \sum_k w_{kj}^{l+1}\delta_k^{l+1}\sigma'(z_j^l).}$$

$$\boxed{\frac{\partial C}{\partial w_{jk}^l} = a_k^{l-1}\delta_j^l.}$$

Recap (3)

Backpropogation Algorithm

1. **Input a set of training examples**

2. **For each training example** x**:** Set the corresponding input activation $a^{x,1}$, and perform the following steps:

   - **Feedforward:** For each $l = 2, 3, \ldots, L$ compute $z^{x,l} = w^l a^{x,l-1} + b^l$ and $a^{x,l} = \sigma(z^{x,l})$.

   - **Output error** $\delta^{x,L}$**:** Compute the vector $\delta^{x,L} = \nabla_a C_x \odot \sigma'(z^{x,L})$.

   - **Backpropagate the error:** For each $l = L-1, L-2, \ldots, 2$ compute $\delta^{x,l} = ((w^{l+1})^T \delta^{x,l+1}) \odot \sigma'(z^{x,l})$.

3. **Gradient descent:** For each $l = L, L-1, \ldots, 2$ update the weights according to the rule $w^l \rightarrow w^l - \frac{\eta}{m} \sum_x \delta^{x,l} (a^{x,l-1})^T$, and the biases according to the rule $b^l \rightarrow b^l - \frac{\eta}{m} \sum_x \delta^{x,l}$.

Image Convolution and Their Applications

Convolutions

- Sounds fancy but it is not!

- Every time we do image blurring, smoothing, sharpening, edge detection, etc. we are doing convolutions

- Convolutions are one of the most critical, fundamental building-blocks in computer vision and image processing.

What is a convolution?

"***In terms of deep learning, an (image) convolution is an element-wise multiplication of two matrices followed by a sum***"

1. Take two matrices (both have the same dimensions).
2. Multiply them, element-by-element (i.e., not the dot product, simple element-to-element multiplication).
3. Sum the elements of the resulting Matrix.

What is a convolution?

$$\begin{bmatrix}1&0&-1\\1&0&-1\\1&0&-1\end{bmatrix} \quad \text{element-to-element multiplication} \quad \begin{bmatrix}3&0&1\\1&5&8\\2&7&2\end{bmatrix} = \begin{bmatrix}3&0&-1\\1&0&-8\\2&0&-2\end{bmatrix}$$

-5 = Sum all Elements

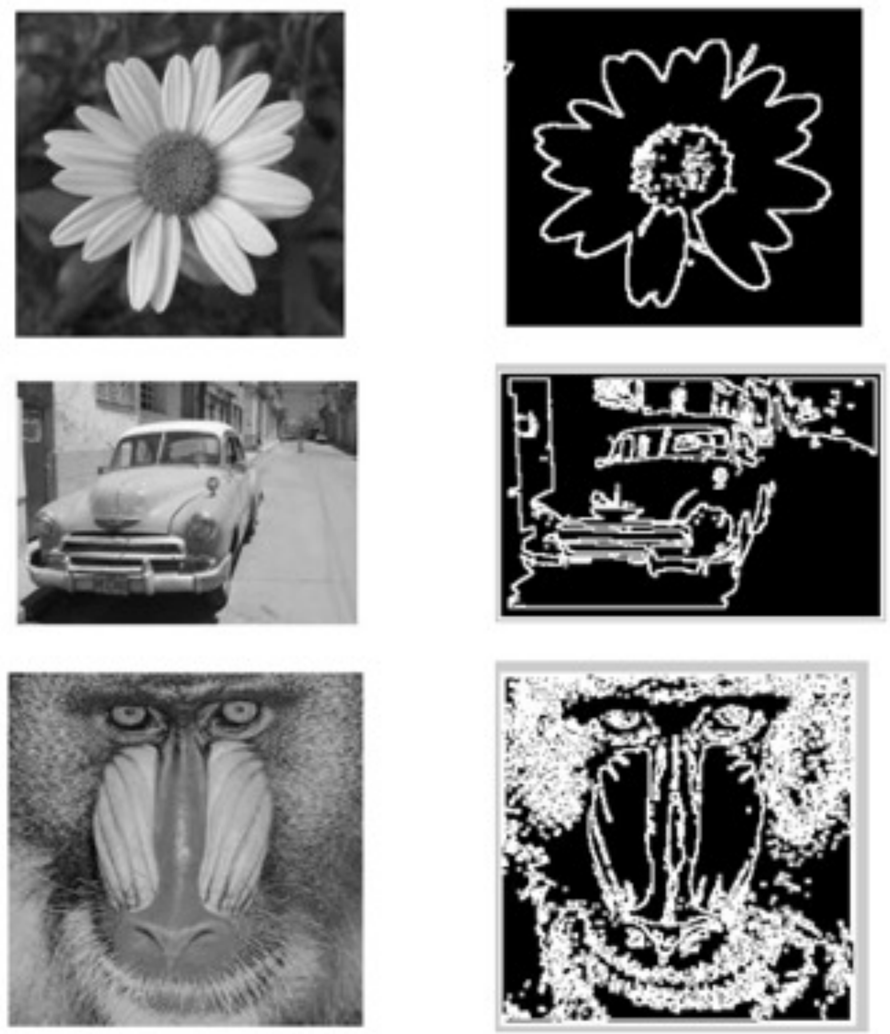
Applications of Convolutions in Images

- Edge Detection
- Image smoothing
- Image sharpening
- Image enhancement

Let’s look at edge detection in more detail.

Edge Detection

- Two types of edges
  - Horizontal Edges
  - Vertical Edges



So how do we detect edges?

- Vertical Edge Detection in a Grey Scale Image

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

6 x 6

- Example of a simple image of resolution 6 x 6
- Each cell contains a grayscale value
- This is what an image looks like to a computer
- We want to detect vertical edges in this image

So how do we detect edges?

- Vertical Edge Detection in a Grey Scale Image

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

6 x 6

*

Convolution

3 x 3

Filter or Kernel

- To do that, we will choose a small matrix called *Kernel*
- And convolve the Kernel with the image

So how do we detect edges?

- Vertical Edge Detection in a Grey Scale Image

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

6 x 6

*

Convolution

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

3 x 3

Filter or Kernel

- How do we select the size of the Kernel and the values in the kernel?

So how do we detect edges?

- Vertical Edge Detection in a Grey Scale Image

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

6 x 6

*

Convolution

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

3 x 3

Filter or Kernel

- Why can this Kernel detect vertical edges?

So how do we detect edges?

- Let's see why the result will be a 4 x 4 matrix.

- Vertical Edge Detection in a Grey Scale Image

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

6 x 6

*

Convolution

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

3 x 3

Filter or Kernel

=

4 x 4

So how do we detect edges?

- Vertical Edge Detection in a Grey Scale Image

3^{1}	0^{0}	1^{-1}	2	7	4
1^{1}	5^{0}	8^{-1}	9	3	1
2^{1}	7^{0}	2^{-1}	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6 x 6

*

Convolution

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

3 x 3

Filter or Kernel

=

4 x 4

So how do we detect edges?

- Vertical Edge Detection in a Grey Scale Image

3^{1}	0^{0}	1^{-1}	2	7	4
1^{1}	5^{0}	8^{-1}	9	3	1
2^{1}	7^{0}	2^{-1}	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6 x 6

*

Convolution

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

3 x 3

Filter or Kernel

=

-5			

4 x 4

Next we roll the window forward on step!

So how do we detect edges?

- Vertical Edge Detection in a Grey Scale Image

3	0 [1]	1 [0]	2 [-1]	7	4
1	5 [1]	8 [0]	9 [-1]	3	1
2	7 [1]	2 [0]	5 [-1]	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6 x 6

$*$

Convolution

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

3 x 3

Filter or Kernel

=

-5			

4 x 4

Calculate the result of convolution!

So how do we detect edges?

- Vertical Edge Detection in a Grey Scale Image

3	0^{1}	1^{0}	2^{-1}	7	4
1	5^{1}	8^{0}	9^{-1}	3	1
2	7^{1}	2^{0}	5^{-1}	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6 x 6

*

Convolution

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

3 x 3

Filter or Kernel

=

-5	-4		

4 x 4

So how do we detect edges?

- Vertical Edge Detection in a Grey Scale Image

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

6 x 6

*

Convolution

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

3 x 3

Filter or Kernel

=

-5	-4	0	8

4 x 4

We have computed the entire first row, what now?

So how do we detect edges?

- Vertical Edge Detection in a Grey Scale Image

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

6 x 6

*

Convolution

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

3 x 3

Filter or Kernel

=

-5	-4	0	8
-10			

4 x 4

We move the window down and start from left again

So how do we detect edges?

- Vertical Edge Detection in a Grey Scale Image

- Dimensions of output depend on: (1) Size of Kernel, (2) Step size

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

6 x 6

*

Convolution

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

3 x 3

Filter or Kernel

=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4 x 4

Kernels for Vertical and Horizontal Edge Detection

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

Vertical

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

Horizontal

Edges as *Features*

- Edges represent the boundary of an object in an image
- We can use them to identify the objects: face, car, street signs, etc.
- Thus, edges can be thought of as features, or predictors.

More Generally

- Kernels (or filters) and Convolution can help us find features in a given input.

- Thus Kernels (or filters) can be thought of as ***feature detectors***

- Idea-1: *we can learn which features are important for a given task, and learn kernels for detecting those features from the data instead of hand-designing them!*

Learning Kernels or Filters

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

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

- Filter is represented by the parameters that we want to learn
- Learning happens under a loss function
- In other words, we learn those filters that helps us discover features that improve classification

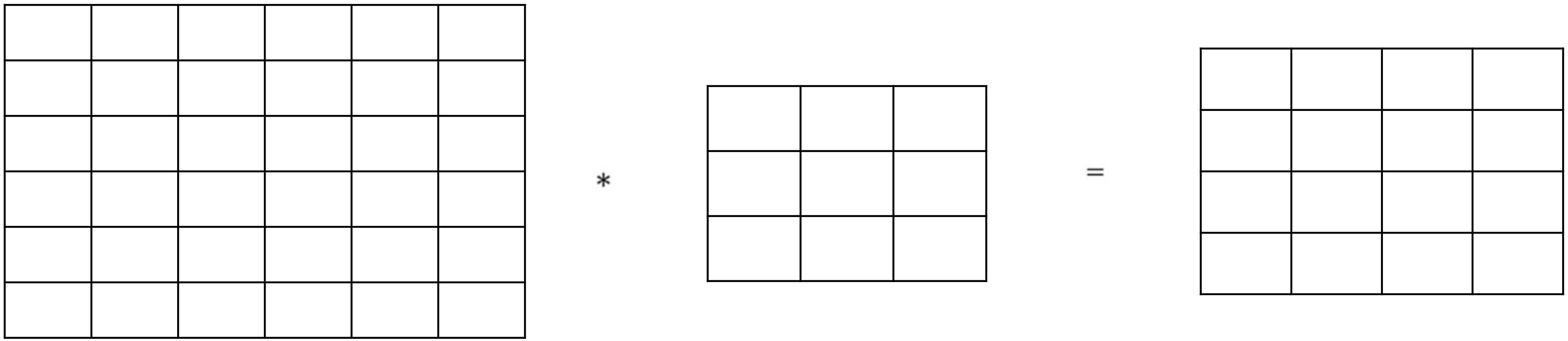
Hierarchical Features

- Higher-level features can be built from lower-level features
- For example, edges can be combined to detect noses, eyes, and lips
- Nose, eyes, and lips can be combined to detect faces
- Etc.
- Idea-2: *we can learn kernels from the data in a hierarchical fashion*

And this brings us to CNNs.

Padding, Stride, Multiple Channels, and Multiple Filters

Dimensions in Convolution

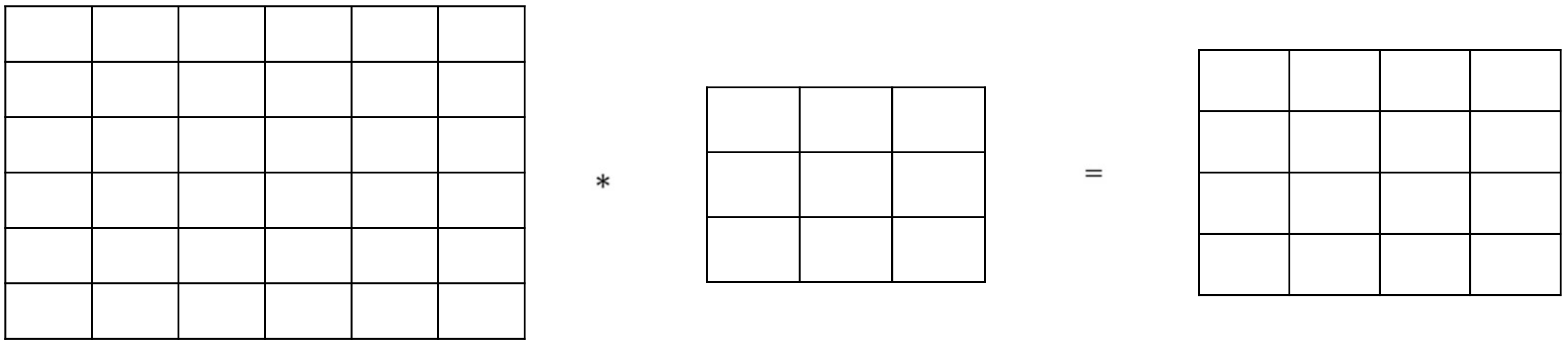


3 x 3

4 x 4

6 x 6

In General



6 x 6 | 3 x 3 | 4 x 4

$n \times n$ | $f \times f$ | $(n - f + 1) \times (n - f + 1)$

f is usually odd!

