filters to detect different edges:

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

Vertical

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

Horizontal

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

Sobel filter

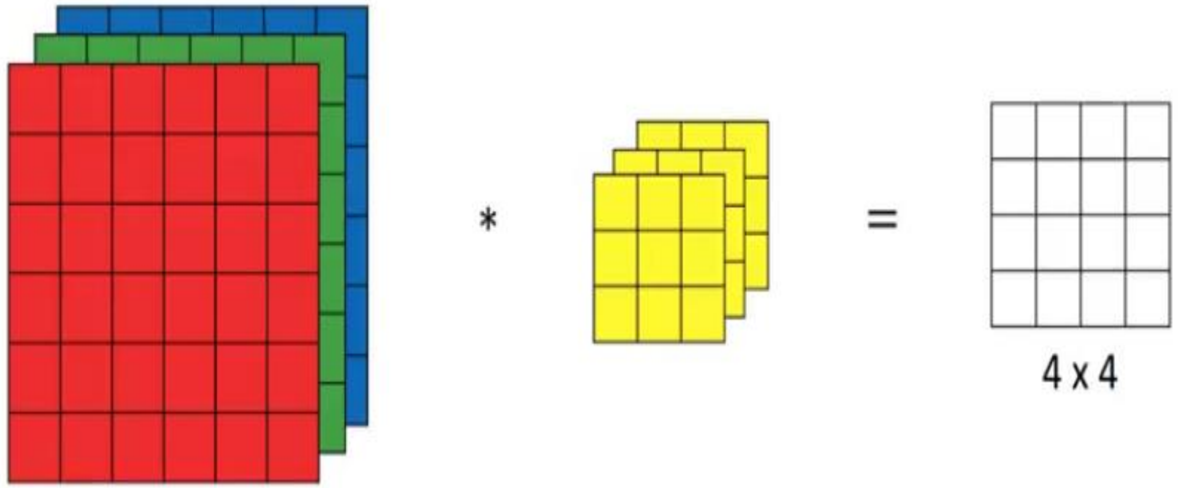
3	0	-3
10	0	-10
3	0	-3

Scharr filter

The Sobel filter puts a little bit more weight on the central pixels. Instead of using these filters, we can create our own as well

- Suppose, instead of a 2-D image, we have a 3-D input image of shape 6 X 6 X 3. How will we apply convolution on this image? We will use a 3 X 3 X 3 filter instead of a 3 X 3 filter. Let's look at an example:
- **Input:** 6 X 6 X 3
- **Filter:** 3 X 3 X 3

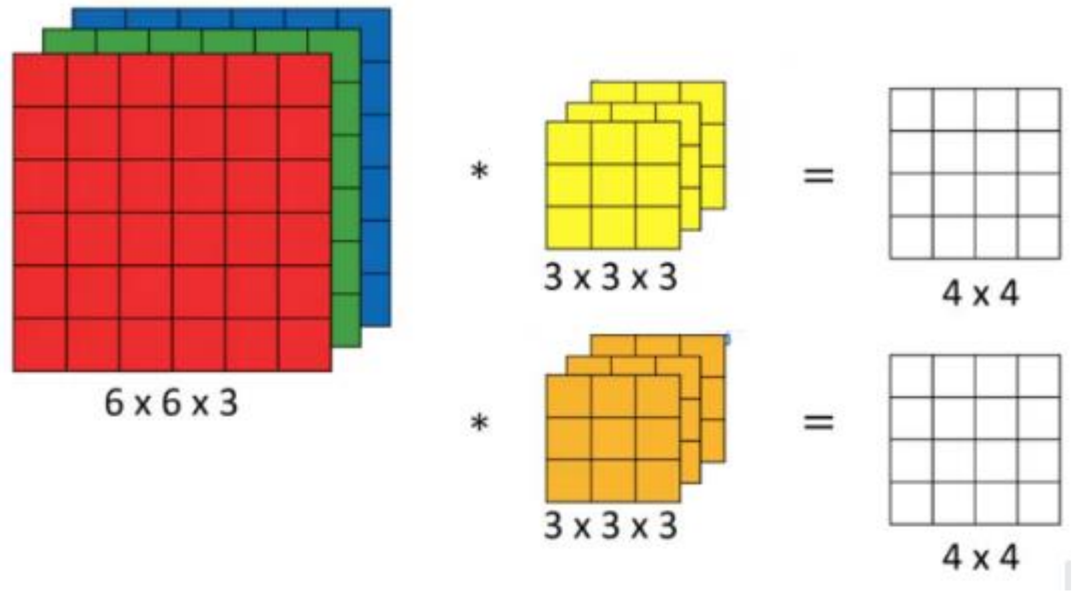
The number of channels in the input and filter should be same. This will result in an output of 4 X 4.



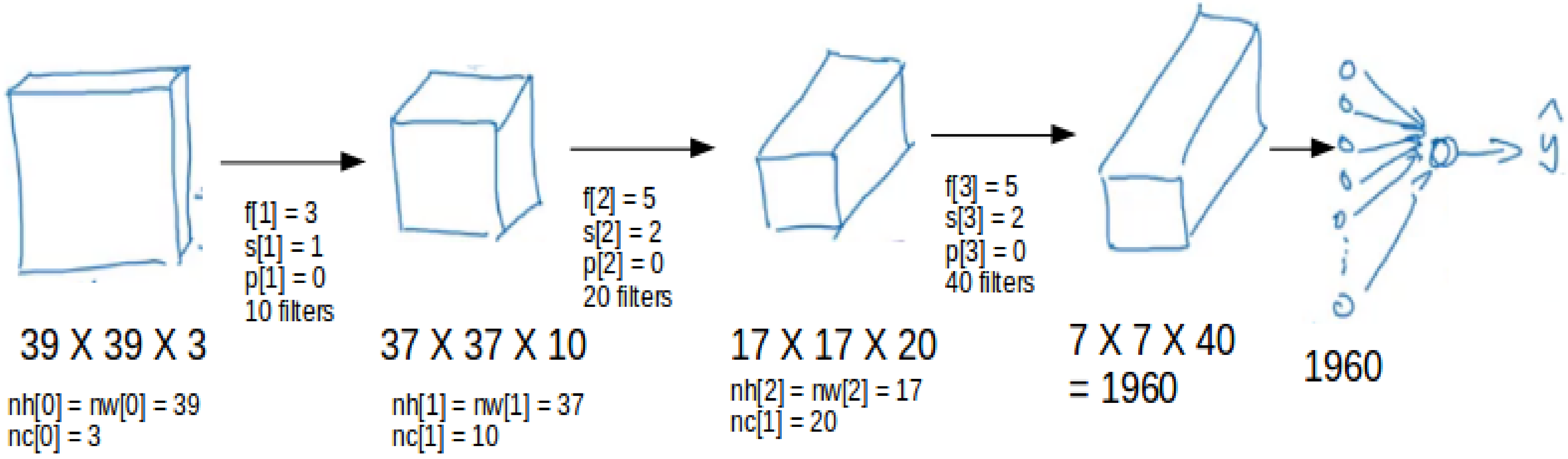
- Since there are three channels in the input, the filter will consequently also have three channels. After convolution, the output shape is a 4 X 4 matrix.
- So, the first element of the output is the sum of the element-wise product of the first 27 values from the input (9 values from each channel) and the 27 values from the filter. After that we convolve over the entire image.

Multiple Filters Edges

- Instead of using just a single filter, we can use multiple filters as well. How do we do that? Let's say the first filter will detect vertical edges and the second filter will detect horizontal edges from the image. If we use multiple filters, the output dimension will change. So, instead of having a 4 X 4 output as in the above example, we would have a 4 X 4 X 2 output (if we have used 2 filters)



Simple Convolutional Neural Networks



nh[1] = nw[1] = (n + 2p – f) / s + 1
nh[1] = nw[1] = (39 + 0 – 3) / 1 + 1
nh[1] = nw[1] = 37

One Layer of a Convolutional Network

- Once we get an output after convolving over the entire image using a filter, we add a bias term to those outputs and finally apply an activation function to generate activations. *This is one layer of a convolutional network*. the equation for one forward pass is given by:
- $z^{[1]} = w^{[1]} * x^{[0]} + b^{[1]}$
  $a^{[1]} = g(z^{[1]})$
- In our case, input (6 X 6 X 3) is $x^{[0]}$and filters (3 X 3 X 3) are the weights $w^{[1]}$. These activations from layer 1 act as the input for layer 2, and so on.