Problem

1. Thus, the size decreases (shrinking output)

2. As well as, the pixels on the edges are used less than those in the center of the image

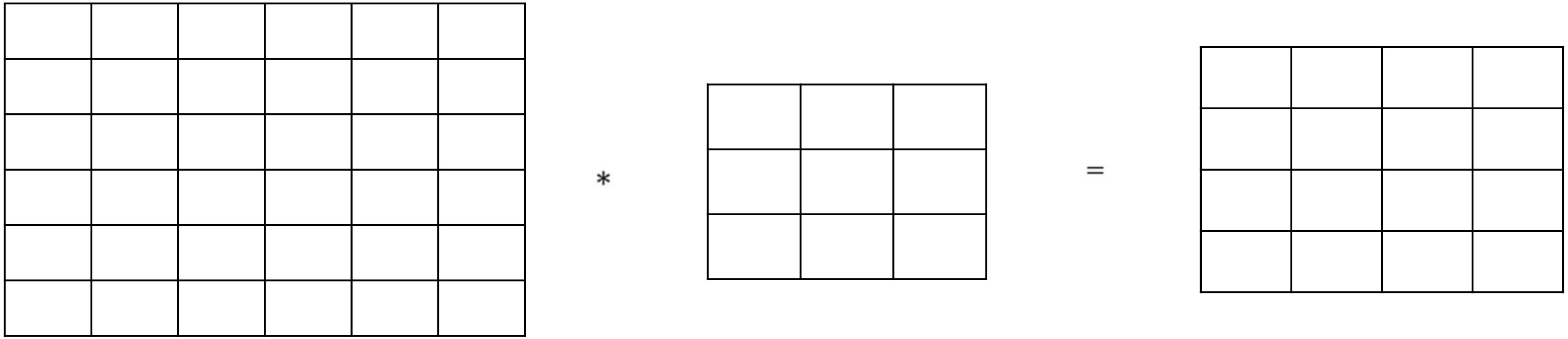
- And in order to fix these problems, we can do padding (pad the image with additional border of pixels)

Padding

95	242	186	152	39
39	14	220	153	180
5	247	212	54	46
46	77	133	110	74
156	35	74	93	116

0	0	0	0	0	0	0
0	95	242	186	152	39	0
0	39	14	220	153	180	0
0	5	247	212	54	46	0
0	46	77	133	110	74	0
0	156	35	74	93	116	0
0	0	0	0	0	0	0

Original Image is 6 x 6, Output Image is 4 x 4



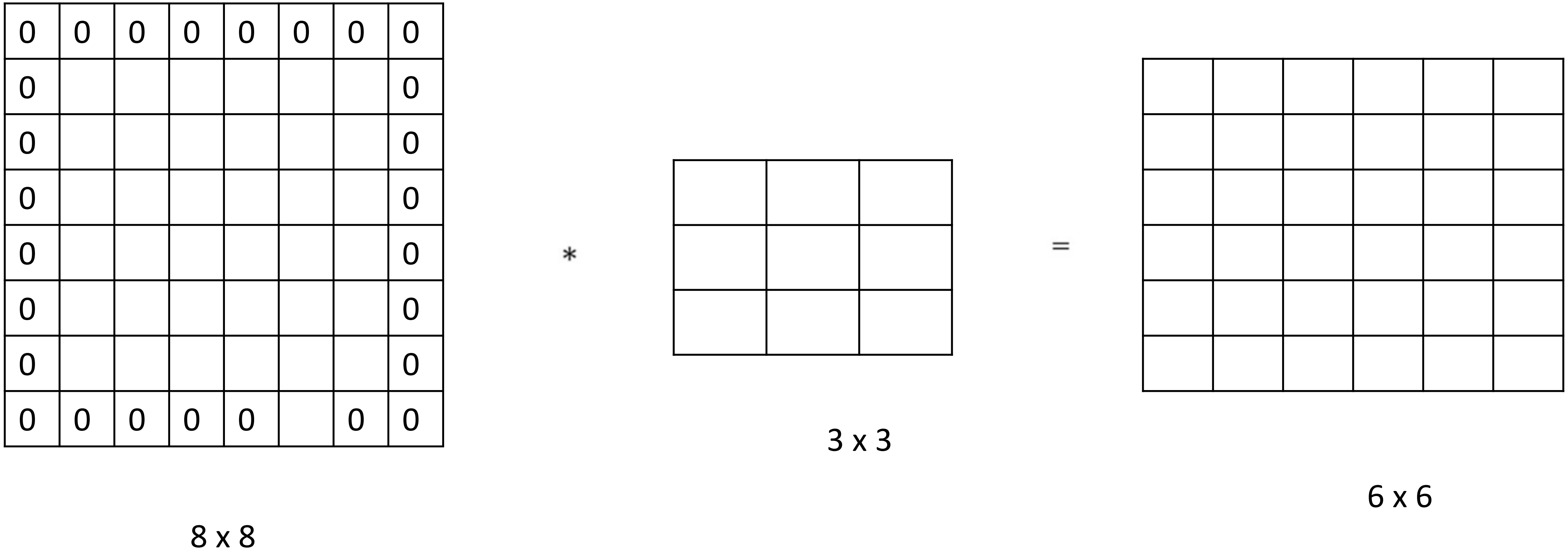
6 x 6

3 x 3

4 x 4

If we want the output image to be 6 x 6 then we have to do the padding.

New Dimensions After Padding



Padding (Dimensions)

- Input Size: $n \times n$
- Filter Size: $f \times f$
- Padding: p
- Output Size: $(n + 2p - f + 1) \times (n + 2p - f + 1)$

Valid and Same Convolutions

- Define whether convolution is with or without padding
- Valid: when convolution is done without padding.
- Same: when padding is done to keep the output size the same as input size

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

Stride

95	242	186	152	39
39	14	220	153	180
5	247	212	54	46
46	77	133	110	74
156	35	74	93	116

*

0	1	0
1	-4	1
0	1	0

=

692	-315	-6
-680	-194	305
153	-59	-86

When moving filter across the input, we are stopping at each coordinate

What if we did a stride of 2?

Stride

95	242	186	152	39
39	14	220	153	180
5	247	212	54	46
46	77	133	110	74
156	35	74	93	116

*

0	1	0
1	-4	1
0	1	0

=

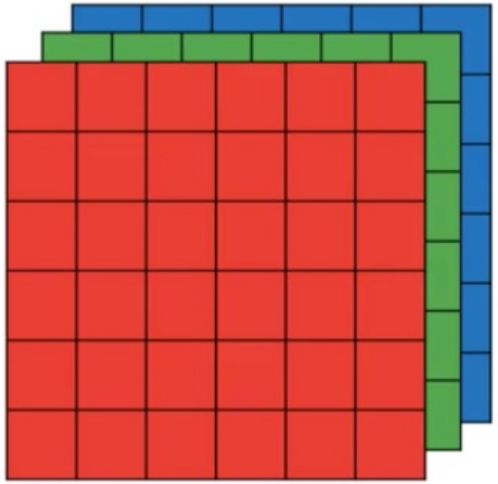
692	-6
153	-86

With a Stride of 2

Dimensions with Stride

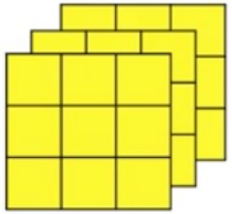
- Input Size: n x n
- Filter Size: f x f
- Padding: p
- Stride: S
- Output Size: $\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$ x $\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$

Convolutions on RGB Image



6 x 6 x 3

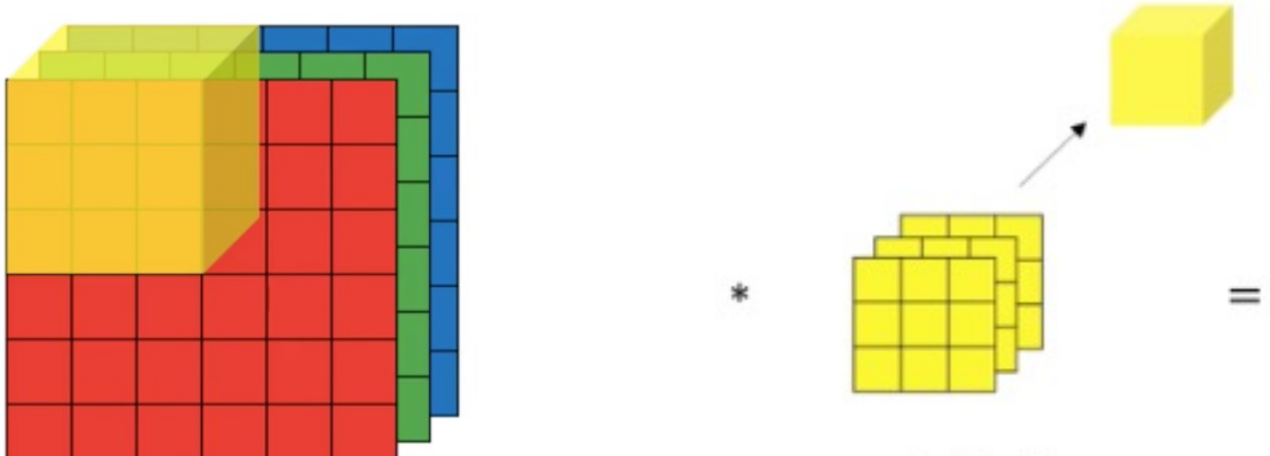
*



3 x 3 x 3

=

Convolutions on RGB Image

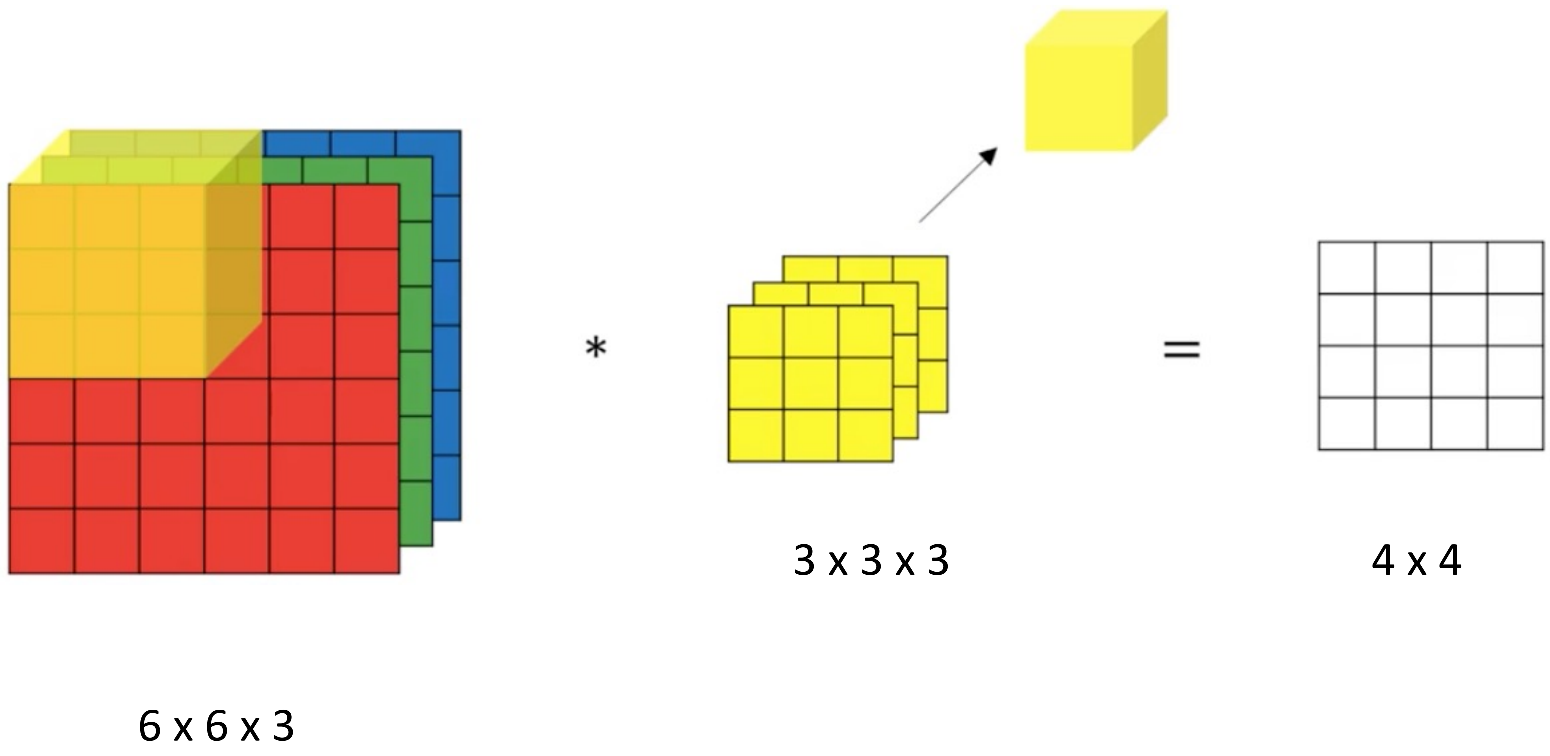


3 x 3 x 3

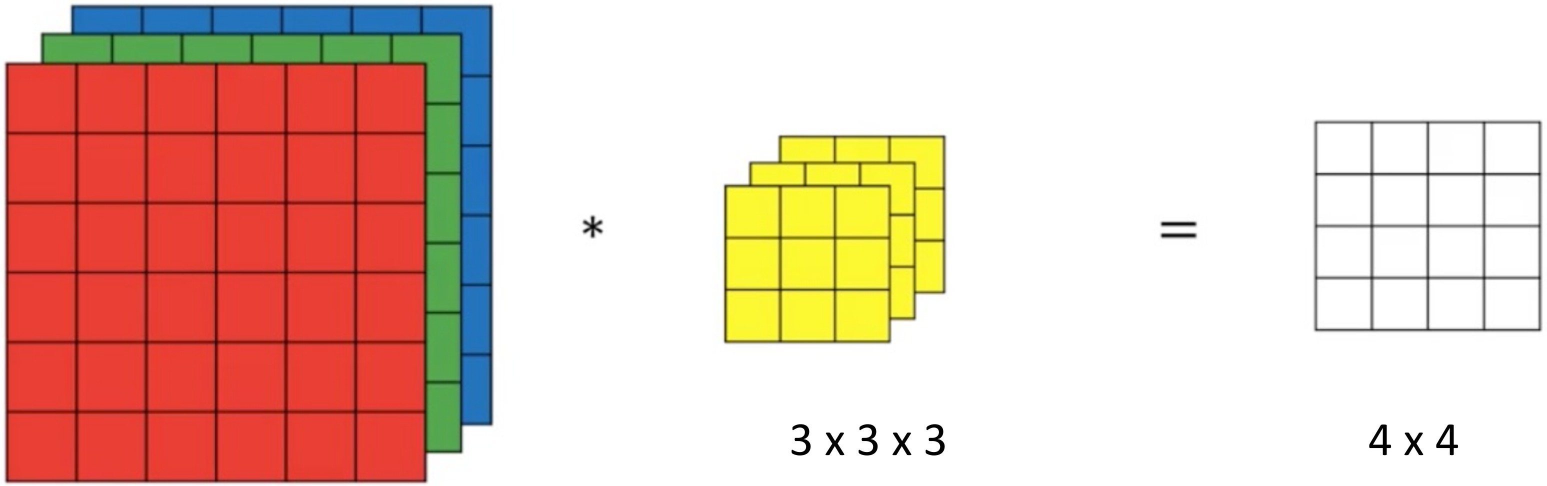
6 x 6 x 3

You do $f \times f \times ch$ multiplication, and then add the result of all multiplications

Convolutions on RGB Image



Convolutions on RGB Image



3 x 3 x 3

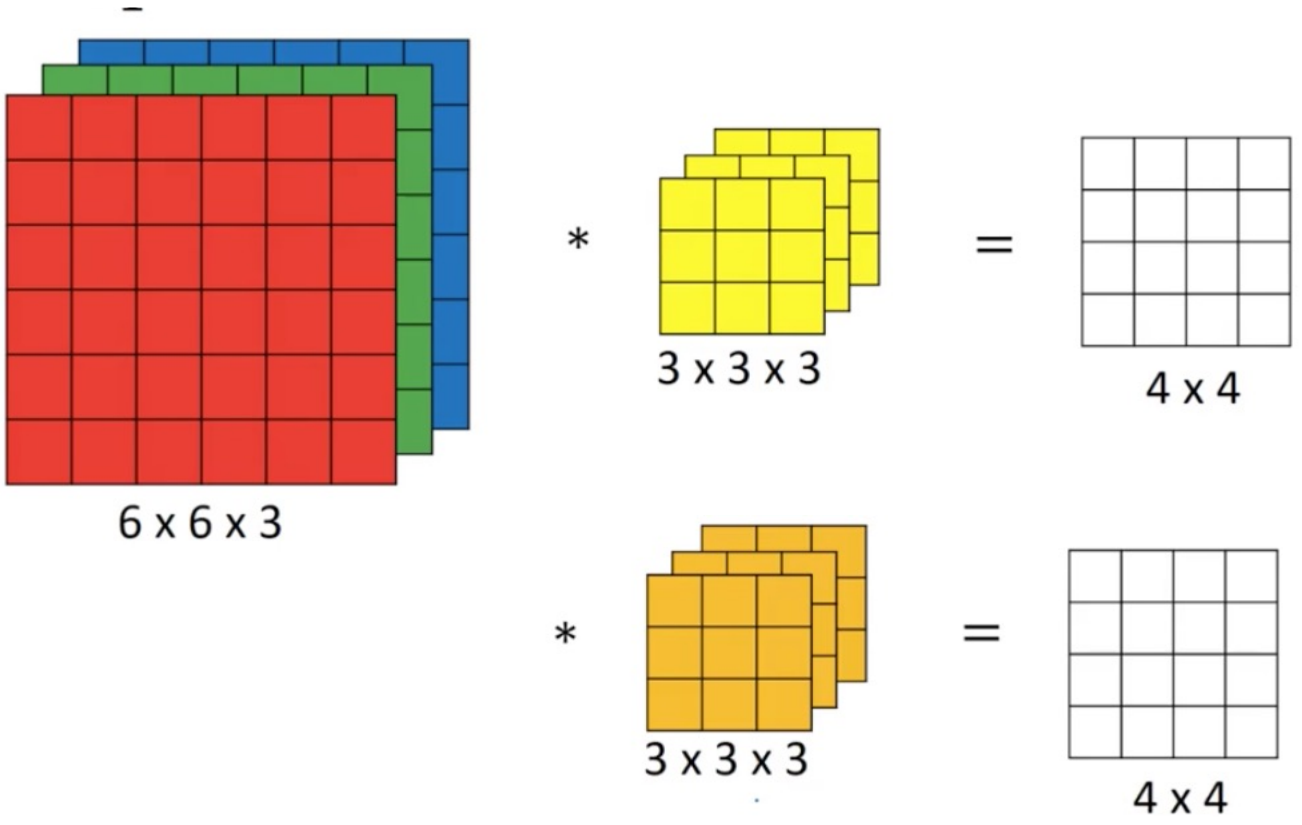
4 x 4

6 x 6 x 3

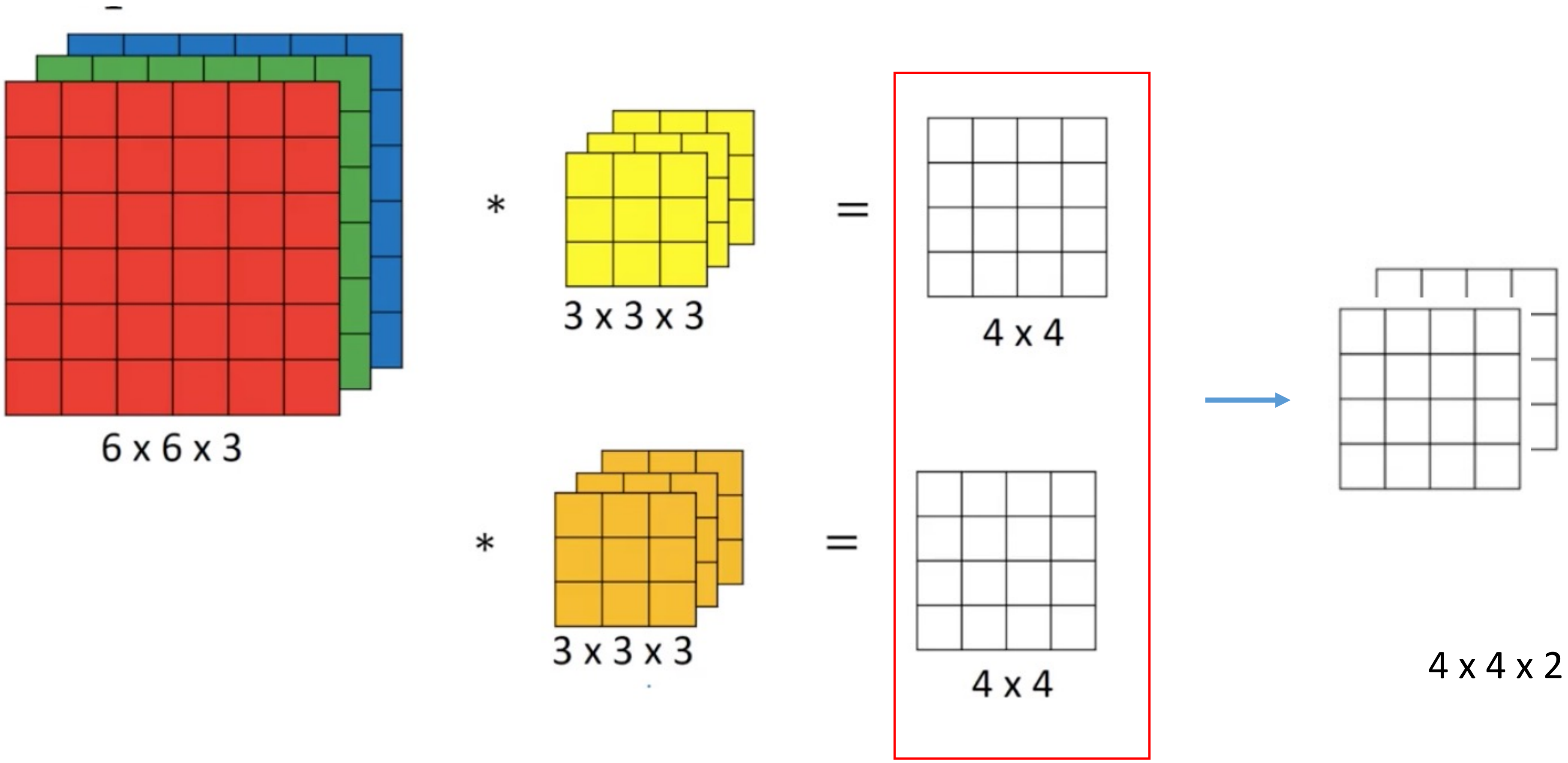
Three dimensional input, three dimensional kernel!

But a 2 dimensional output!

Multiple Filters

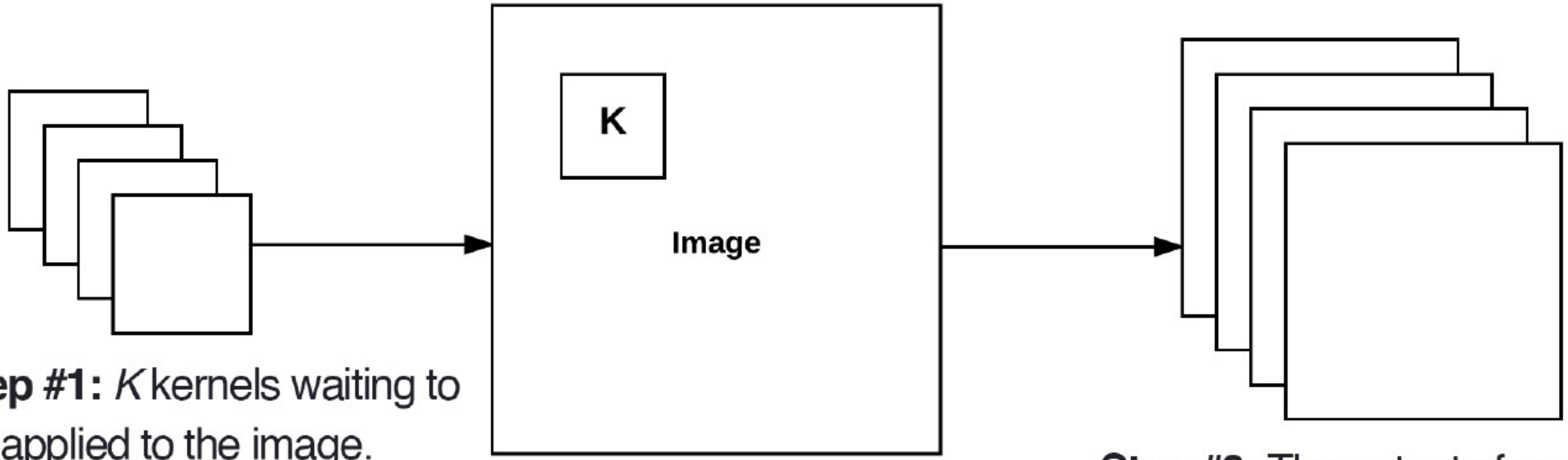


Multiple Filters



Activation Maps – combined as a volume

In General

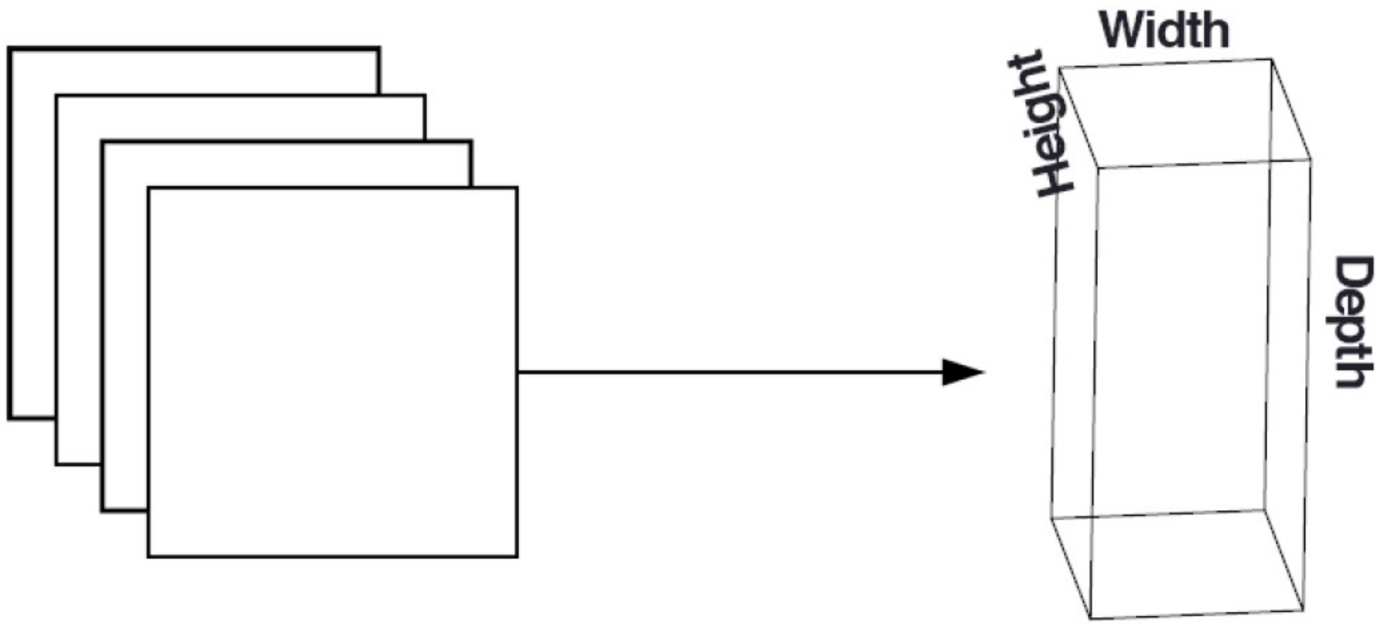


Step #1: *K* kernels waiting to be applied to the image.

Step #2: Each kernel is convolved with the input volume.

Step #3: The output of each convolution operation produces a 2D output, called an “activation map”.

Activation Maps



After obtaining the K activation maps, they are stacked together to form the input volume to the next layer in the network

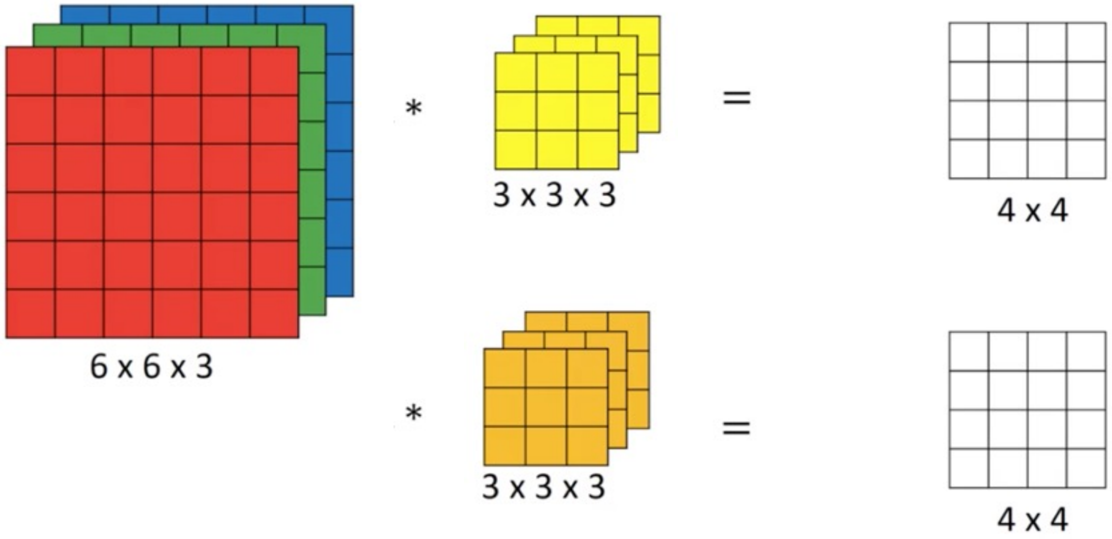
Different types of Layers in a CNN

Types of Layers in a Convolutional Net:

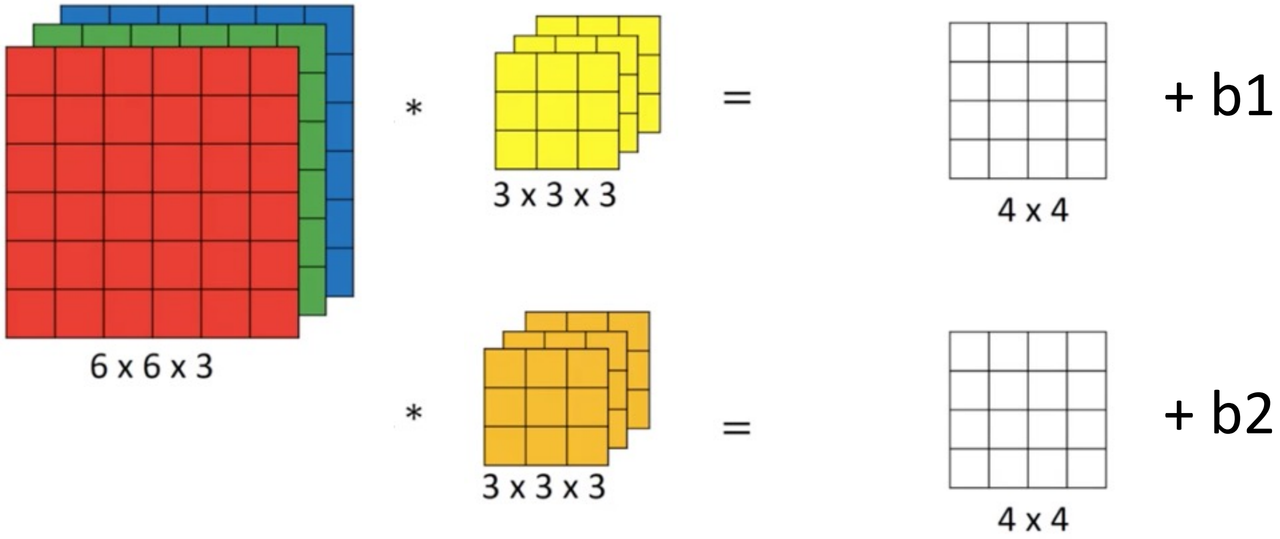
- Convolutional Layers (CONV)
- Pooling Layers (POOL)
- Fully connected (FC)

What Happens in a CONV layer?

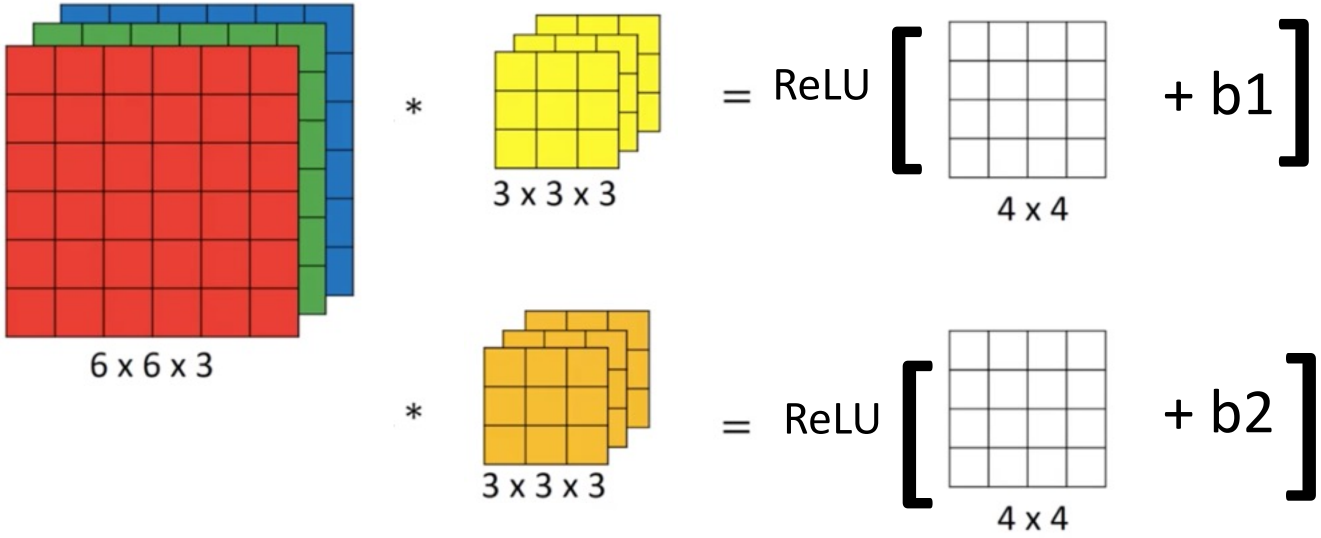
Conv Layer of a CNN



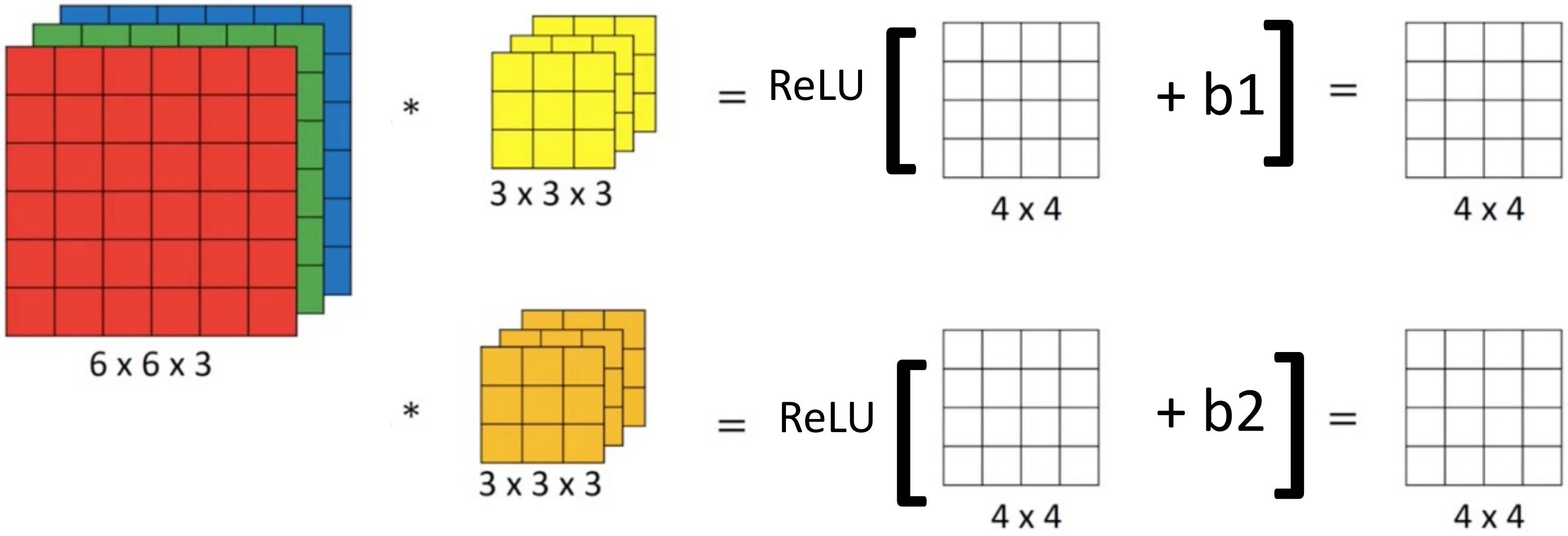
Conv Layer of a CNN



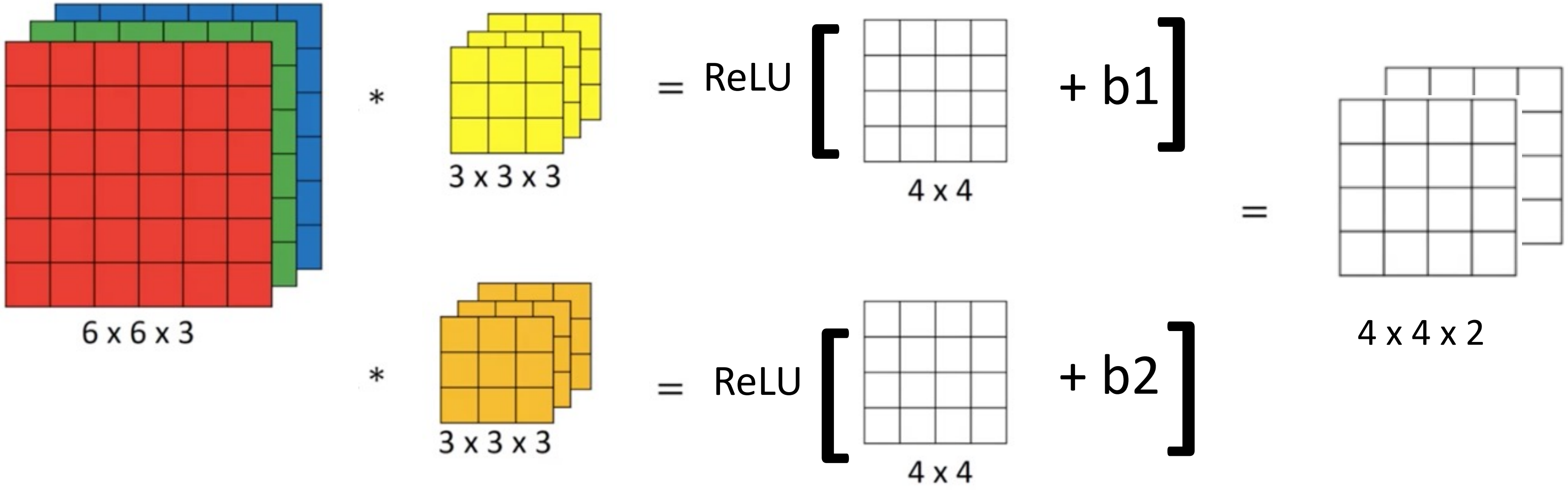
Conv Layer of a CNN



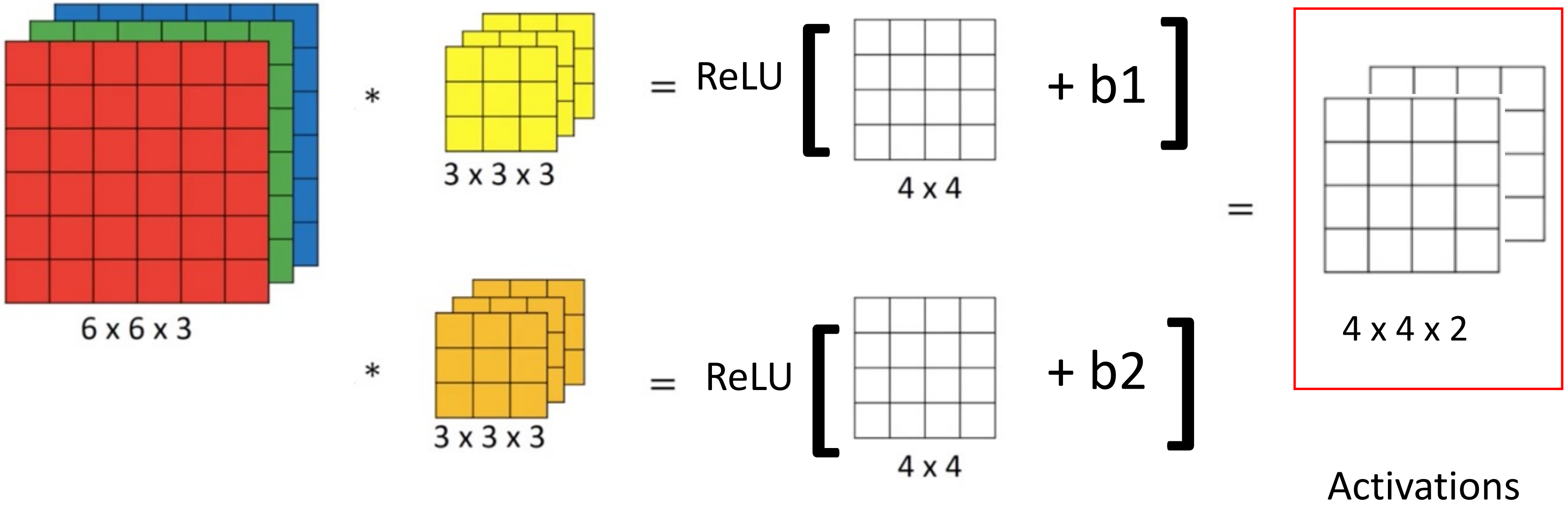
Conv Layer of a CNN



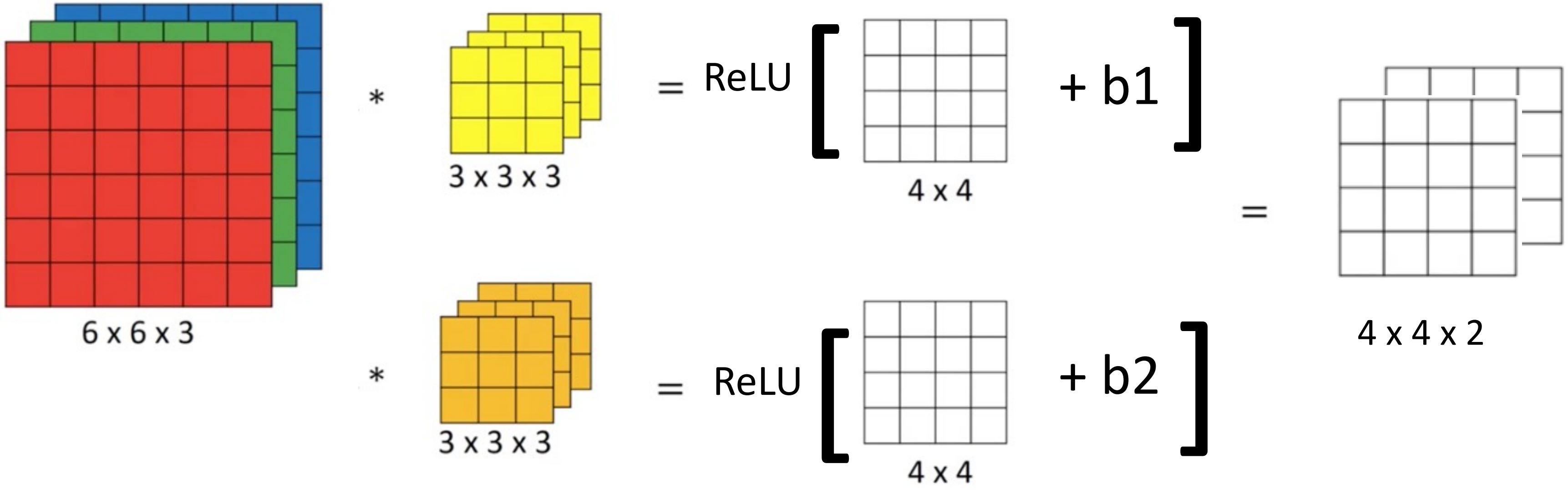
Conv Layer of a CNN



Conv Layer of a CNN



Conv Layer of a CNN



ReLU is just for example, you can apply any other activation function

Number of Parameters in a Conv Layer

- Let’s say that in a layer:
  - You have 10 filters
  - Each filter is of size 3 x 3 x 3
  - How many parameters does this layer have?

Number of Parameters in a Conv Layer

- Let's say that in a layer:

  - You have 10 filters
  - Each filter is of size 3 x 3 x 3
  - How many parameters does this layer have?

  - For each filter we have 27 + 1 parameters (1 for bias)
  - Total number of parameters for 10 filters = 280

What Happens in a POOLING layer?

Pooling

- In convolutional neural nets, the input size decreases as we move deeper in to the network

- It happens due to filter size and stride

- Further reduction in dimensions can be achieved via pooling

Max Pooling Filter

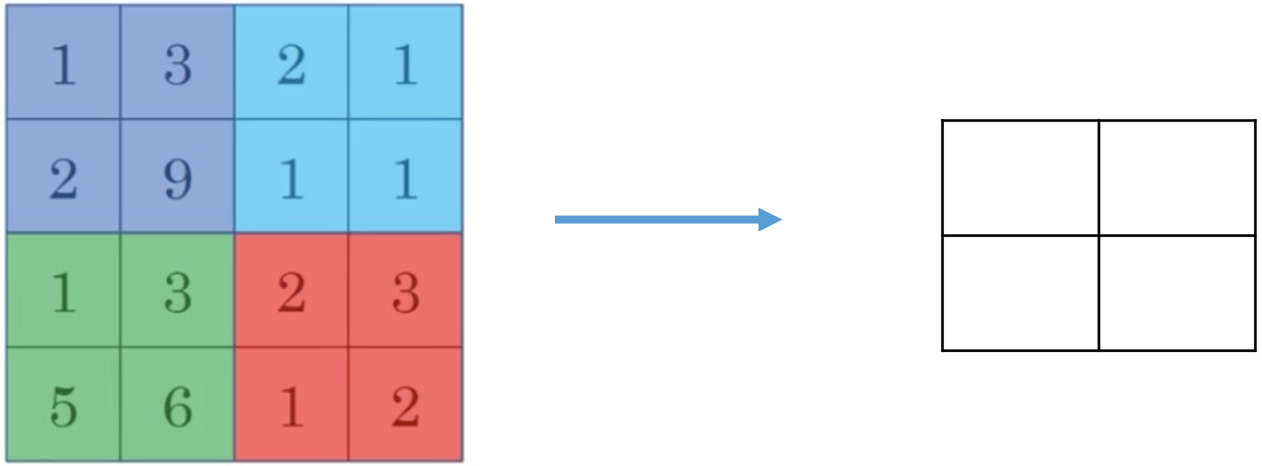
- Process a region of size f x f and reduce it to single value: maximum value in that region

Pooling Layer: Max Pooling

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2

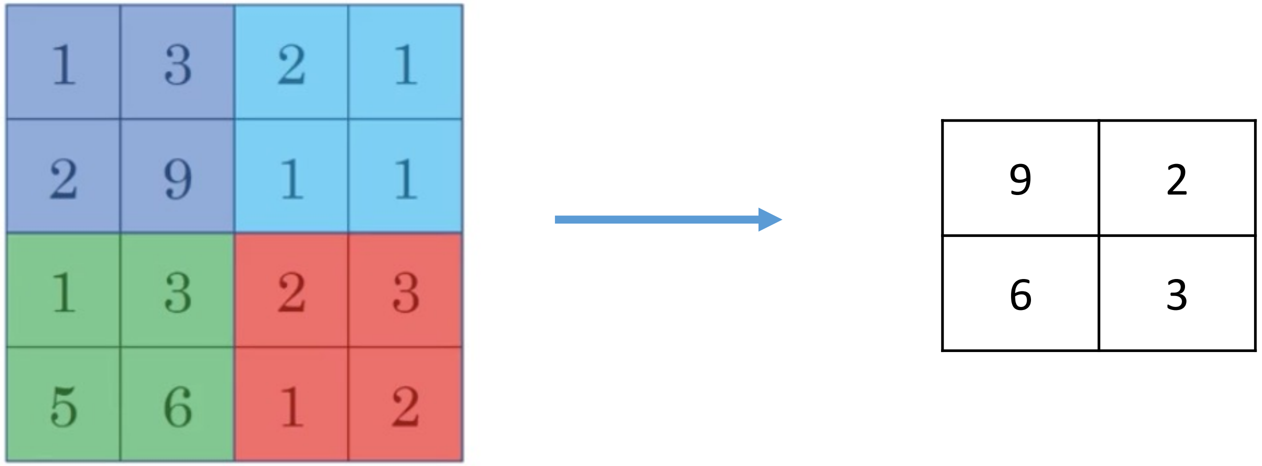
Let' say this is the input of 4 x 4, and we want to reduce it to 2 x 2

Pooling Layer: Max Pooling



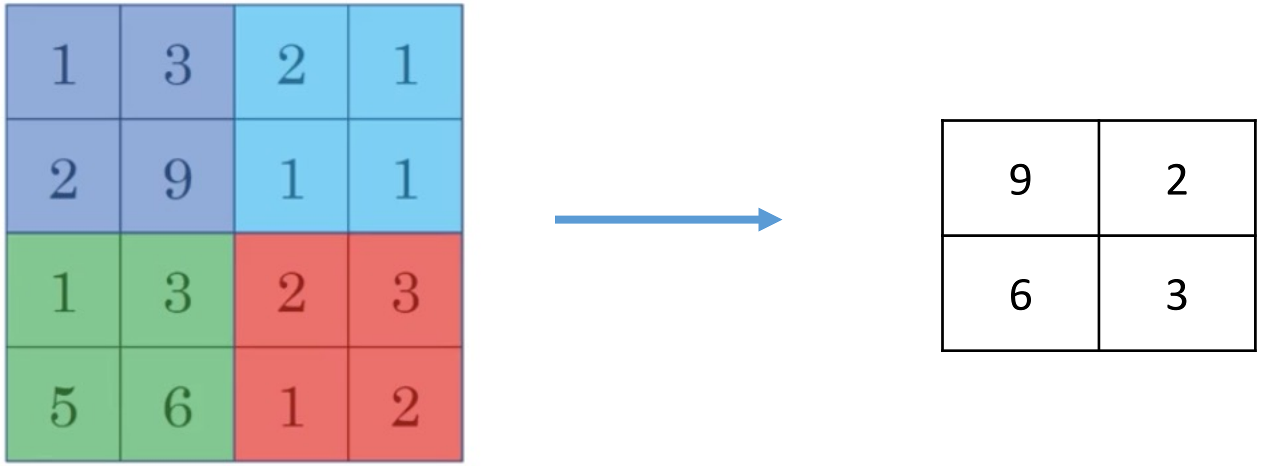
We can achieve this by applying a filter of size 2 x 2 with a stride of 2

Pooling Layer: Max Pooling



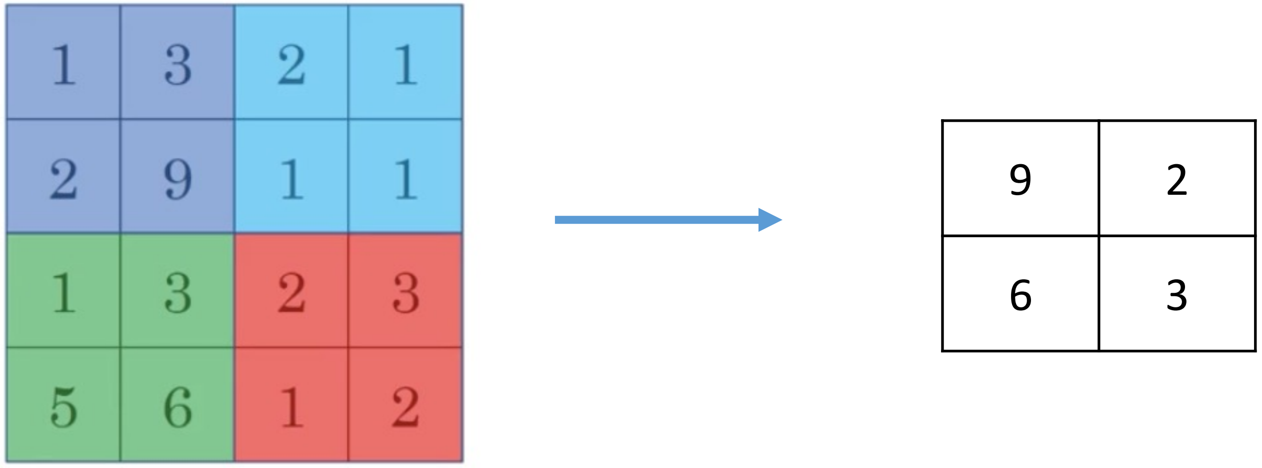
- Filter size and stride length are the hyerparameters of the pooling layer
- There are no parameters to learn

Pooling Layer: Max Pooling



- Max pooling can be thought of as a filter that works on a feature map
- Looking for a particular feature
- Preserving the strength of finding that feature in a region

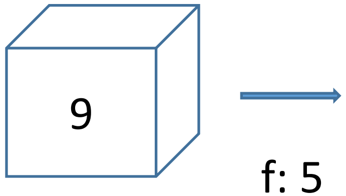
Pooling Layer: Max Pooling



- Finally, you do pooling on each channel in the input separately

Finally, Let’s look at a Complete CNN

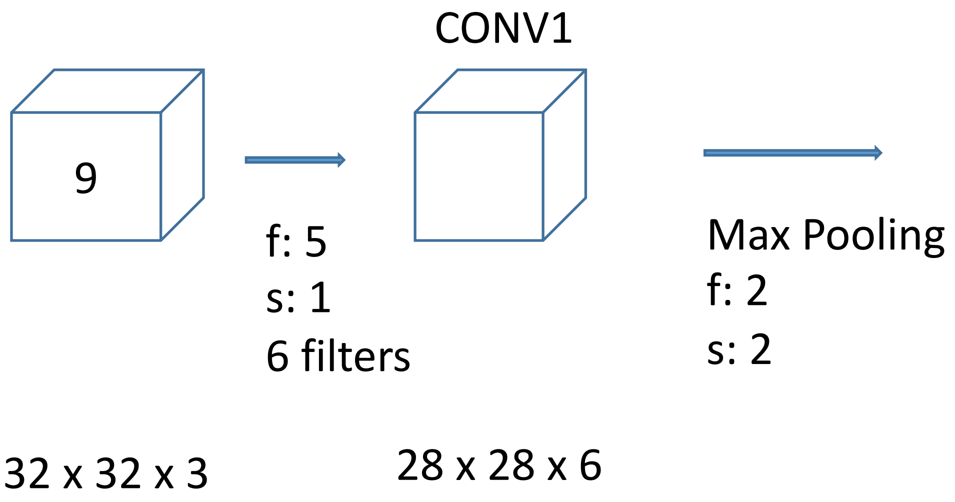
Example: *Digit Classification via CNN*



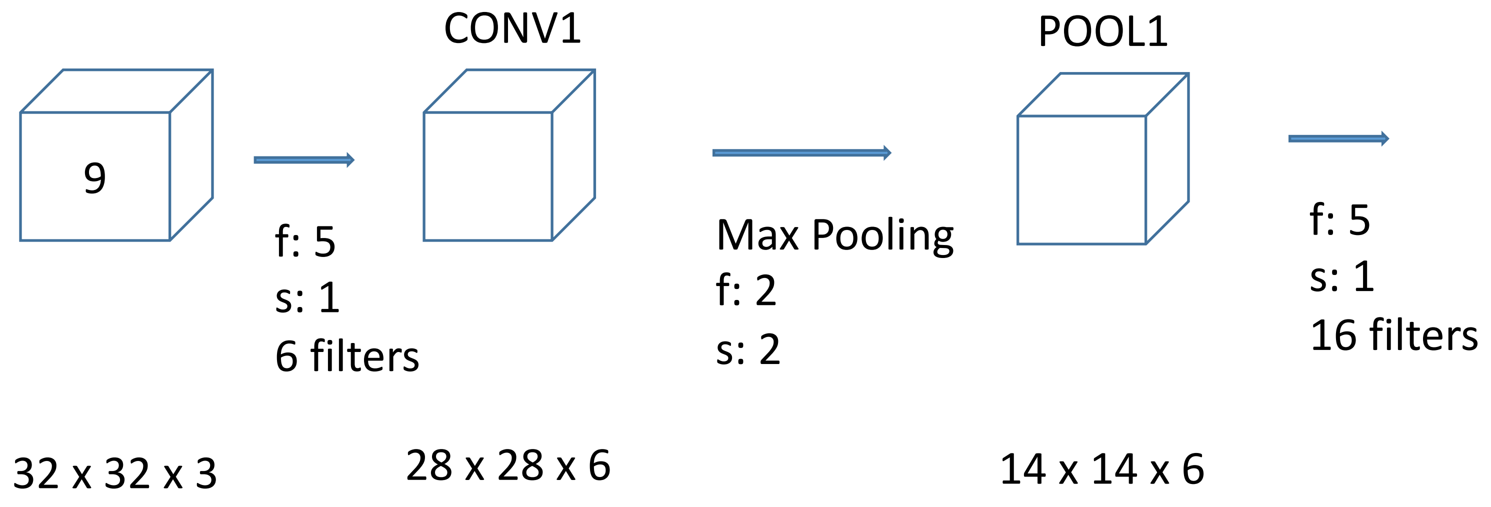
f: 5
s: 1
6 filters

32 x 32 x 3

Example

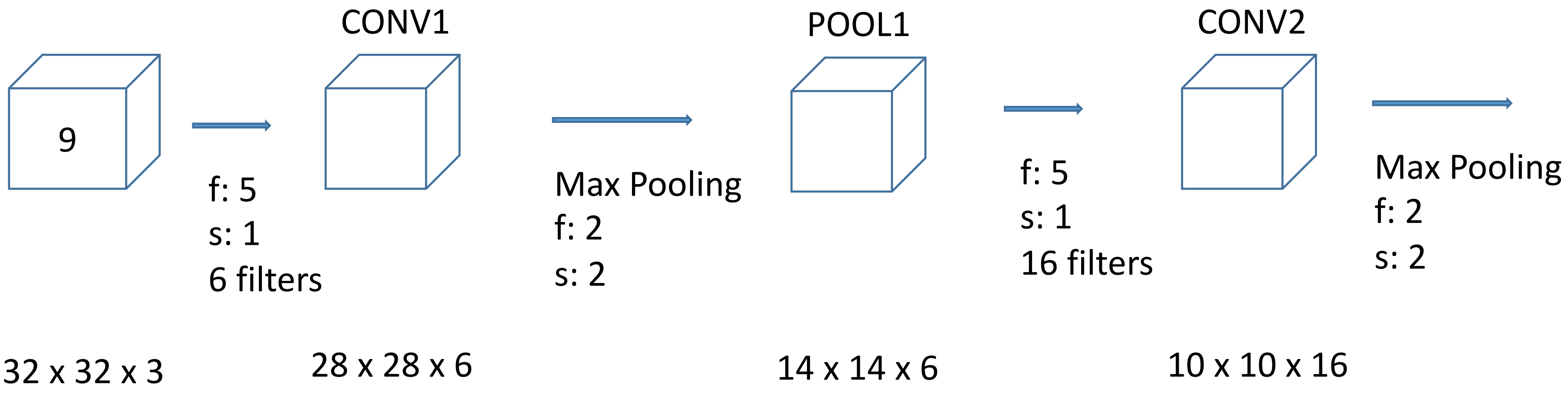


Example

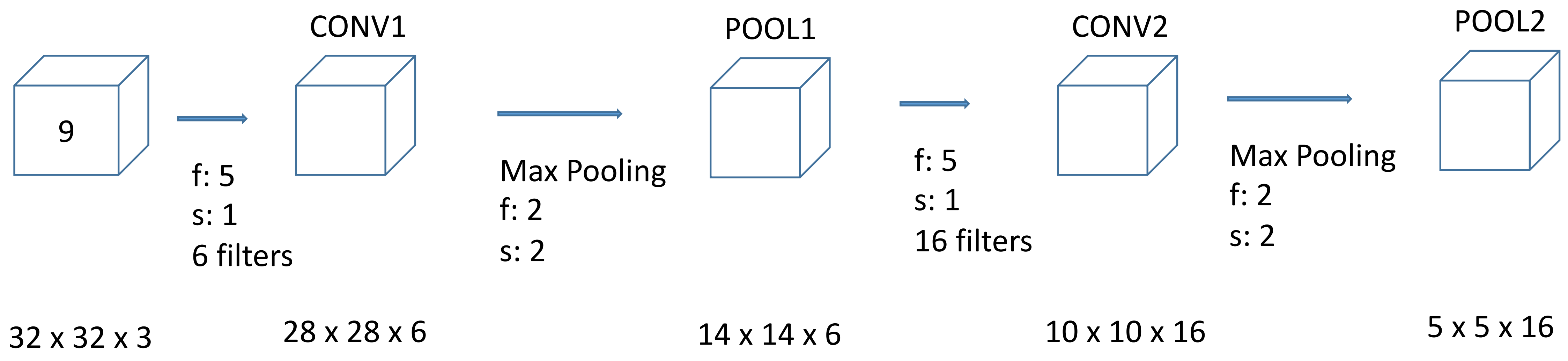


CONV1
POOL1
9
f: 5
s: 1
6 filters
Max Pooling
f: 2
s: 2
f: 5
s: 1
16 filters
32 x 32 x 3
28 x 28 x 6
14 x 14 x 6

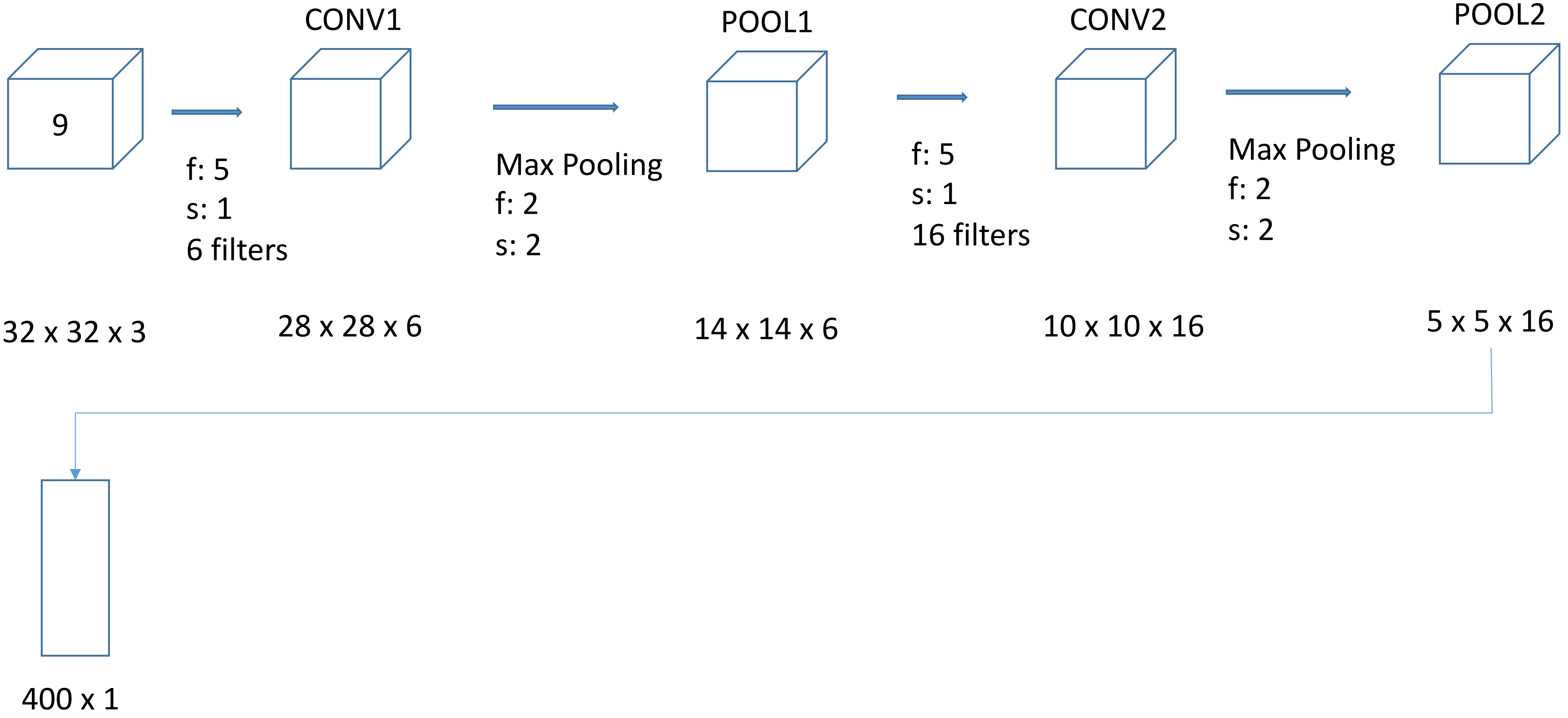
Example


CONV1
POOL1
CONV2
9
f: 5
s: 1
6 filters
Max Pooling
f: 2
s: 2
f: 5
s: 1
16 filters
Max Pooling
f: 2
s: 2
32 x 32 x 3
28 x 28 x 6
14 x 14 x 6
10 x 10 x 16

Example

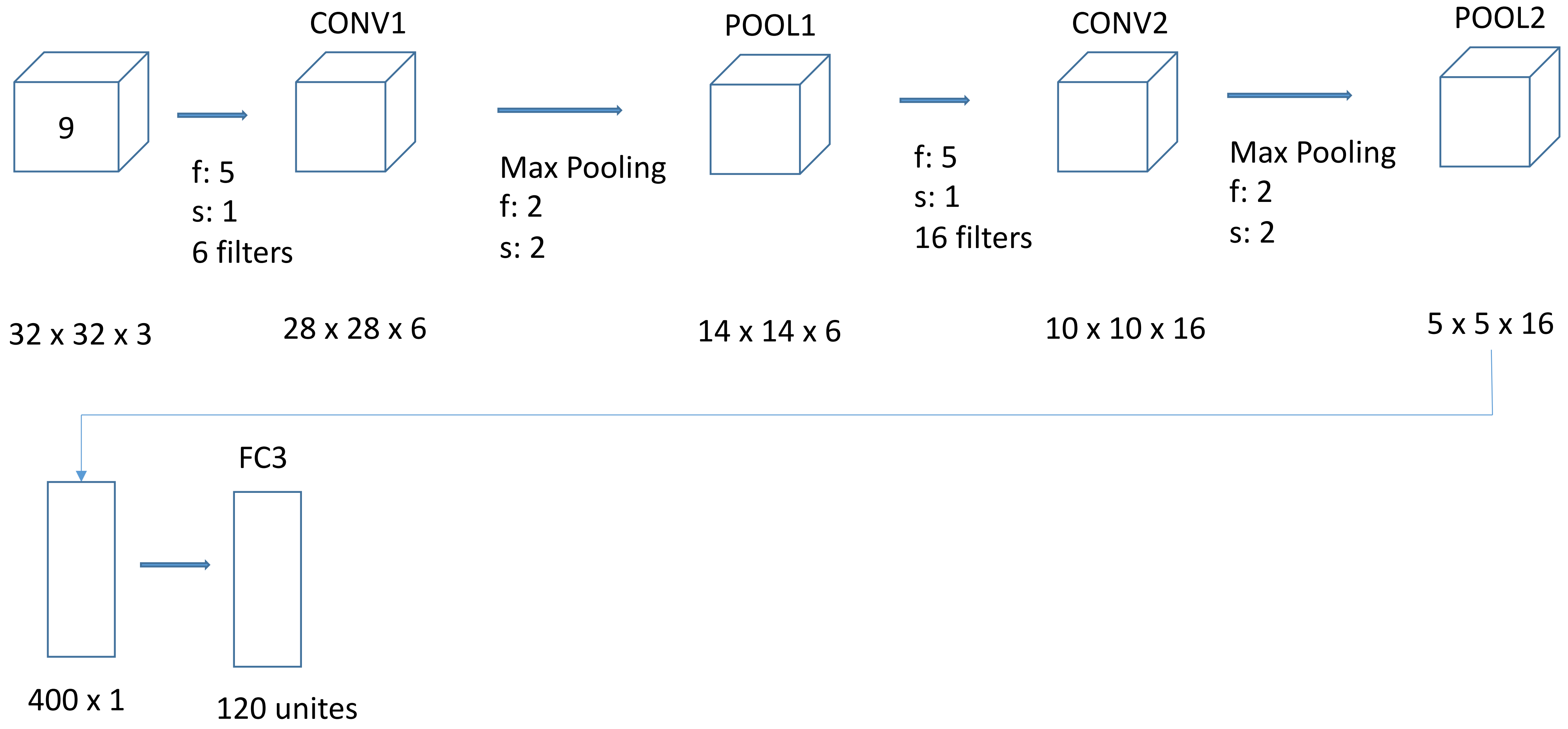

CONV1
POOL1
CONV2
POOL2
9
f: 5
s: 1
6 filters
Max Pooling
f: 2
s: 2
f: 5
s: 1
16 filters
Max Pooling
f: 2
s: 2
32 x 32 x 3
28 x 28 x 6
14 x 14 x 6
10 x 10 x 16
5 x 5 x 16

Example



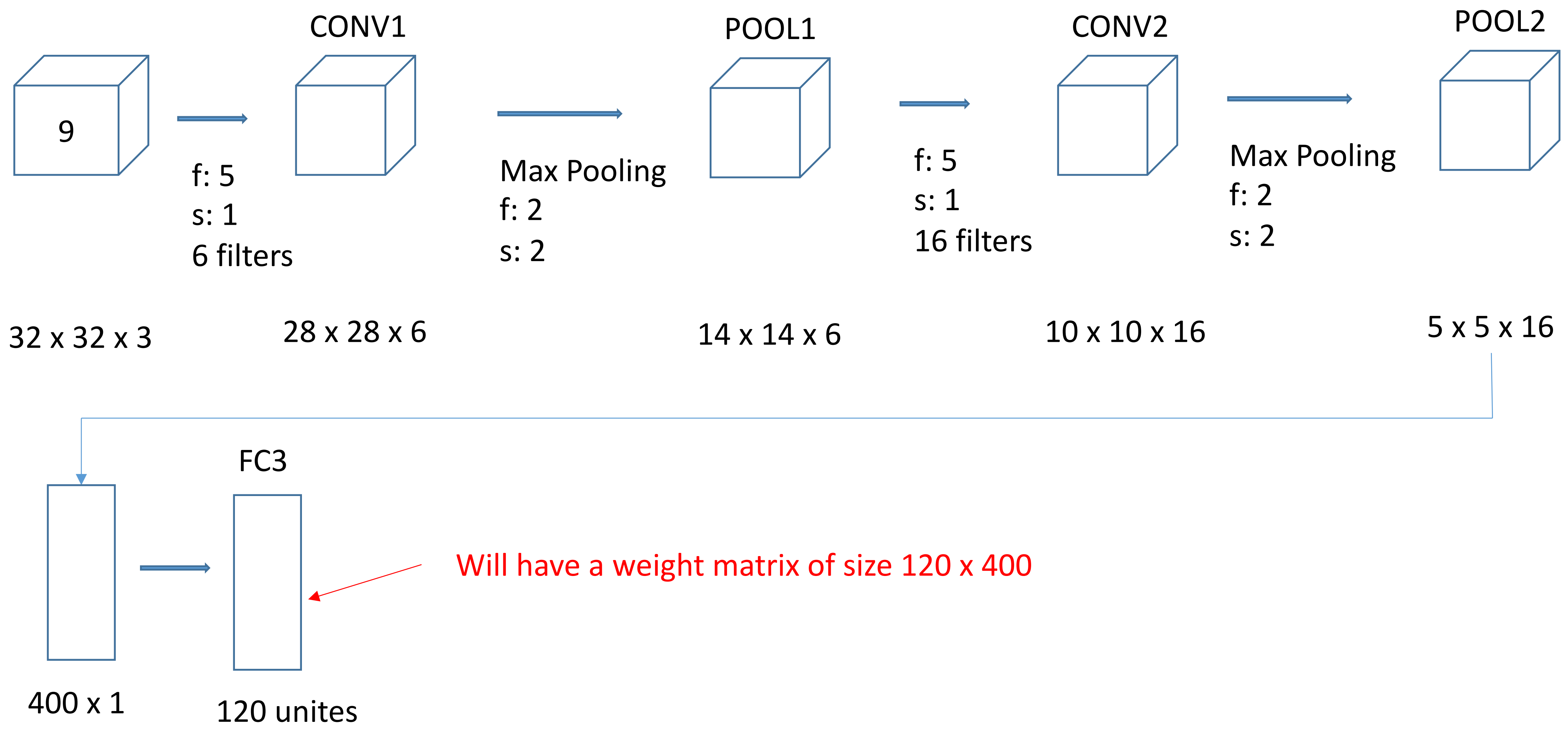
CONV1
POOL1
CONV2
POOL2
9
f: 5
s: 1
6 filters
Max Pooling
f: 2
s: 2
f: 5
s: 1
16 filters
Max Pooling
f: 2
s: 2
32 x 32 x 3
28 x 28 x 6
14 x 14 x 6
10 x 10 x 16
5 x 5 x 16
400 x 1

Example

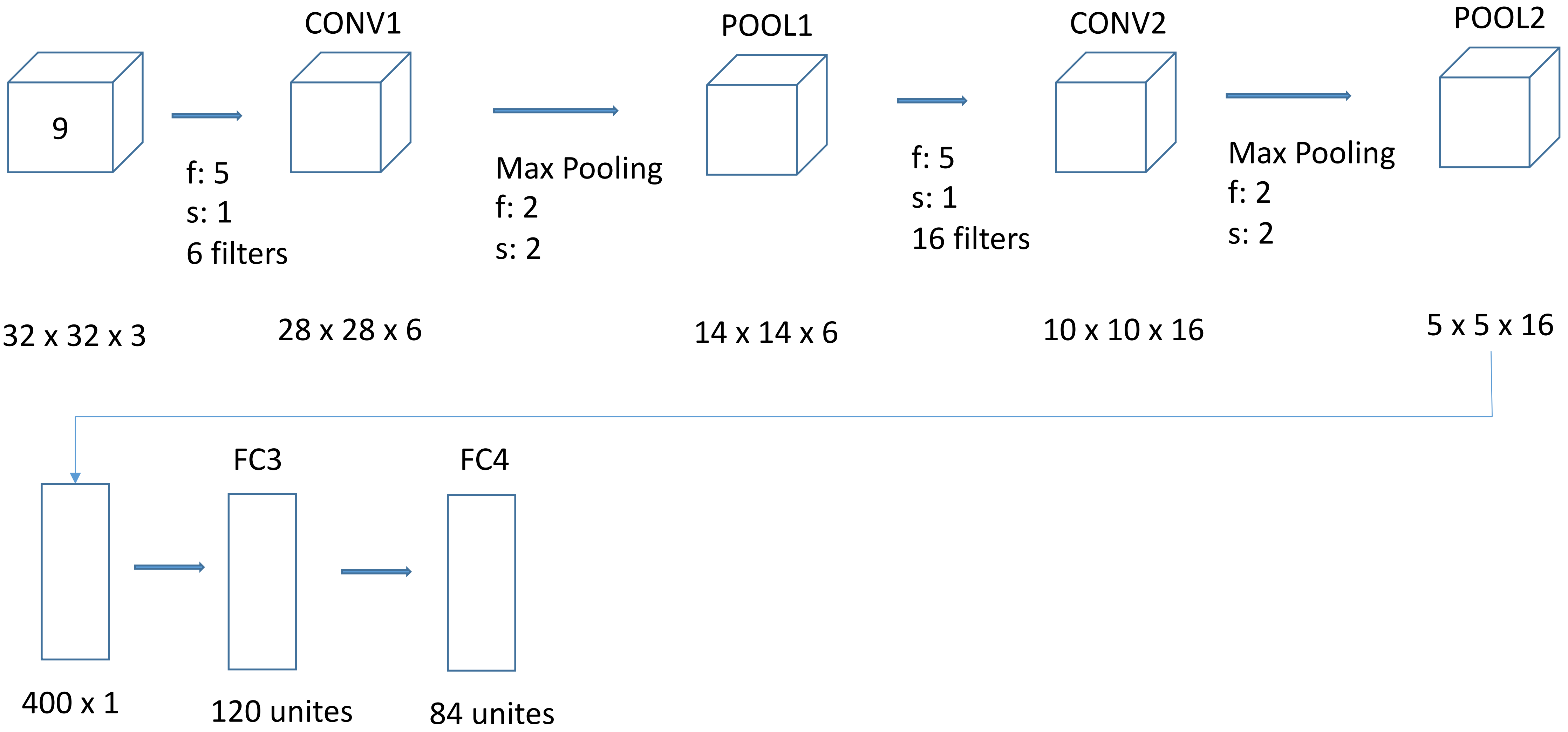


9
32 x 32 x 3
f: 5
s: 1
6 filters
CONV1
28 x 28 x 6
Max Pooling
f: 2
s: 2
POOL1
14 x 14 x 6
f: 5
s: 1
16 filters
CONV2
10 x 10 x 16
Max Pooling
f: 2
s: 2
POOL2
5 x 5 x 16
400 x 1
FC3
120 unites

Example

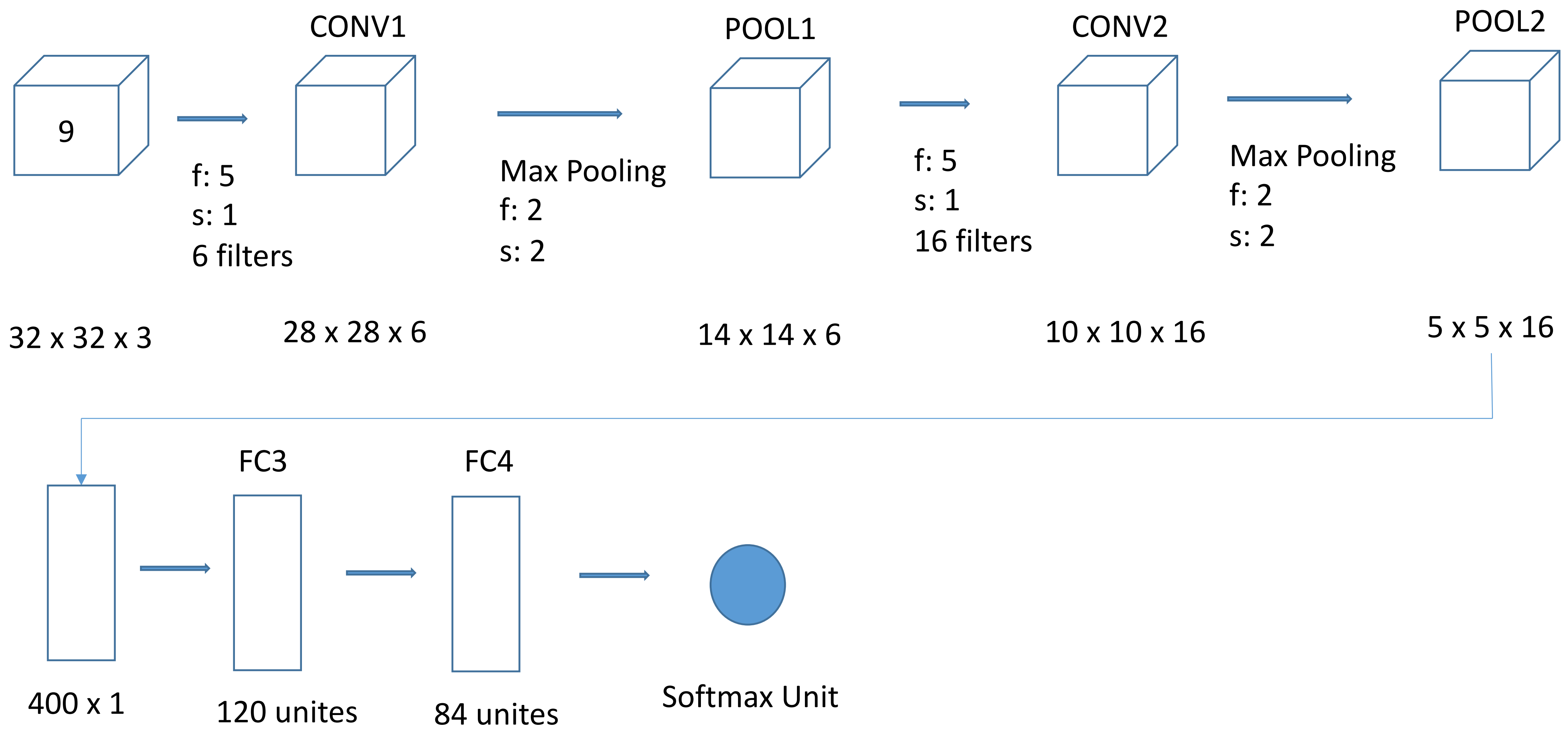

CONV1
POOL1
CONV2
POOL2
9
f: 5
s: 1
6 filters
Max Pooling
f: 2
s: 2
f: 5
s: 1
16 filters
Max Pooling
f: 2
s: 2
32 x 32 x 3
28 x 28 x 6
14 x 14 x 6
10 x 10 x 16
5 x 5 x 16
FC3
Will have a weight matrix of size 120 x 400
400 x 1
120 unites

Example



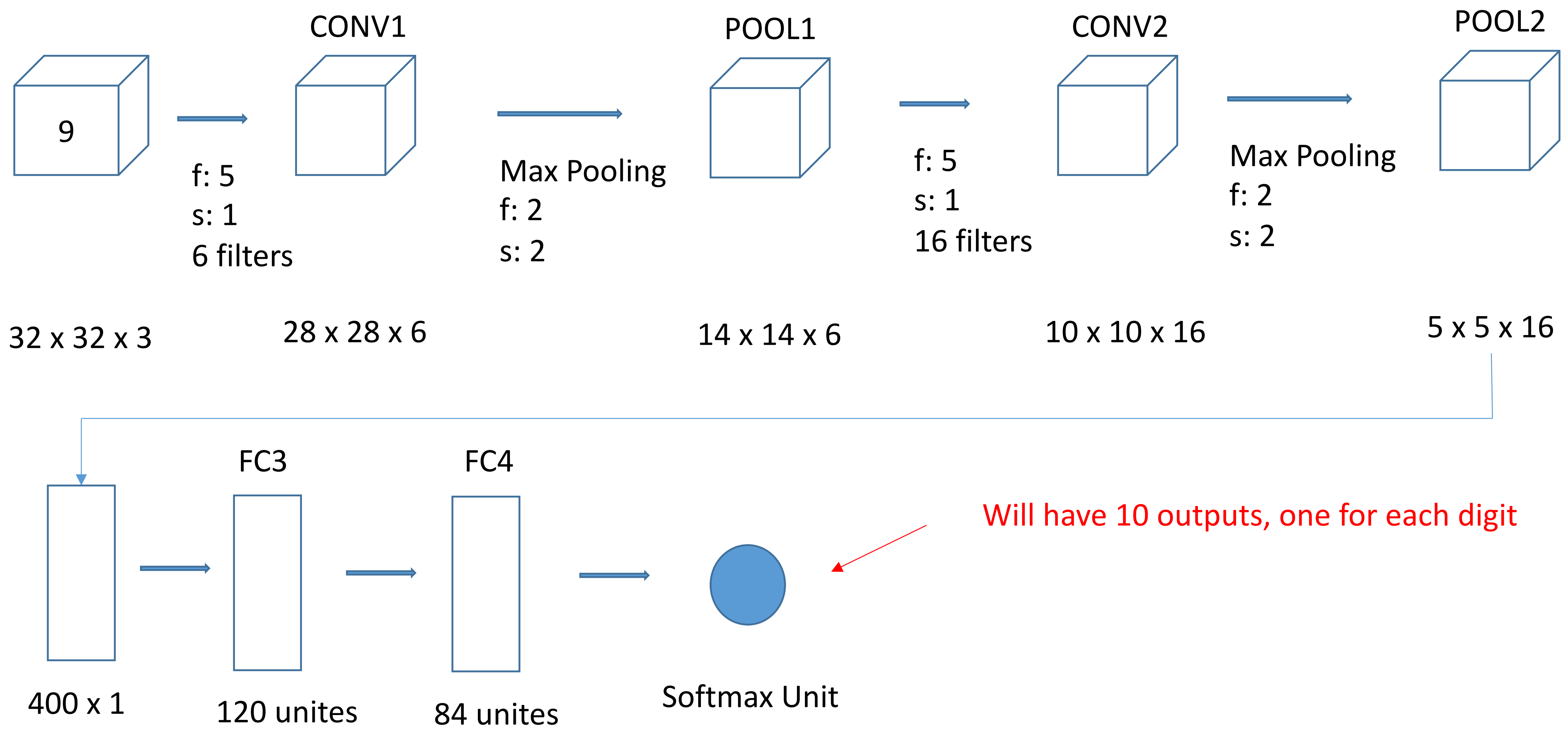
9
CONV1
POOL1
CONV2
POOL2
f: 5
s: 1
6 filters
Max Pooling
f: 2
s: 2
f: 5
s: 1
16 filters
Max Pooling
f: 2
s: 2
32 x 32 x 3
28 x 28 x 6
14 x 14 x 6
10 x 10 x 16
5 x 5 x 16
FC3
FC4
400 x 1
120 unites
84 unites

Example



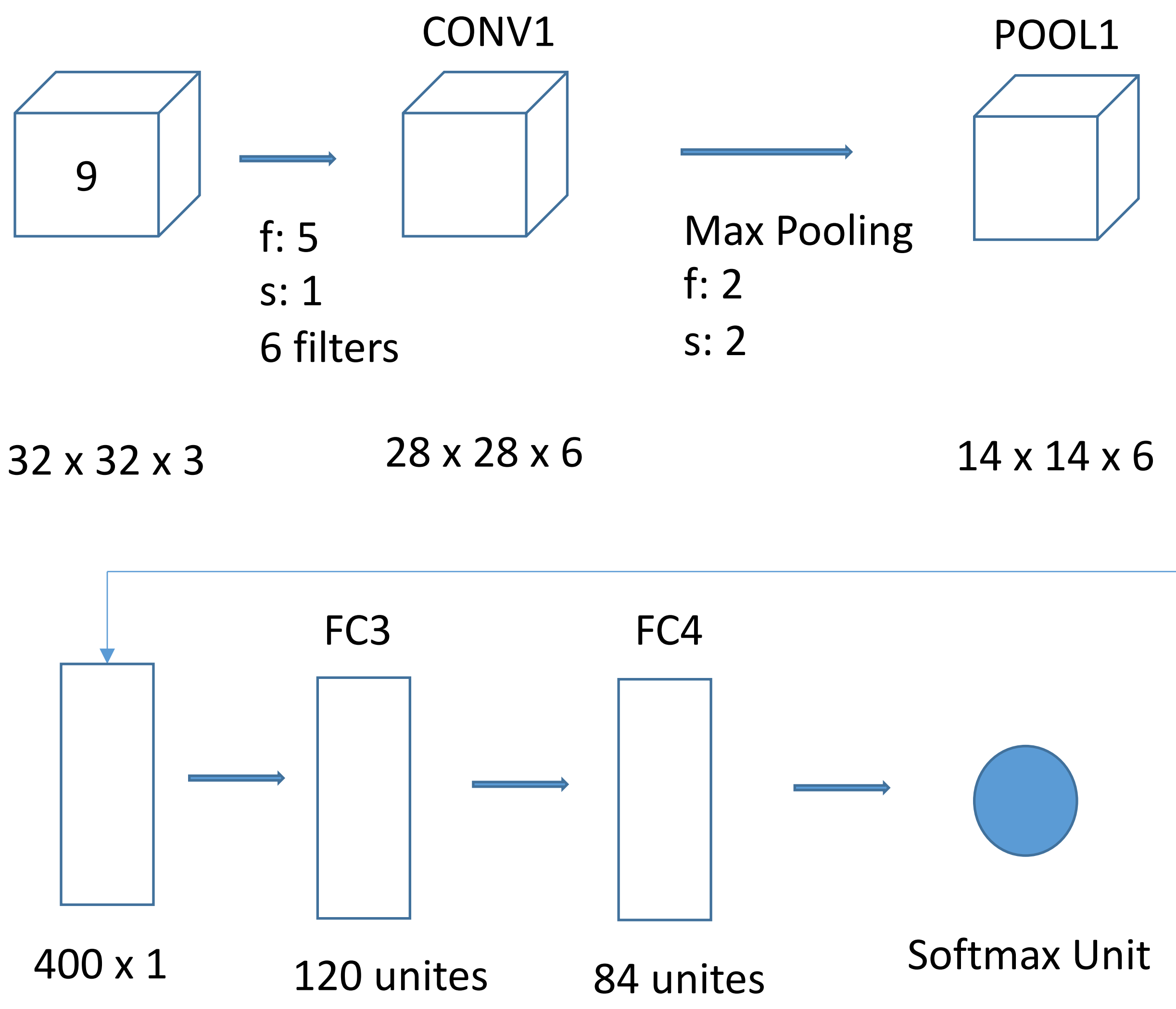
CONV1
POOL1
CONV2
POOL2
9
f: 5
s: 1
6 filters
Max Pooling
f: 2
s: 2
f: 5
s: 1
16 filters
Max Pooling
f: 2
s: 2
32 x 32 x 3
28 x 28 x 6
14 x 14 x 6
10 x 10 x 16
5 x 5 x 16
FC3
FC4
400 x 1
120 unites
84 unites
Softmax Unit

Example



CONV1
POOL1
CONV2
POOL2
9
f: 5
s: 1
6 filters
Max Pooling
f: 2
s: 2
f: 5
s: 1
16 filters
Max Pooling
f: 2
s: 2
32 x 32 x 3
28 x 28 x 6
14 x 14 x 6
10 x 10 x 16
5 x 5 x 16
FC3
FC4
Will have 10 outputs, one for each digit
400 x 1
120 unites
84 unites
Softmax Unit

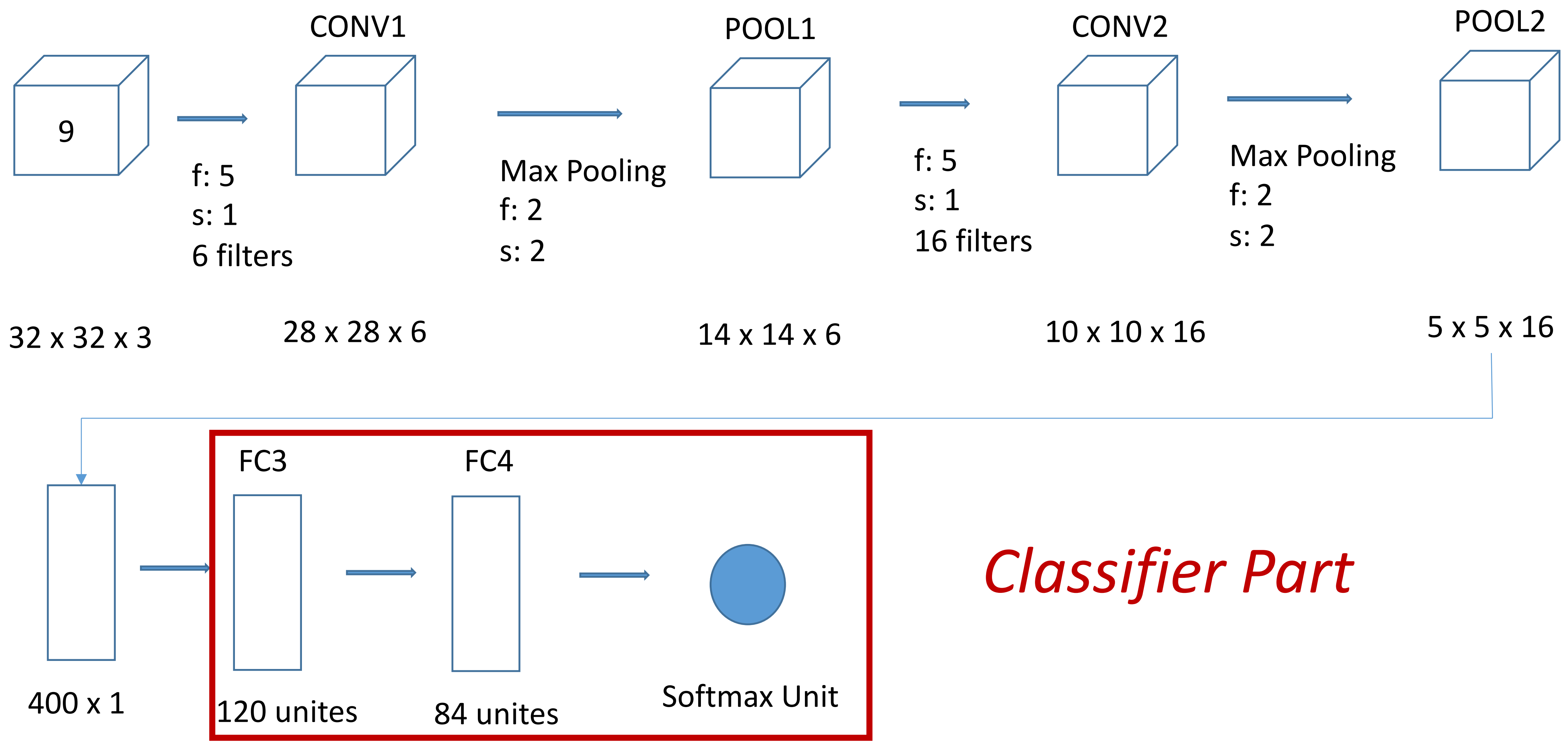
Example



What happens in a forward pass?

What happens in a backward pass?

Example



CONV1
POOL1
CONV2
POOL2
9
f: 5
s: 1
6 filters
Max Pooling
f: 2
s: 2
f: 5
s: 1
16 filters
Max Pooling
f: 2
s: 2
32 x 32 x 3
28 x 28 x 6
14 x 14 x 6
10 x 10 x 16
5 x 5 x 16
FC3
FC4
400 x 1
120 unites
84 unites
Softmax Unit
Classifier Part

CNNs as Feature Extractor

- A trained CNN (for classification) can also be used as a feature extractor for other classifiers.
- When treating networks as a feature extractor, we essentially "chop off" the classifier part
- But it can be at any other arbitrary point depending on the dataset

CNNs

- As you saw in the example, there could be a *large number of hyperparameters*

- A common practice is to look for their values in literature, and use what has worked for others

Reading

- Victor Powell. *Image Kernels Explained Visually*. http : / / setosa . io / ev / image - kernels/. 2015.
- Andrej Karpathy. *Convolutional Networks*. http://cs231n.github.io/convolutional- networks/
- Jost Tobias Springenberg et al. "Striving for Simplicity: The All Convolutional Net". In: *CoRR* abs/1412.6806 (2014). URL: http://arxiv.org/abs/1412.6806
- Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *CoRR* abs/1502.03167 (2015). URL: http://arxiv.org/abs/1502.03167
- Andrew Ng's lecture on CNN

Summary

- Understanding image convolution and its applications
- Why do we need convolutional neural networks (CNNs)?
- Padding, Stride, Multiple Channels, and Multiple Filters
- Types of Layers in a CNN
- See a simple example of CNN step by step