- LeNet is a convolutional neural network that Yann LeCun introduced in 1989.
- LeNet is a common term for LeNet-5, a simple convolutional neural

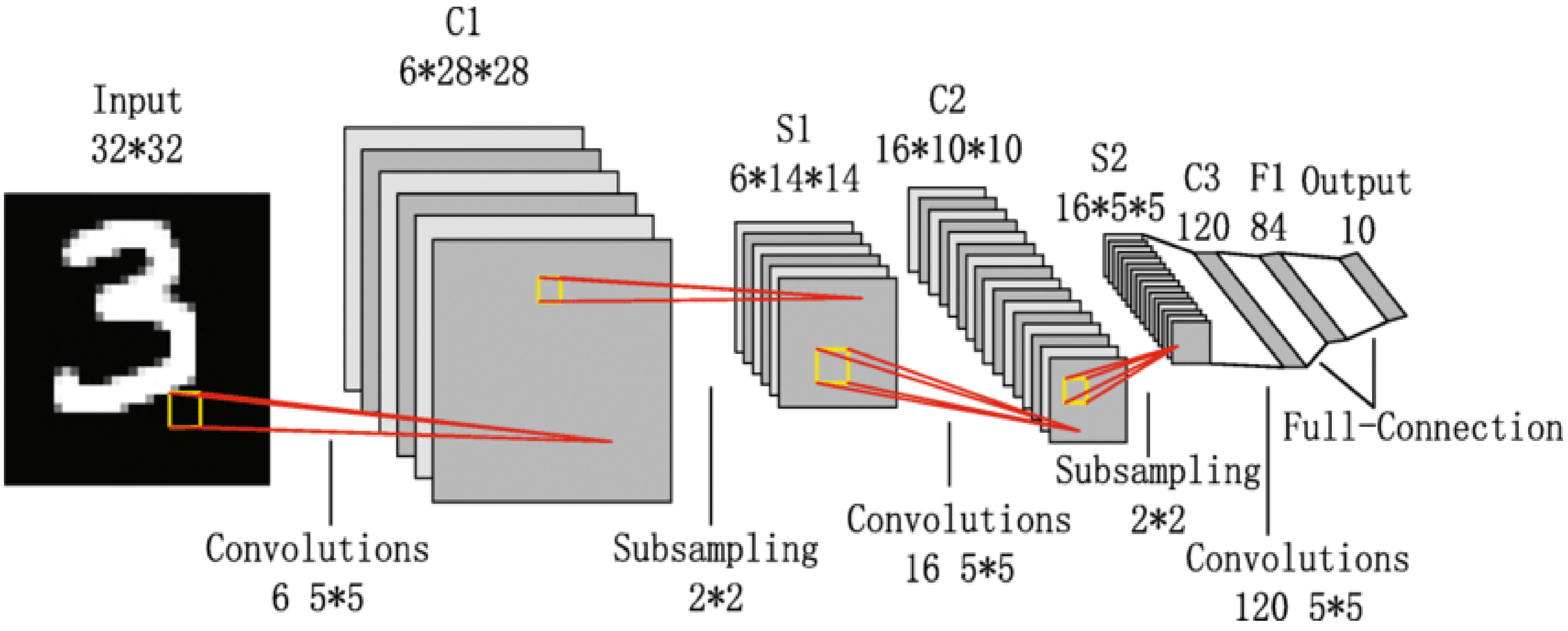
Features of LeNet-5

- Every convolutional layer includes three parts: convolution, pooling, and nonlinear activation functions
- Using convolution to extract spatial features (Convolution was called receptive fields originally)
- **The average pooling layer** is used for subsampling.
- ‘**tanh**’ is used as the activation function
- Using **Multi-Layered Perceptron** or **Fully Connected Layers** as the last classifier
- The sparse connection between layers reduces the complexity of computation

- It consists of 7 layers. The first layer consists of an input image with dimensions of 32×32. It is convolved with 6 filters of size 5×5 resulting in dimension of 28x28x6. The second layer is a Pooling operation which filter size 2×2 and stride of 2. Hence the resulting image dimension will be 14x14x6.

- Similarly, the third layer also involves in a convolution operation with 16 filters of size 5×5 followed by a fourth pooling layer with similar filter size of 2×2 and stride of 2. Thus, the resulting image dimension will be reduced to 5x5x16.

The Architecture of LeNet5



LeNet-5 Architecture

Summary of LeNet-5 Architecture

Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax