Convolutional Neural Networks

Learning Objectives:

Explain the convolution operation

Apply two different types of pooling operations

Identify the components used in a convolutional neural network (padding, stride, filter, ...) and their purpose

Build a convolutional neural network

Implement convolutional and pooling layers in numpy, including forward propagation

Implement helper functions to use when implementing a TensorFlow model

Create a mood classifer using the TF Keras Sequential API

Convolutional Neural Networks

Learning Objectives:

Build a ConvNet to identify sign language digits using the TF Keras Functional API

Build and train a ConvNet in TensorFlow for a binary classification problem

Build and train a ConvNet in TensorFlow for a multiclass classification problem

Explain different use cases for the Sequential and Functional APIs

Convolutional Neural Networks

1 Why Convolutions Neural Networks?
2 Computer vision
3 Edge detection example
4 More edge detection
5 Padding
6 Strided convolutions
7 Convolutions over volumes
8 One layer of a convolutional network
9 A simple convolution network example
10 Pooling layers
11 Convolutional neural network example

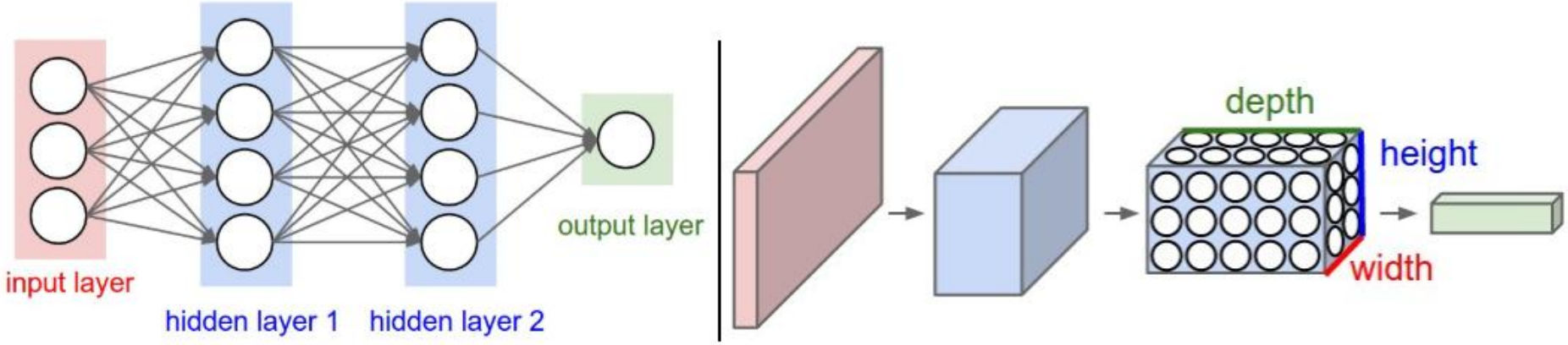
Why Convolutions Neural Networks?

Why Convolutional Neural Networks?

- 1. Regular Neural Nets struggle with image scalability:
- In the MNIST dataset, with 28x28x1 images, a single fully-connected neuron in the first hidden layer would have 786 weights, which is manageable.
- However, for larger images like 200x200x3, a neuron would have 120,000 weights, and having multiple neurons would lead to a vast number of parameters, promoting overfitting. This full connectivity becomes impractical and wasteful.

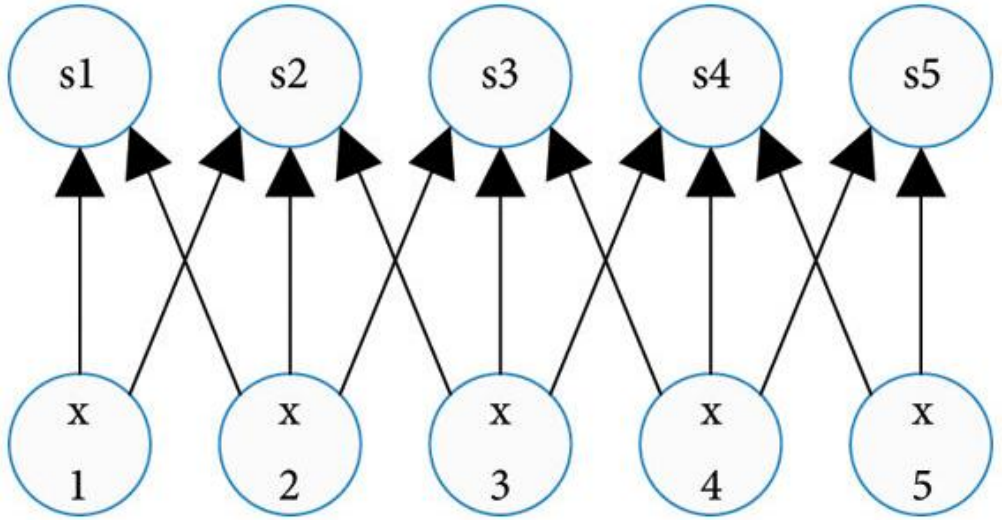
Why Convolutional Neural Networks?

- 2. The main advantages of convolutional neural networks are parameter sharing and sparsity of connections:
  - Parameter sharing allows the reuse of feature detectors across different parts of the image, reducing the number of parameters needed.
  - Sparsity of connections ensures that each output unit depends on only a subset of input features, further minimizing the required parameters.
- Convolutional Neural Networks (ConvNets) leverage the 3D structure of input images, unlike regular neural networks. In a ConvNet, layers have neurons arranged in three dimensions: width, height, and depth



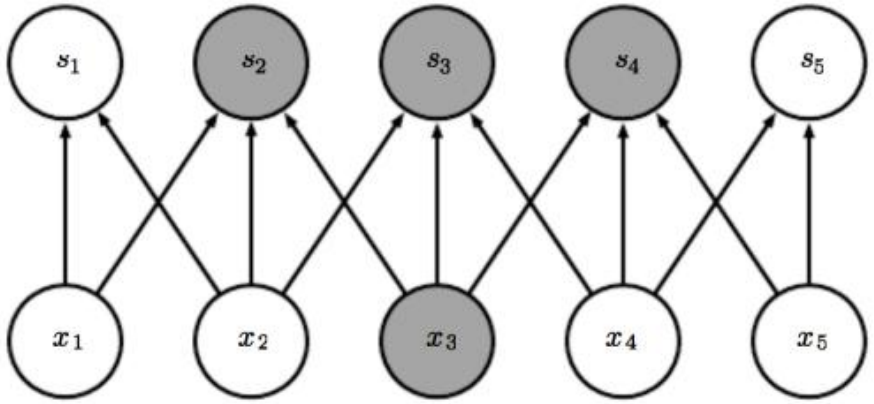
Why Convolutional Neural Networks?

- Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.



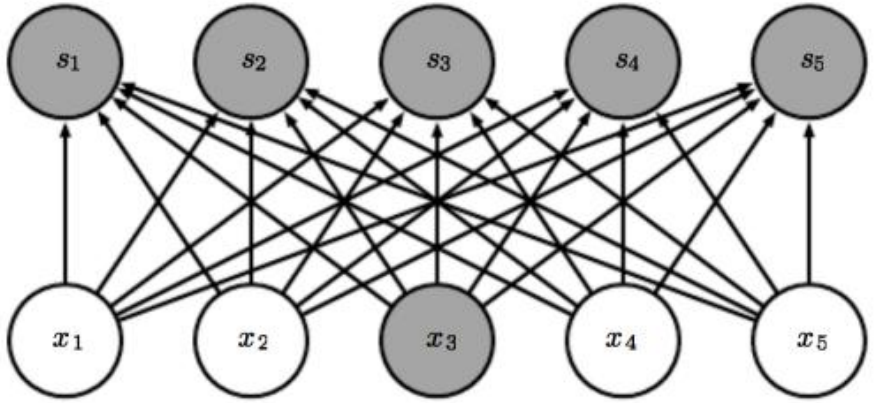
Why Convolutional Neural Networks?

- Sparsity of connections: In each layer, each output value depends only on a small number of inputs.



Highlight one input x_3 and output units s affected by it.

The above image: when s is formed by convolution with a kernel of width 3, only three outputs are affected by x_3

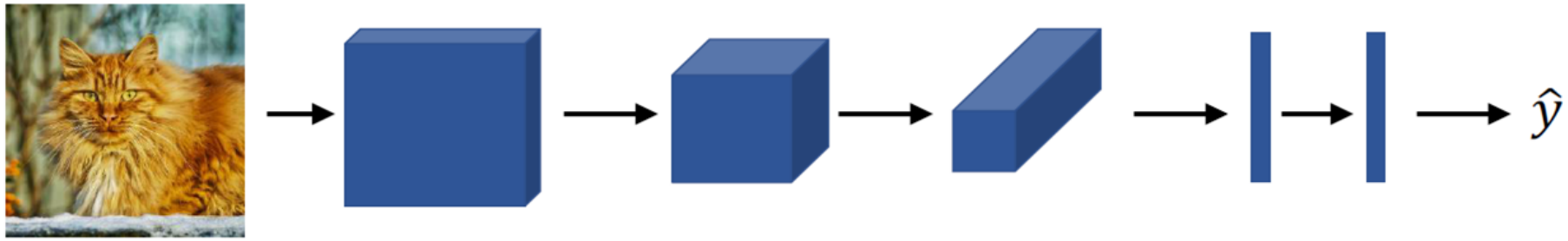


The below image: when s is formed by matrix multiplication connectivity is no longer sparse

– So all outputs are affected by x_3

Putting it together

Training set $(x^{(1)}, y^{(1)})$ ... $(x^{(m)}, y^{(m)})$.



$$\text{Cost } J = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

Use gradient descent to optimize parameters to reduce J

FPT UNIVERSITY

Convolutional Neural Networks

Computer vision

Computer vision

- A major issue arises with large image inputs, containing millions of features, making standard fully connected networks impractical.
- To overcome this, convolutional neural networks (CNNs) are introduced. CNNs use convolutions, better suited for processing large images. Convolutional networks are illustrated using the example of edge detection.

Computer Vision Problems

Image Classification

64x64x3

Cat? (0/1)

Object detection

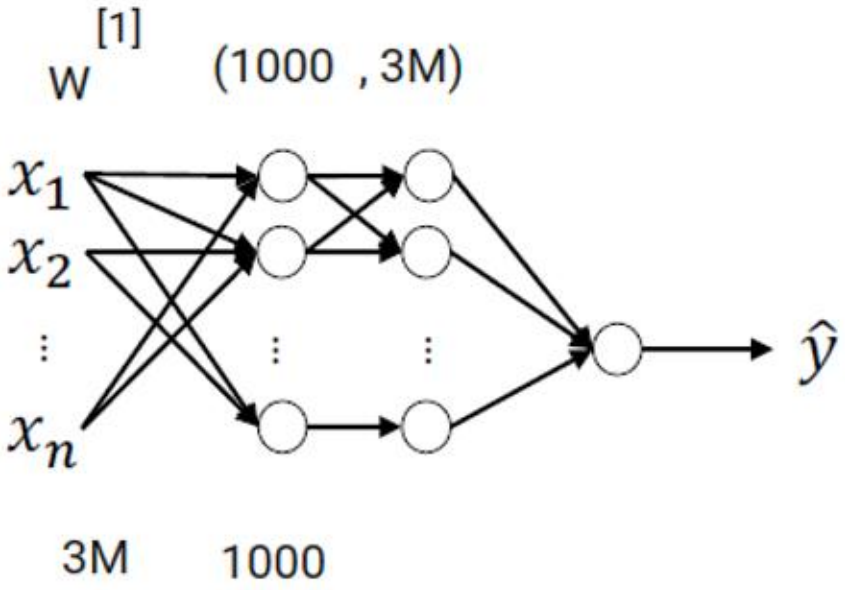
Neural Style Transfer

Deep Learning on large images

64x64x3 = 12228

→ Cat? (0/1)

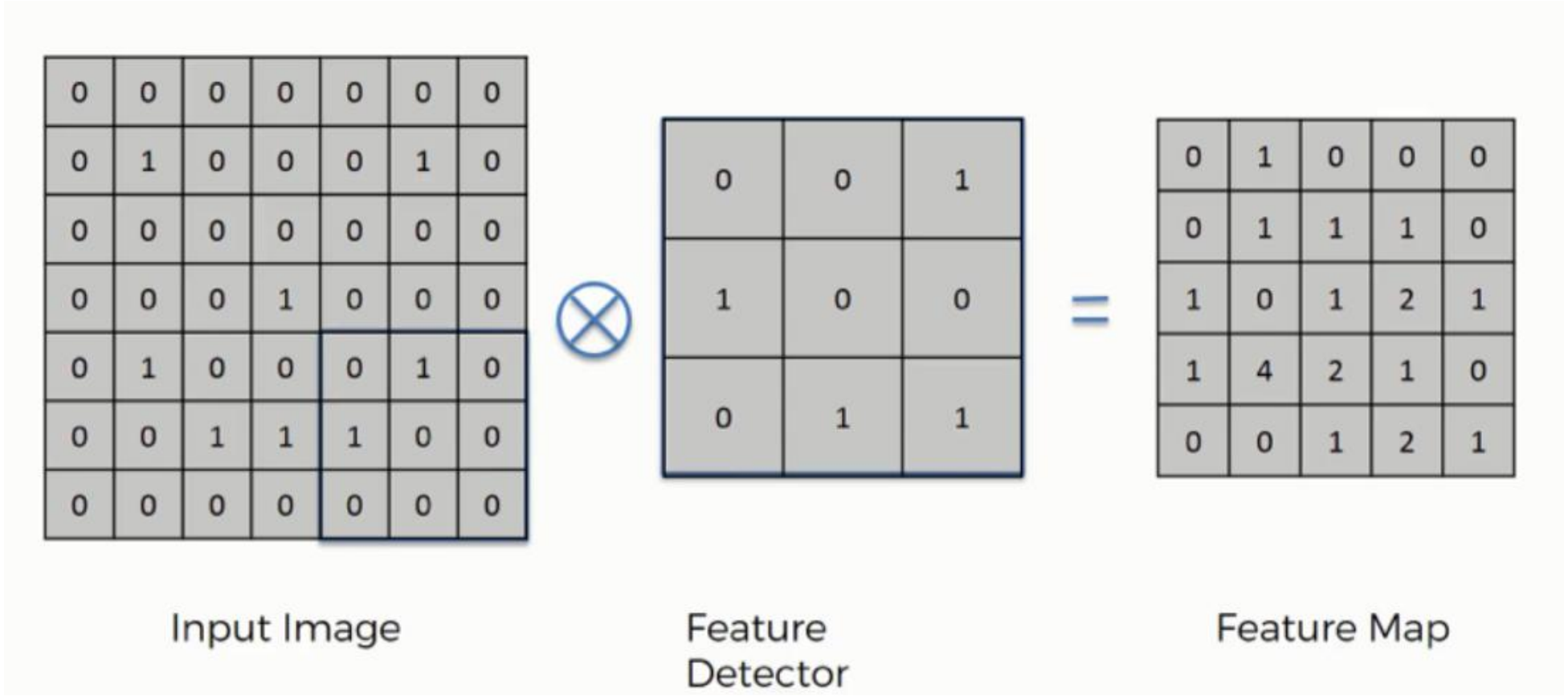
1000 x 1000 x 3 = 3 M (million)



Convolution

- Convolution is an operation done to extract features from the images as these features will be used by the network to learn about a particular image. In the case of a dog image, the feature could be the shape of a nose or the shape of an eye which will help the network later to identify an image as a dog.
- Convolution operation is performed with the help of the following three elements:
- 1. Input Image - The image to convolve on
- 2. Filter/ Feature Detector/Kernel - They are the bunch of numbers in a matrix form intended to extract features from an image. They can be 1-dimensional, 2-dimensional or 3-dimensional
- 3. Output/ Feature Map/ Activation Map - The resultant of the convolution operation performed between image and feature detector gives a Feature Map.

Convolution



Convolution

Input image

Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map

FPT Education

FPT UNIVERSITY

Edge detection example

Convolutional Neural Networks

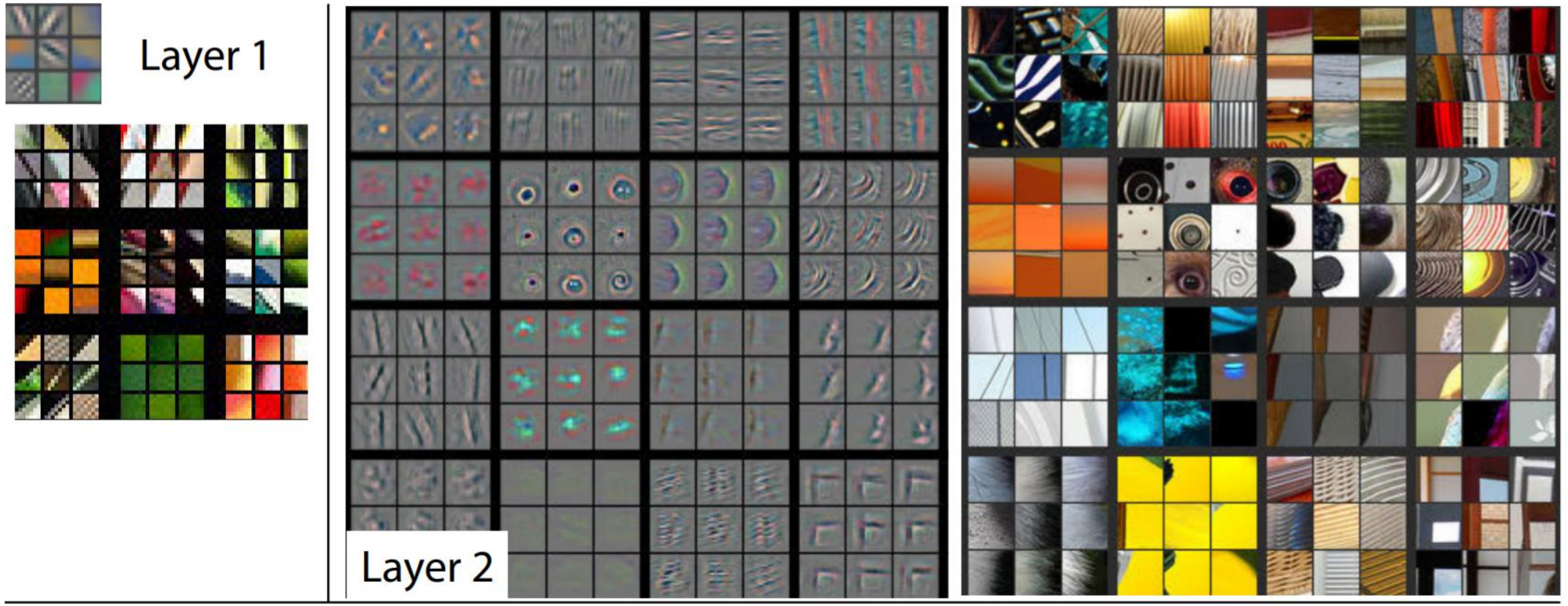
Edge detection example

- The convolution operation is a fundamental part of convolutional neural networks.
- Early neural network layers can identify edges, while subsequent layers progress to recognizing objects and eventually complete objects like faces.

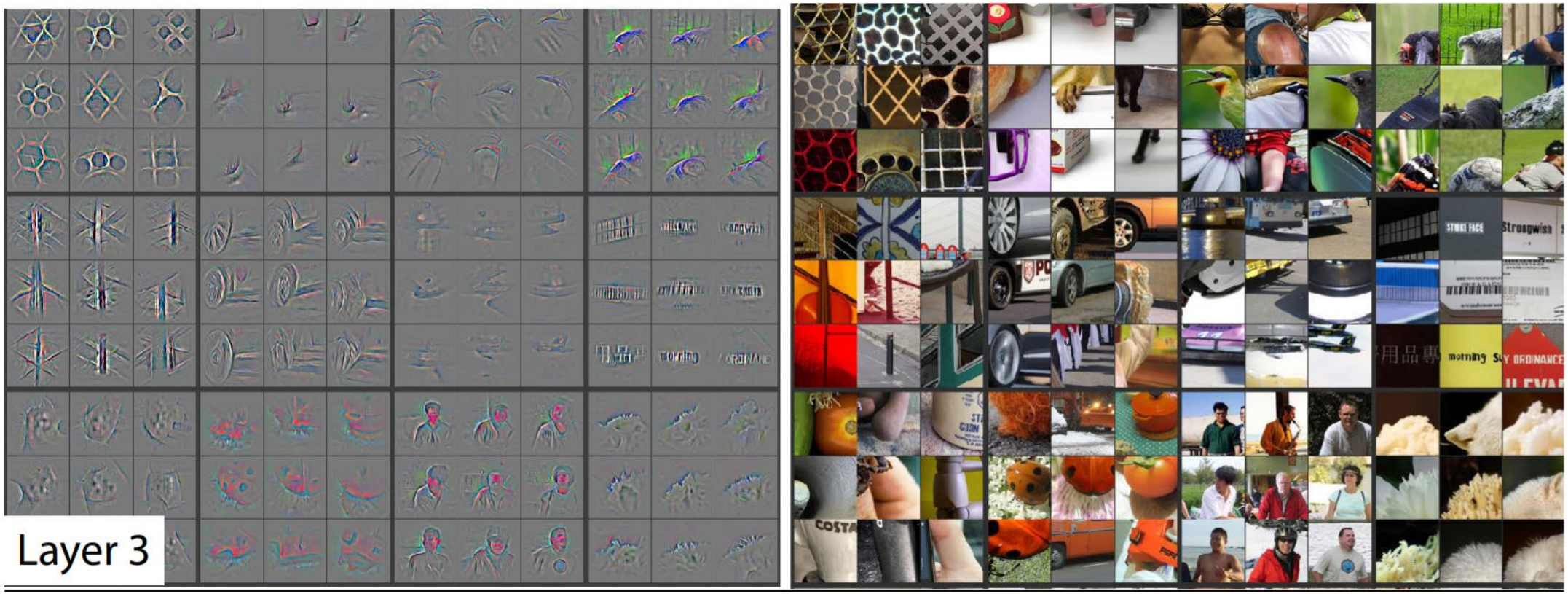
- Example 1:

Convolution

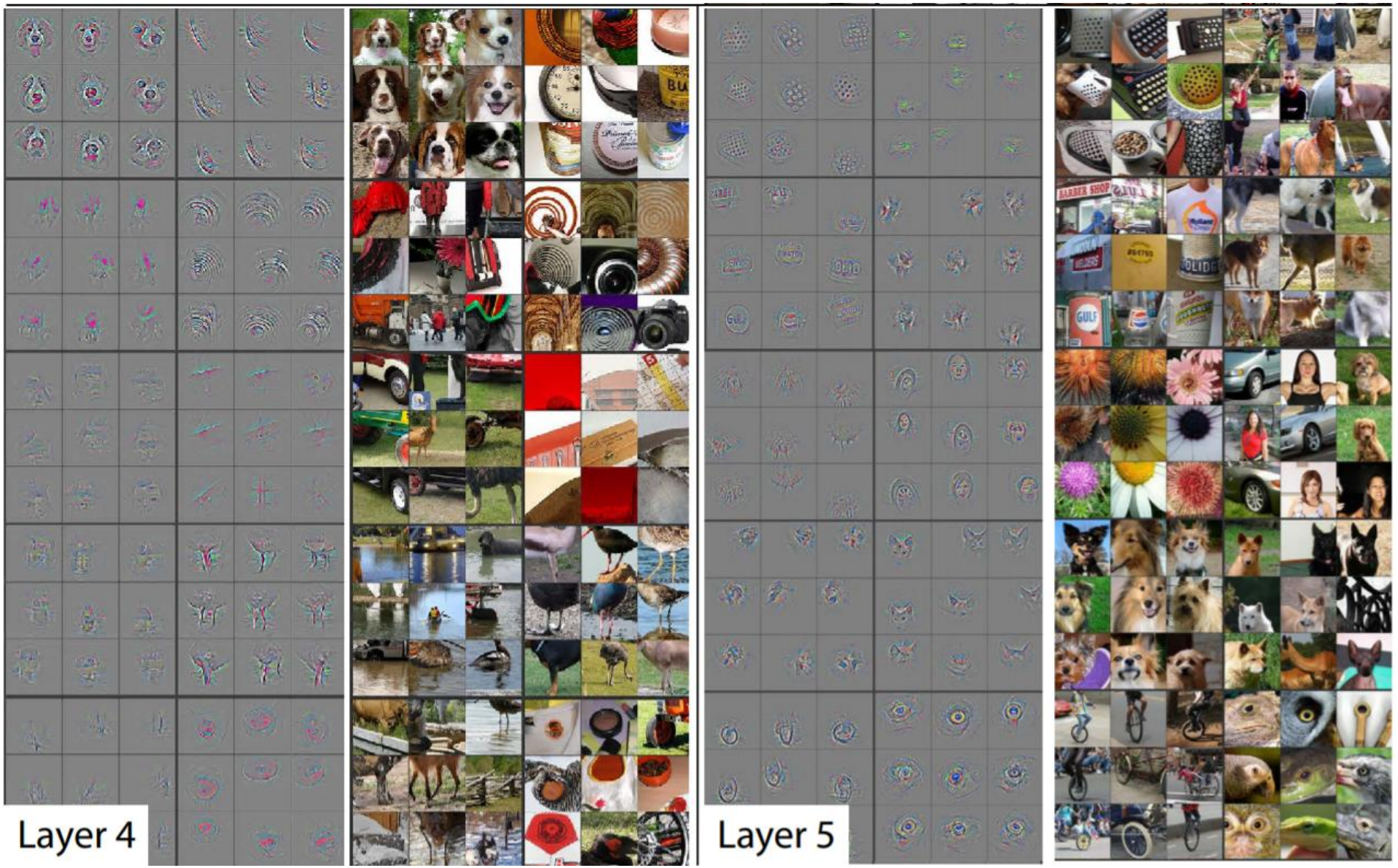
- Example 2:



Convolution

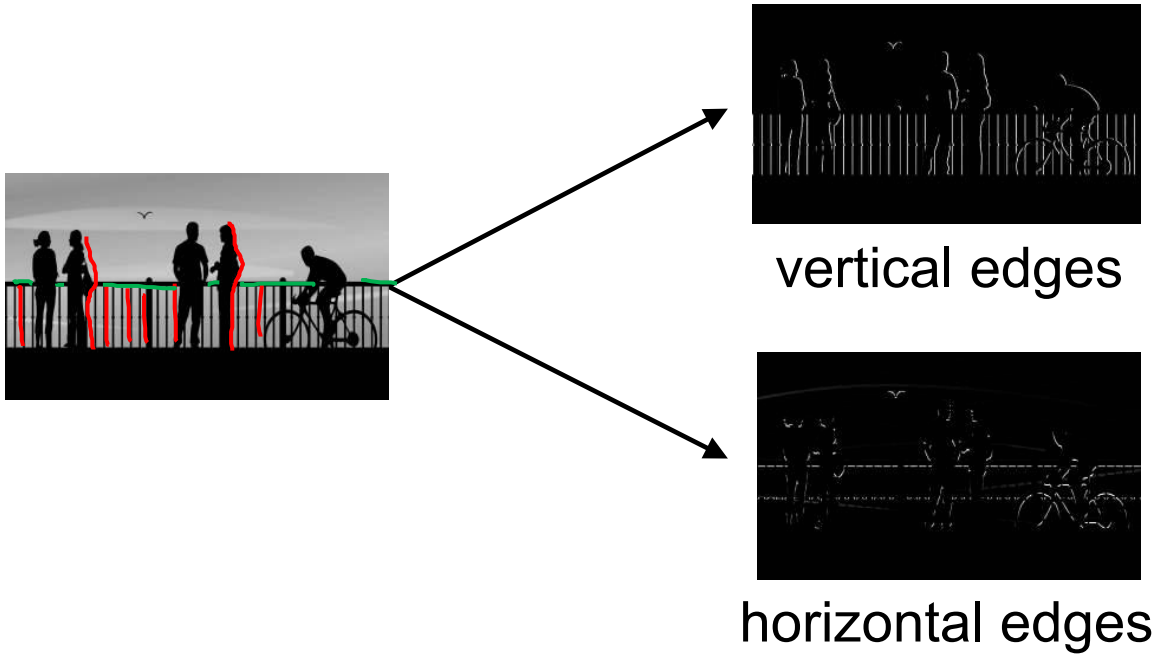


Convolution



Edge detection example

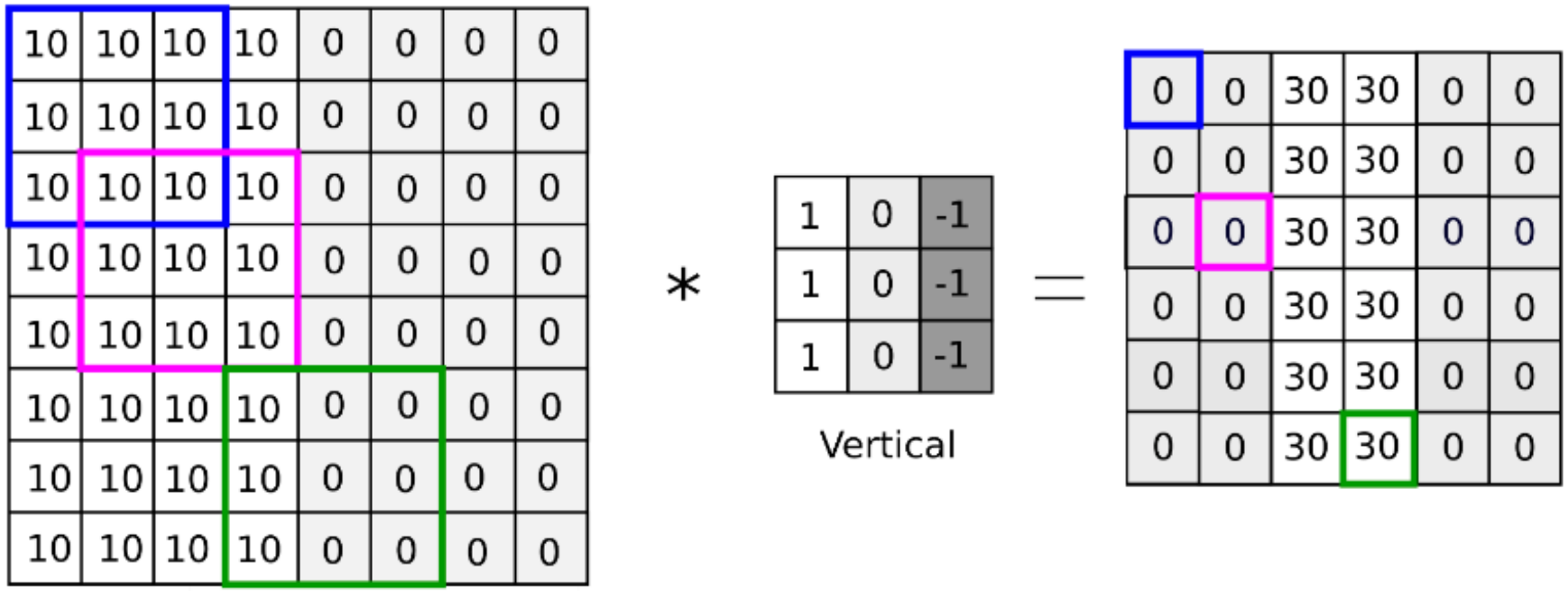
- To identify objects in a picture, a computer might start by detecting vertical edges, which correspond to buildings and pedestrians in the given example. Similarly, horizontal edges can be detected, such as the railing line in the image.



Edge detection example

- Consider an example:
  - A grayscale 8x8 image.
  - A 3x3 filter is constructed with weights of 1, 1, 1, 0, 0, 0, -1, -1, -1.
  - The filter is convolved with the input image producing a 6x6 feature map.
  - The values in the feature map correspond to the strength of vertical edges detected at different locations in the input image.

Vertical edge detection



For example, in the figure above, applying the convolution operator between the 3×3 filter (center) and the blue region of the original image (left) gives rise to the element in the new image in blue box (right), whose value can be calculated as the sum of the resulting element-wise product between the 2 matrices:

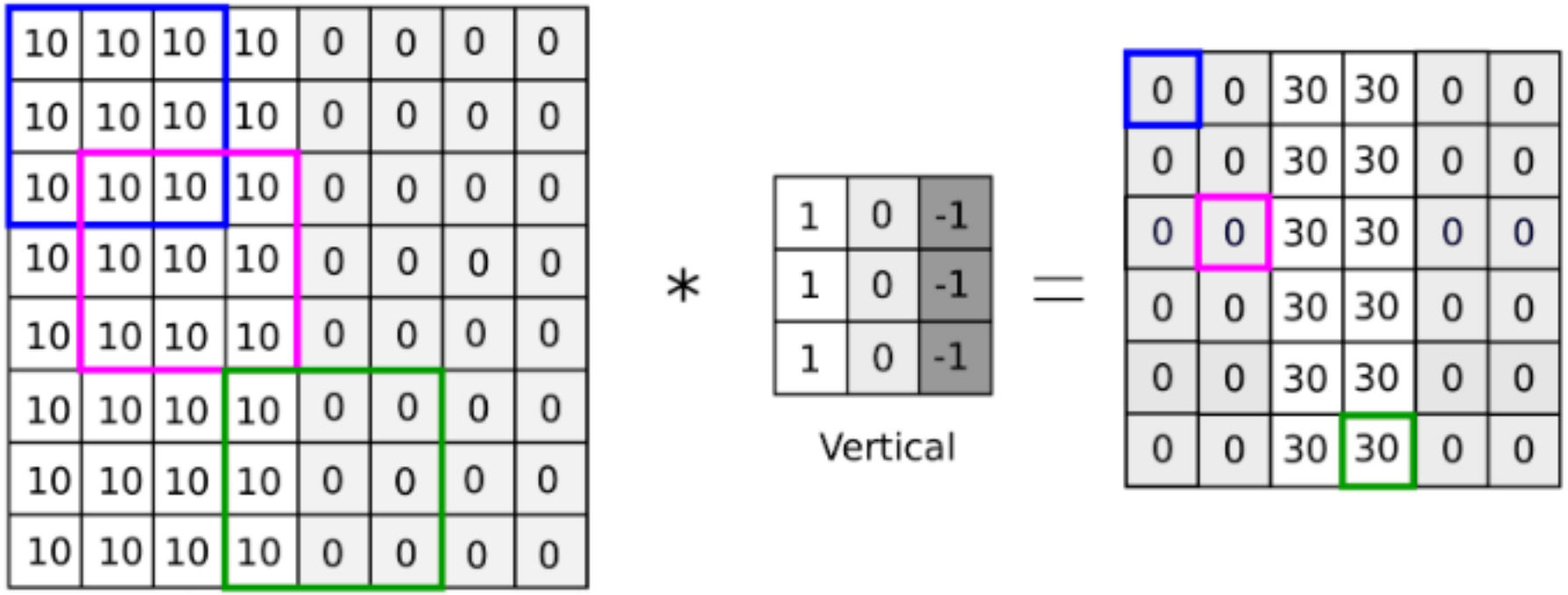
$$0 = 10 \times 1 + 10 \times 0 + 10 \times (-1) + 10 \times 1 + 10 \times 0 + 10 \times (-1)$$
$$+ 10 \times 1 + 10 \times 0 + 10 \times (-1)$$

More edge detection

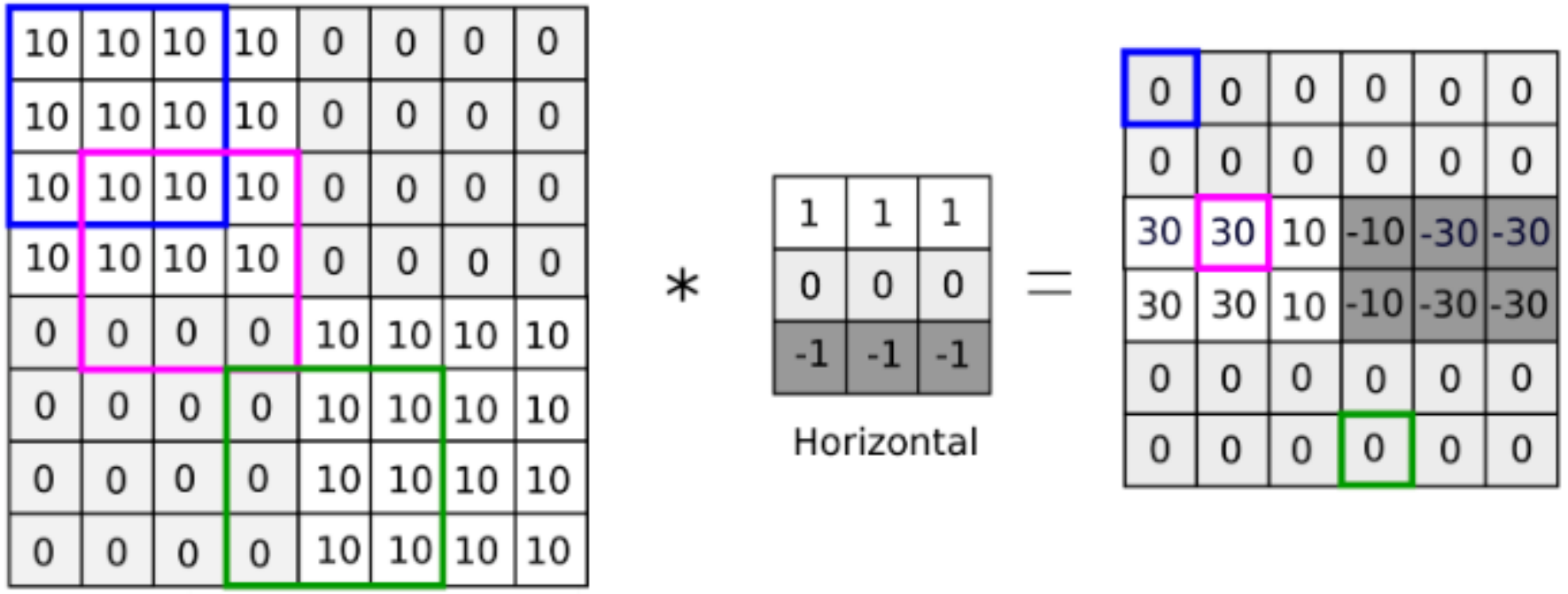
More edge detection

- Convolutional operations and edge detection:
  - The vertical edge detection filter detects edges where pixels are bright on the left and dark on the right.
  - The same filter can be used to detect horizontal edges, where pixels are bright on top and dark on the bottom.
- The concept of positive and negative edges, which are transitions from light to dark or dark to light, respectively:
  - The same vertical edge detection filter behaves differently when applied to an image with colors flipped, where the darker colors are on the left and brighter colors are on the right. In this case, the filter detects negative edges instead of positive edges.
  - Other types of edge detection filters, such as the Sobel and Scharr filters, which are alternatives to the traditional vertical edge detection filter. These filters have different weightings and properties that make them more robust in detecting edges in certain types of images.

Vertical edge detection examples



Horizontal edge detection examples



Learning to detect edges

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

Traditional 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

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

parameterized filter

More edge detection

- Using backpropagation to learn the values of the filter weights automatically:
  - Rather than hand-coding the filter values, a neural network can learn them by adjusting the weights during training.
  - This allows the network to learn edge detection filters that are more effective and efficient than hand-coded filters.

- Padding and stride, which are variations on the basic convolution operation:
  - Padding involves adding additional pixels around the edge of an image to prevent the output from being smaller than the input after convolution.
  - Stride involves skipping some of the pixels in the input during convolution, resulting in a smaller output.
  - These variations are important building blocks for convolutional neural networks.

FPT Education

FPT UNIVERSITY

Convolutional Neural Networks

Padding

Padding

- The basic convolutional operation creates two problems:
  - Firstly, if you apply convolutional operators repeatedly, the image shrinks in size every time, eventually resulting in a very small image.
  - Secondly, pixels on the edges of the image are touched by fewer filters, leading to the loss of important information near the edges.
- Padding addresses this by adding zeros to the input image edges before convolution, preventing size reduction and ensuring edge pixels are influenced by more filters.
- Conventionally, padding is denoted as 'p,' representing the added pixels to each edge of the input image.

Padding

- Input: n x n
- Padding: p
- Filter size: f x f
- Output: (n+2p-f+1) x (n+2p-f+1)

Padding

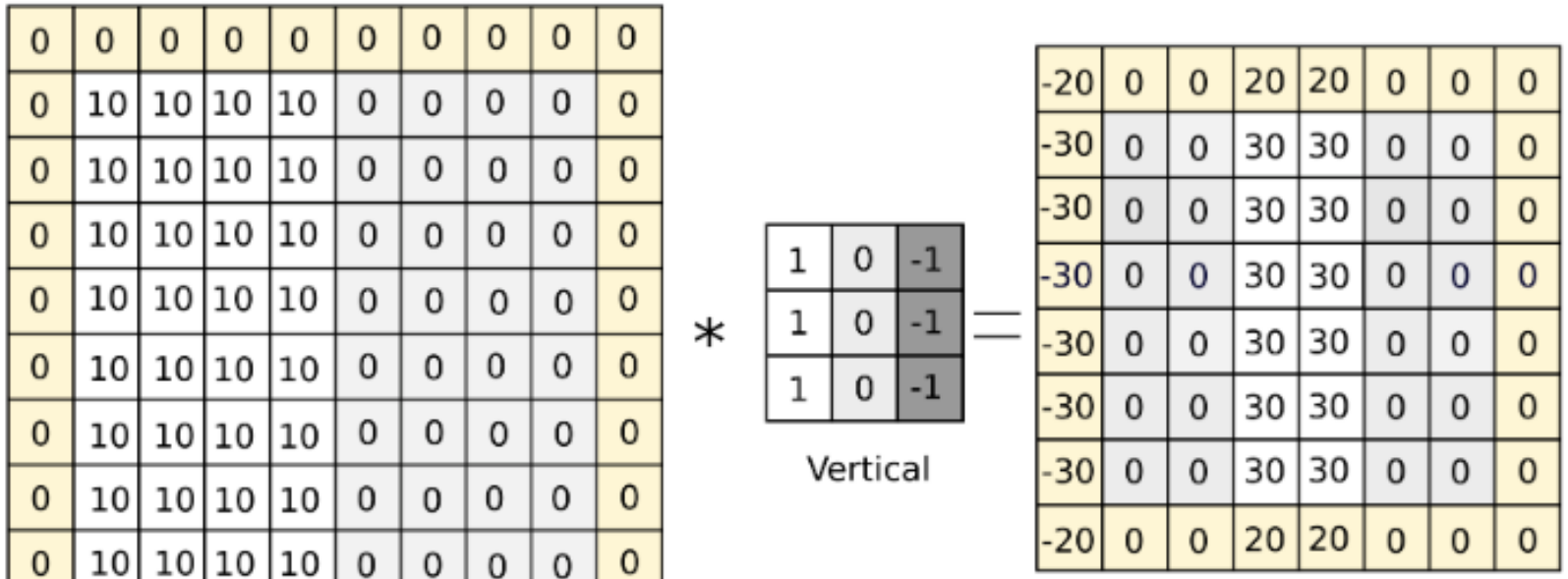
- There are two common choices for padding:
- 1. Valid: It means no padding. If we are using valid padding, the output will be (n-f+1) X (n-f+1)
- 2. Same: Here, we apply padding so that the output size is the same as the input size, i.e.,
- n+2p-f+1 = n
- So, p = (f-1)/2

Padding

If the original image is of size $n \times n$ and the filter is of size $f \times f$, the size of the resulting image is $(n-f+1) \times (n-f+1)$. As a result, the input image size is going to shrink after each convolutional layer. In addition, because the edge of the original image is not used in the convolution operation as often as the central pixels, some valuable information is potentially lost in the process.

The solution to these problems is through padding the border of the original image with p extra layer(s) of zeros in every direction. The dimensions of the input and output images become $(n+2p) \times (n+2p)$ and $(n+2p-f+1) \times (n+2p-f+1)$ respectively. When $p=0$, that is, no padding, this is called "valid" convolution. When $p=(f-1)/2$ so that the sizes of the input and output images are the same, this is called "same" convolution.

Padding



An example of same convolution with padding $p = 1$

FPT UNIVERSITY

Strided convolutions

Strided convolutions

- Strided convolutions in convolutional neural networks (CNNs) control output feature map dimensions by adjusting the filter's step size across the input image.
- Unlike traditional convolutions with one-pixel movement, strided convolutions allow the filter to move multiple pixels, reducing spatial dimensions. The output size is calculated using the formula:
- Output: $(\frac{n+2p-f}{s} + 1) \times (\frac{n+2p-f}{s} + 1)$
- Input: n x n
- Padding: p
- Stride: s
- Filter size: f x f
- If the result isn't an integer, it's rounded down. For example, with a 100×100 image, 6×6 filter, 7 padding, and 4 stride, the result is a 28×28 convolution.

Strided convolutions

- Image obtained after convolution of 6×6 image with a 3×3 filter and a stride of 2

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

*

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

2*1+6*1+3*1+3*0+6*0+4*0+7*-1+9*-1+8*-1	7*1+9*1+8*1+4*0+8*0+3*0+6*-1+7*-1+8*-1	6*1+7*1+8*1+2*0+4*0+9*0+9*-1+3*-1+7*-1
3*1+1*1+1*1+4*0+1*0+1*0+8*-1+2*-1+4*-1	8*1+2*1+4*1+3*0+4*0+5*0+8*-1+6*-1+0*-1	8*169*1+0*1+9*0+8*0+2*0+7*-1+9*-1+2*-1
1*1+2*1+1*1+1*0+2*0+3*0+4*-1+3*-1+2*-1	4*1+3*1+2*1+5*0+8*0+7*0+0*-1+1*-1+2*-1	0*1+1*1+2*1+2*0+1*0+2*0+2*-1+2*-1+0*-1

Strided convolutions

- Stride convolutions, a variant of convolutional operations in deep learning, skip over specific input values, enhancing computational efficiency.
- Although there's terminology inconsistency, such as using cross-correlation, it usually doesn't significantly impact CNN implementation or understanding.

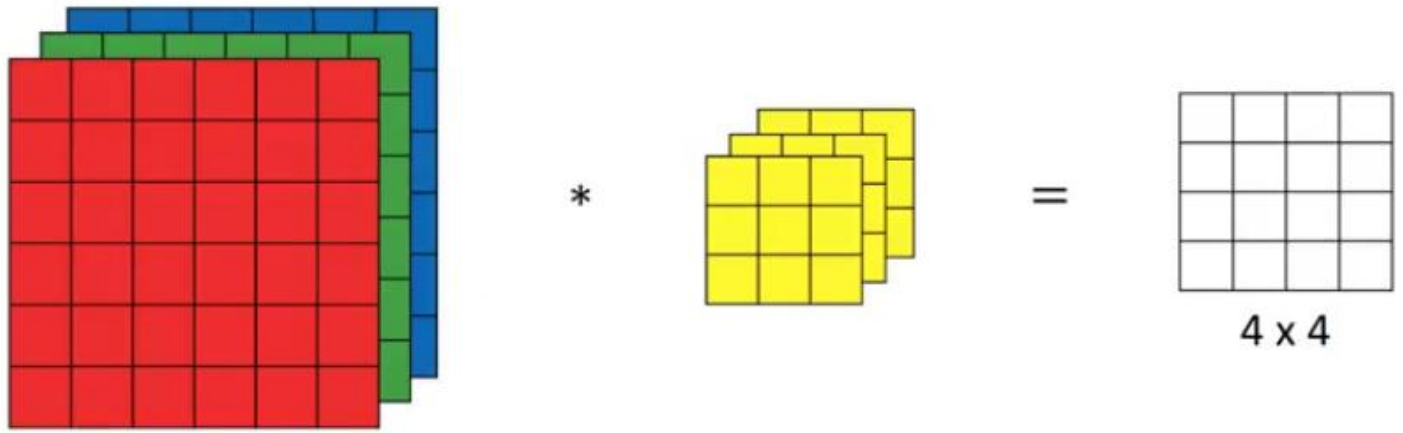
FPT UNIVERSITY

Convolutions over volumes

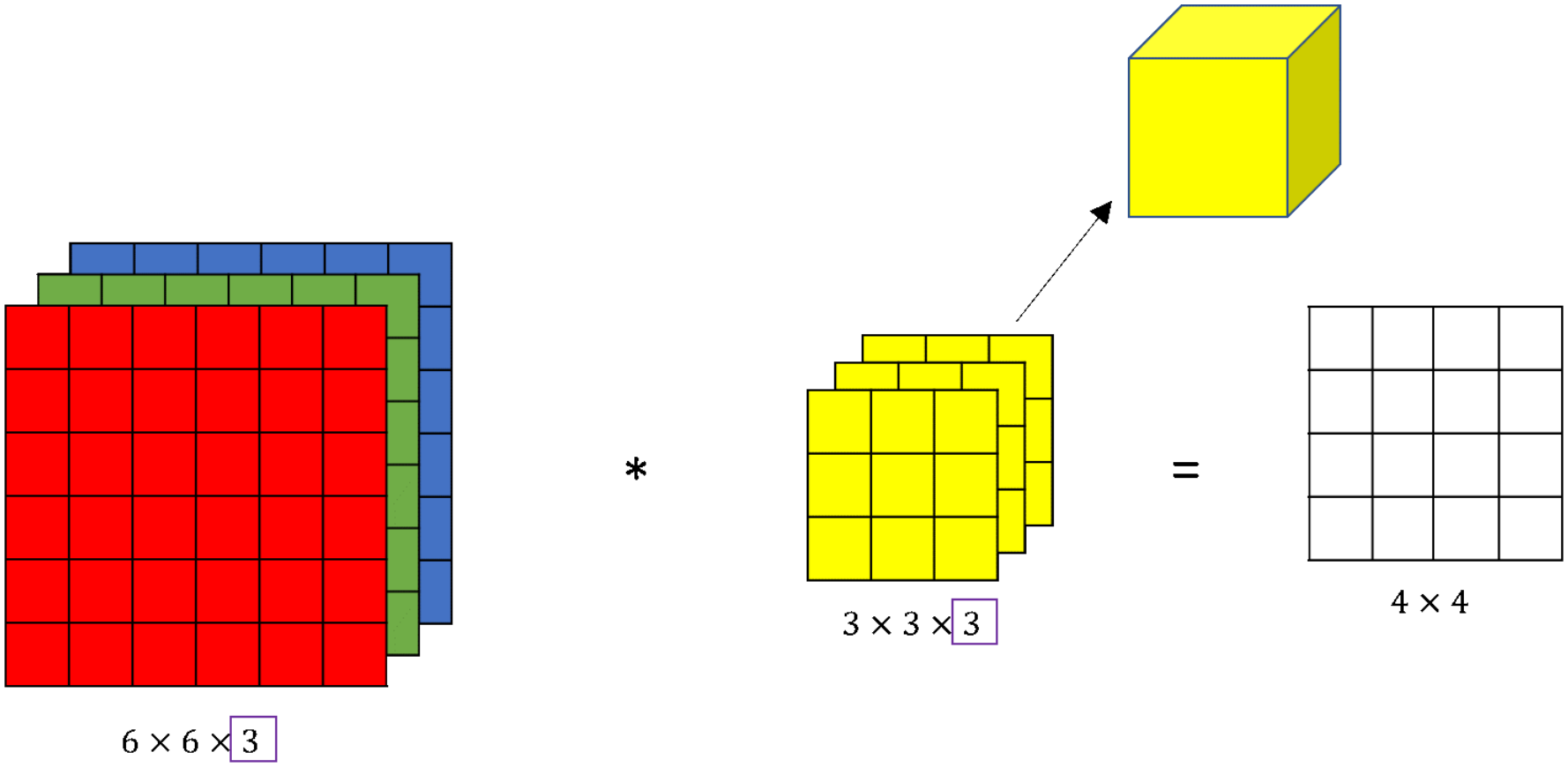
Convolutions on RGB image

FPT Education
FPT UNIVERSITY

- Consider implementing convolutions over 3D volumes, suppose 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 dimensions above represent the height, width and channels in the input and filter.. Keep in mind that the number of channels in the input and filter should be same. This will result in an output of 4 x 4. Let's understand it visually:



Convolutions on RGB image

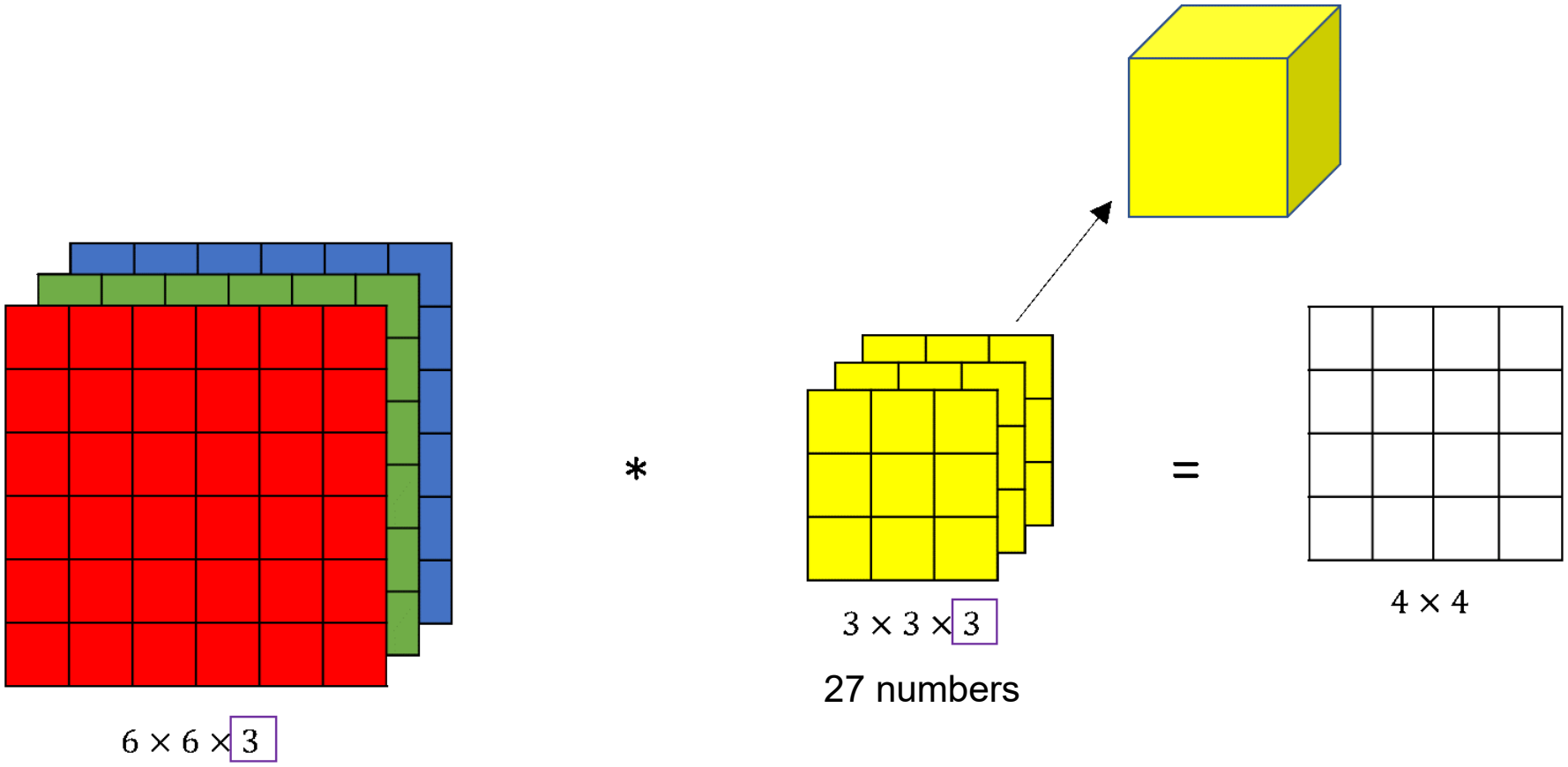


The filter we use we can consider as a volume

Convolutions on RGB image

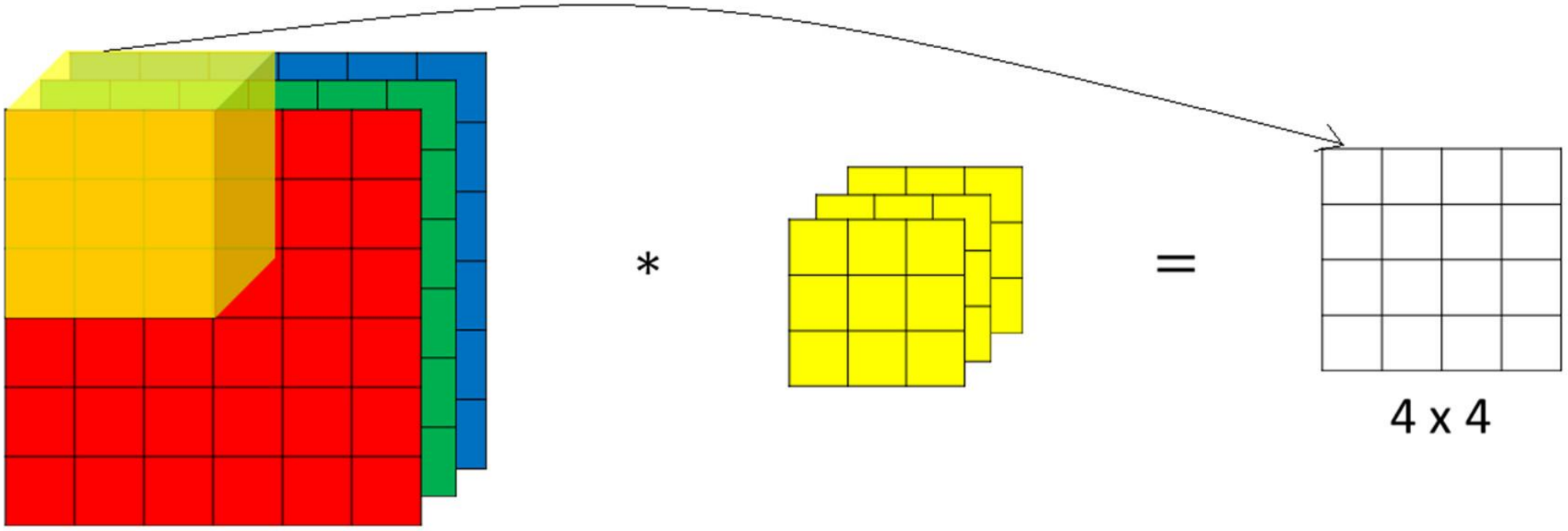
- By sliding the filter over the image and computing the output using the dot product, we can get an output volume.
- Different filters can be used to detect different features in the image, and the outputs of each filter can be stacked to form a 3D volume.

Convolutions on RGB image

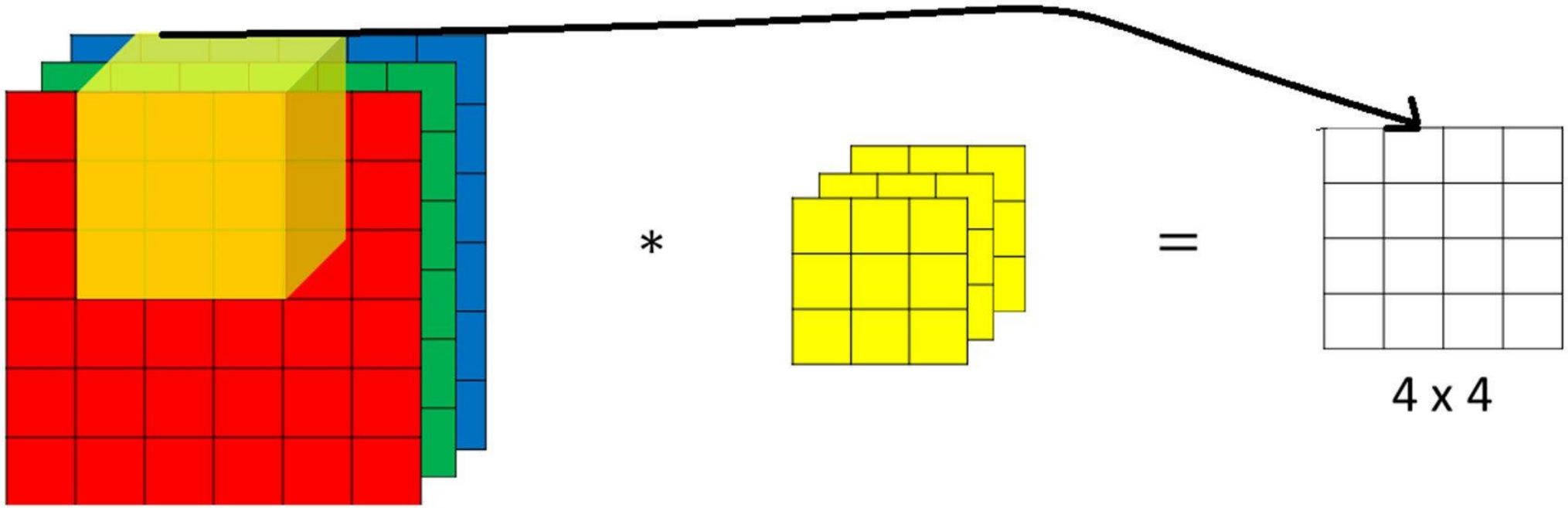


When we apply 3x3x3 filter on the RGB image it is as we implement the volume

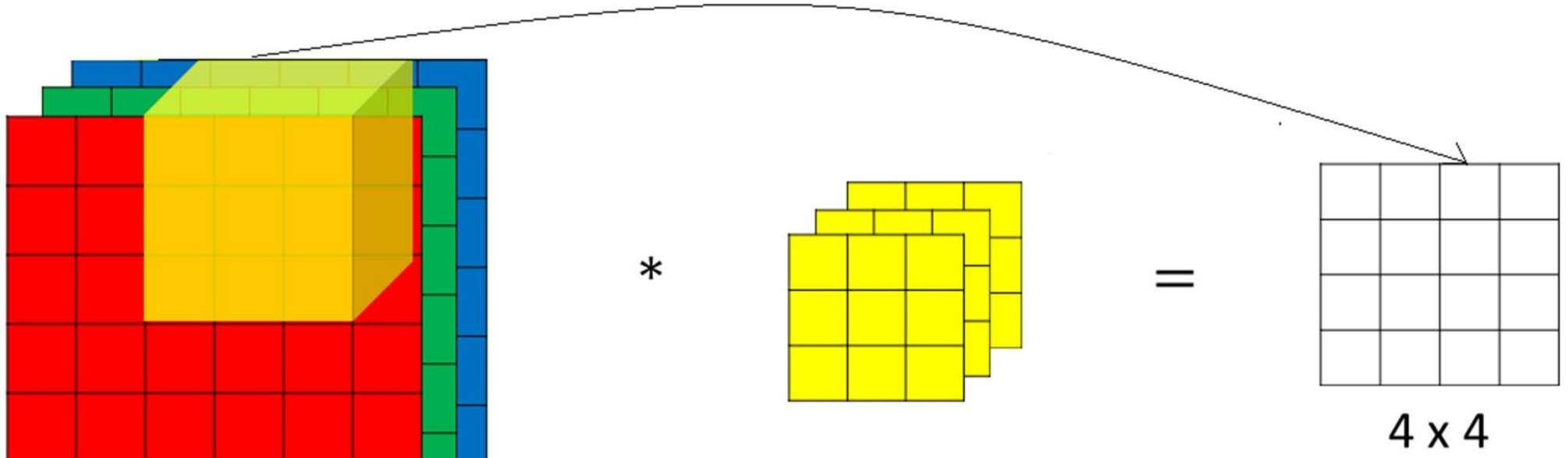
Convolutions on RGB image



Convolutions on RGB image



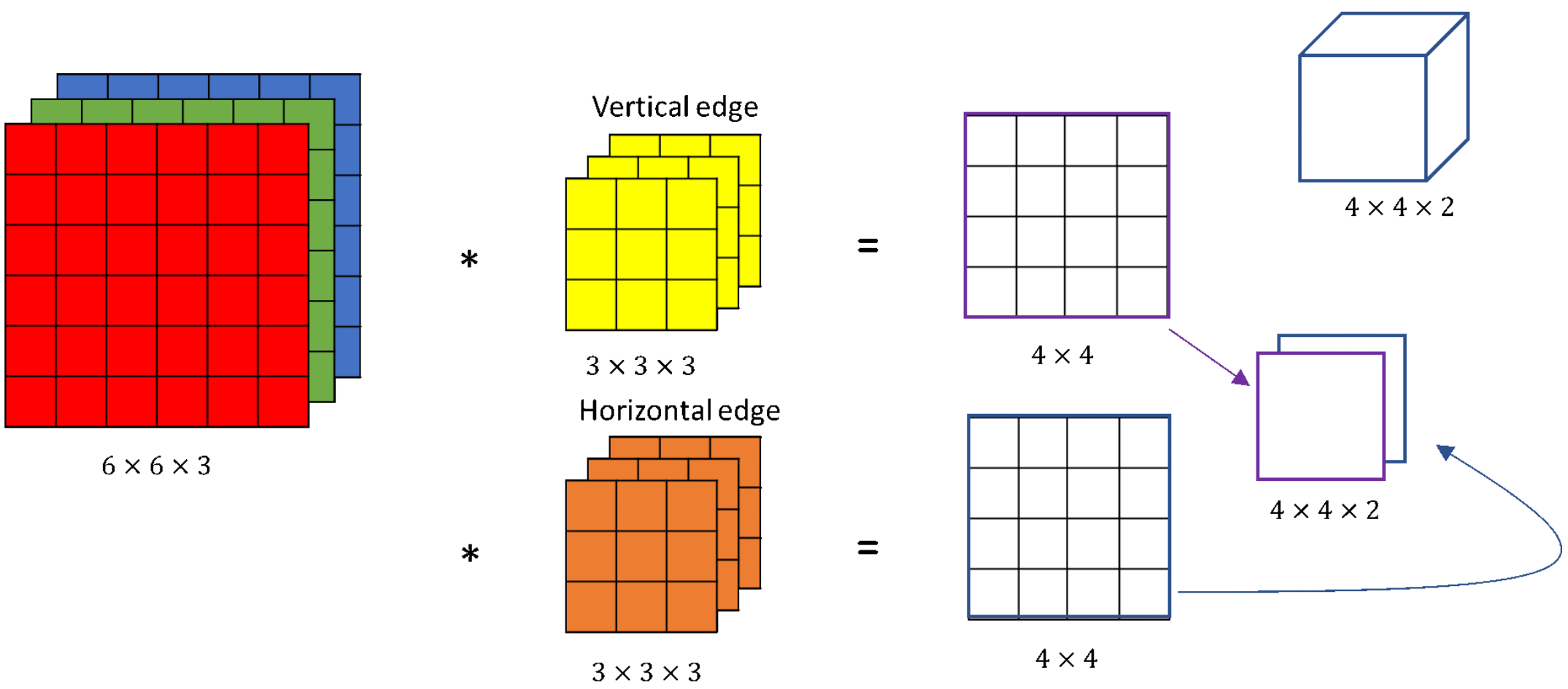
Convolutions on RGB image



Multiple filters

- 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).

Multiple filters



When we convolve with two different filters simultaneously

Multiple filters

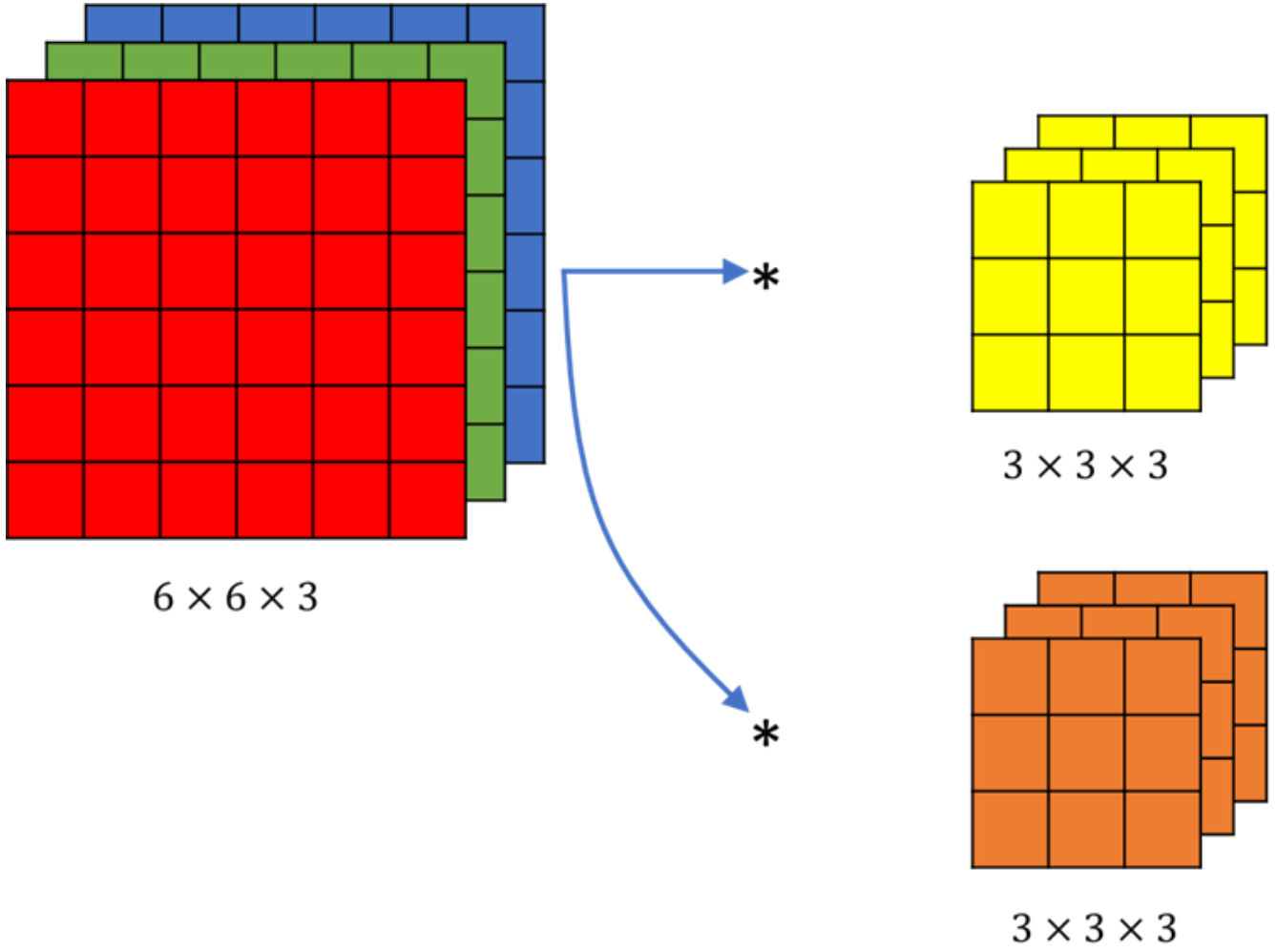
- Generalized dimensions can be given as:
- Input: n x n x $\boldsymbol{n_c}$
- Filter: f x f x $\boldsymbol{n_c}$
- Padding: p
- Stride: s
- Output: ($\frac{\mathbf{n+2p-f}}{\mathbf{s}}$ + 1) x ($\frac{\mathbf{n+2p-f}}{\mathbf{s}}$ + 1) x $\boldsymbol{n_{c'}}$
- Where $\boldsymbol{n_c}$ is the number of channels in the input and filter and $\boldsymbol{n_{c'}}$ is the number of filters.

One layer of a convolutional network

One layer of a convolutional network

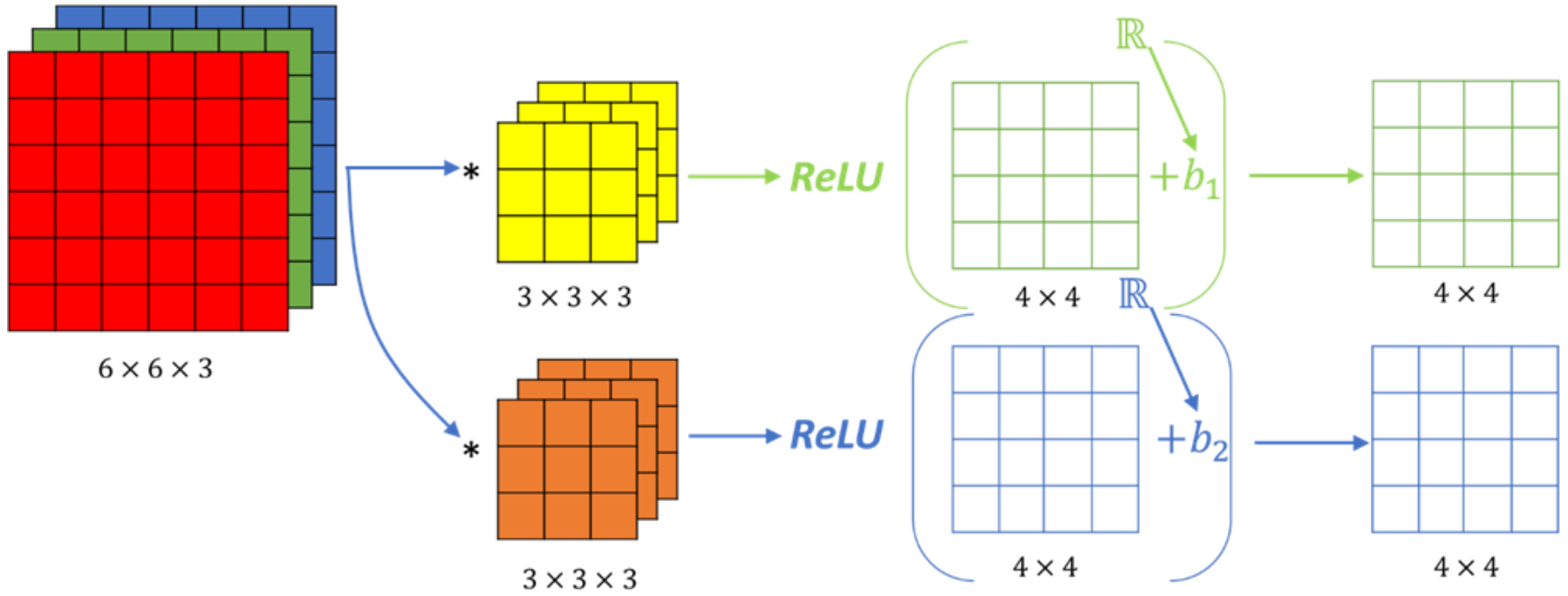
- Building a single layer in a convolutional neural network (CNN) involves convolving a 3D volume with different filters, applying biases and non-linearities, and stacking the 4x4 output to form a 4x4x2 output volume.
- The notation used to describe one layer of a CNN is explained, including filter size, padding, stride, and number of channels. The output volume size is calculated using a formula, and the weights and biases are also explained. There is no universal convention for the ordering of height, width, and channel, but the video follows the convention of listing height and width first and then the number of channels last. Understanding one layer of a CNN is essential for building a deeper CNN.

Example of a layer



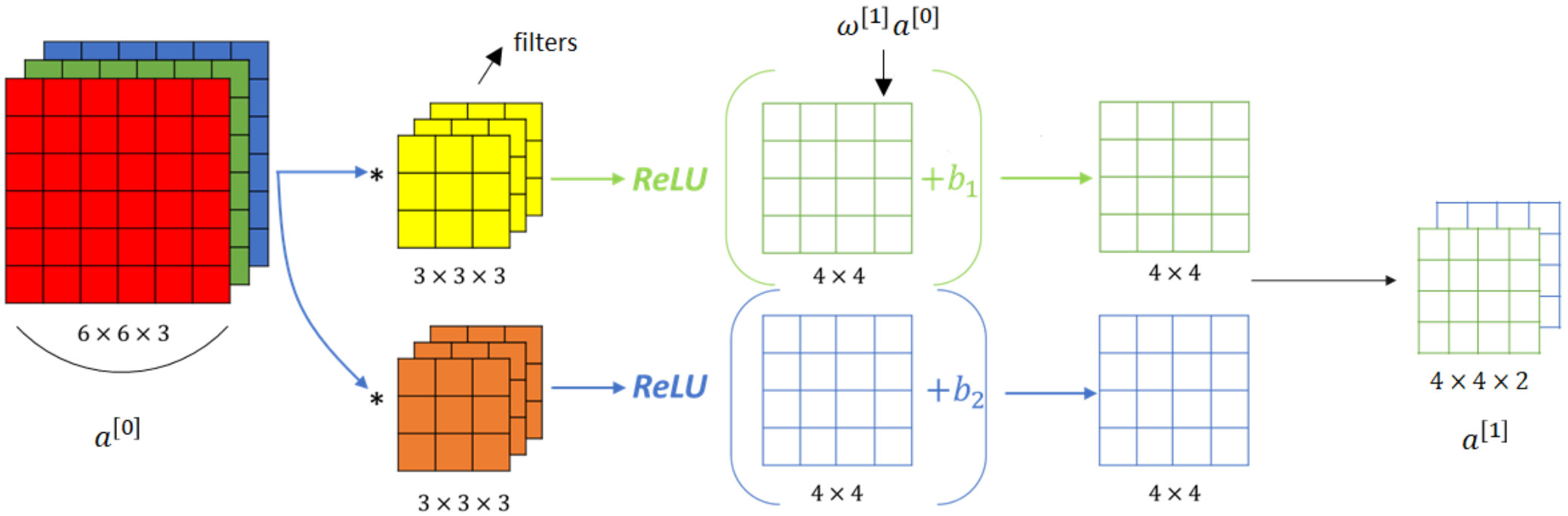
Example of a convolution with two different filters

Example of a layer



The result of convolution with two filters

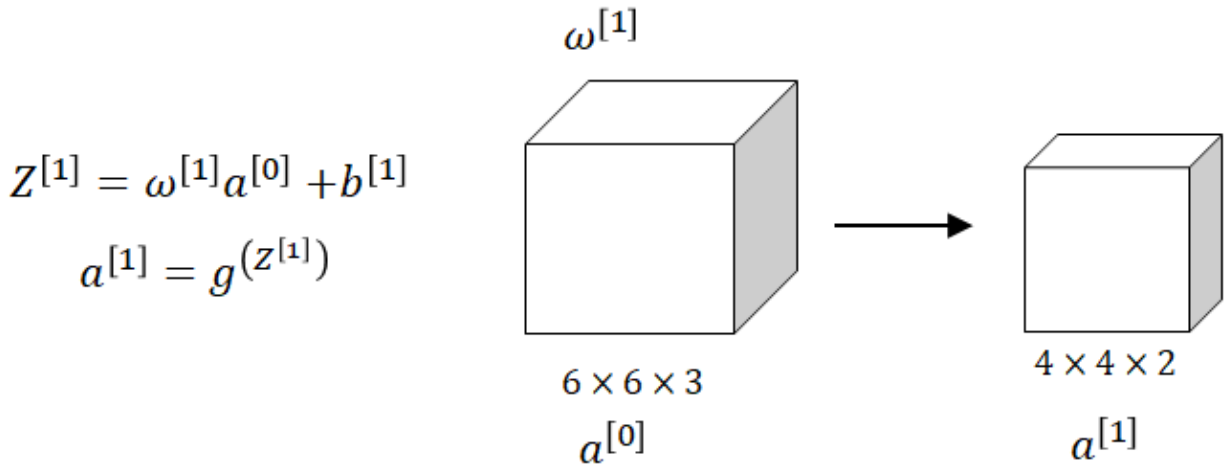
Example of a layer



The result of a convolution of 6x6x3 with two 3x3x3 is a volume of dimension 4x4x2

One layer of a convolutional network

- In neural networks one step of a forward propagation step was:
- Z[1] = W[1] a[0] + b[1] ,
- where a[0] = x.
- Then we applied the non-linearity to get:
- a[1] = $g^{(z^{[1]})}$.
- The same ide[illegible] l Network.



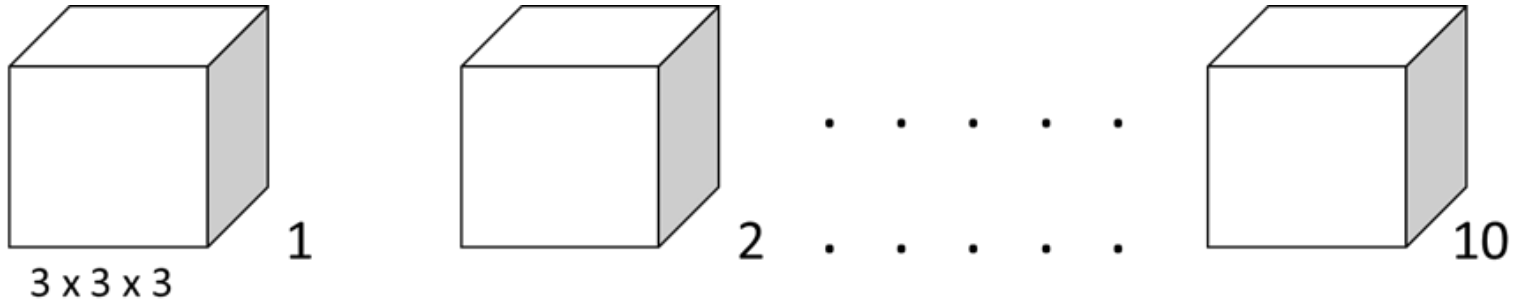
A convolutional layer

Example of a layer

- Compare terms used in Neural Networks with the one that we use in Convolutional Neural Networks:
- In Convolutional Neural Networks (CNNs), the convolution operation involves linear operations, bias addition, and ReLU activation. For a given 6×6×3 input volume and two filters analogous to W[1], the computation results in a 4×4×2 output volume. This process includes applying:
- a linear operation
- adding biases
- applying ReLU activation.
- The transition from a[0] to a[1] represents the transformation from a 6×6×3 input to a 4×4×2 output in one layer of the convolutional network.
- Each filter contributes to the output. If we had 10 filters instead of 2, the output becomes a 4×4×10 dimensional volume, representing a[1]

Number of parameters in one layer

- If you have 10 filters that are 3 x 3 x 3 in one layer of a neural network, how many parameters does this layer have?



27 parameters
+ bias
= 28 parameters

280 parameters

Notation

- For layer l, the filter size, padding, stride are denoted as $f^{[l]}, p^{[l]}, s^{[l]}$ respectively.
- The dimensions of the input image: $n_H^{[l-1]} \times n_W^{[l-1]} \times n_C^{[l-1]}$
- Dimensions of the output image: $n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$
  where

$$n_H^{[l]} = \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1$$

$$n_W^{[l]} = \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1$$

  and $n_C^{[l]}$ is the number of filters used in layer l.
- Size of each filter: $f^{[l]} \times f^{[l]} \times n_C^{[l-1]}$ (because the number of channels of the filter must match that of the input image)
- Each weight tensor has the same size as a the filter. Since there are n_C filters, dimensions of the weight tensor $w^{[l]}$ in layer l is $f^{[l]} \times f^{[l]} \times n_C^{[l-1]} \times n_C^{[l]}$, and the bias vector has dimensions $1 \times 1 \times 1 \times n_C$.
- The activation function $a^{[l]}$ applies nonlinearity on all the output images, and as a result has dimensions $m \times n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$ where m is the number of images, that is, the number of training examples.

FPT UNIVERSITY

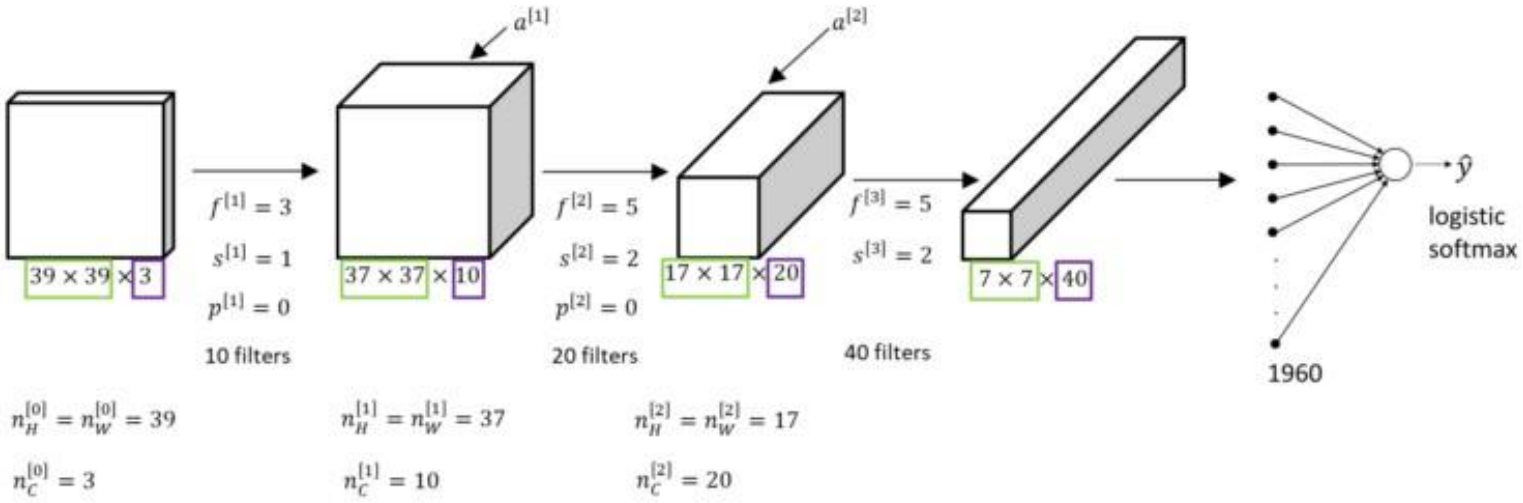
A simple convolution network example

A simple convolution network example

FPT Education
FPT UNIVERSITY

- Consider a concrete example of building a convolutional neural network (ConvNet) for image classification or recognition. The example uses a small image of size 39 x 39 x 3, where 3 represents the number of channels in the image (RGB).
  - The first layer uses 3 x 3 filters with a stride of 1 and no padding, and 10 filters are used in this layer. Using this configuration, the output size of this layer becomes 37 x 37 x 10.
  - The second layer uses 5 x 5 filters with a stride of 2 and no padding, and 20 filters are used in this layer. The output size of this layer becomes 17 x 17 x 20.
  - The third and final layer uses 5 x 5 filters with a stride of 2 and no padding, and 40 filters are used in this layer. The output size of this layer becomes 7 x 7 x 40. The output of this layer is flattened into a vector of size 1,960 and fed into a logistic regression unit or softmax unit for the final prediction.

Example of ConvNet



Types of layer in a convolutional network:

- Convolution
- Pooling
- Fully connected

FPT Education

FPT UNIVERSITY

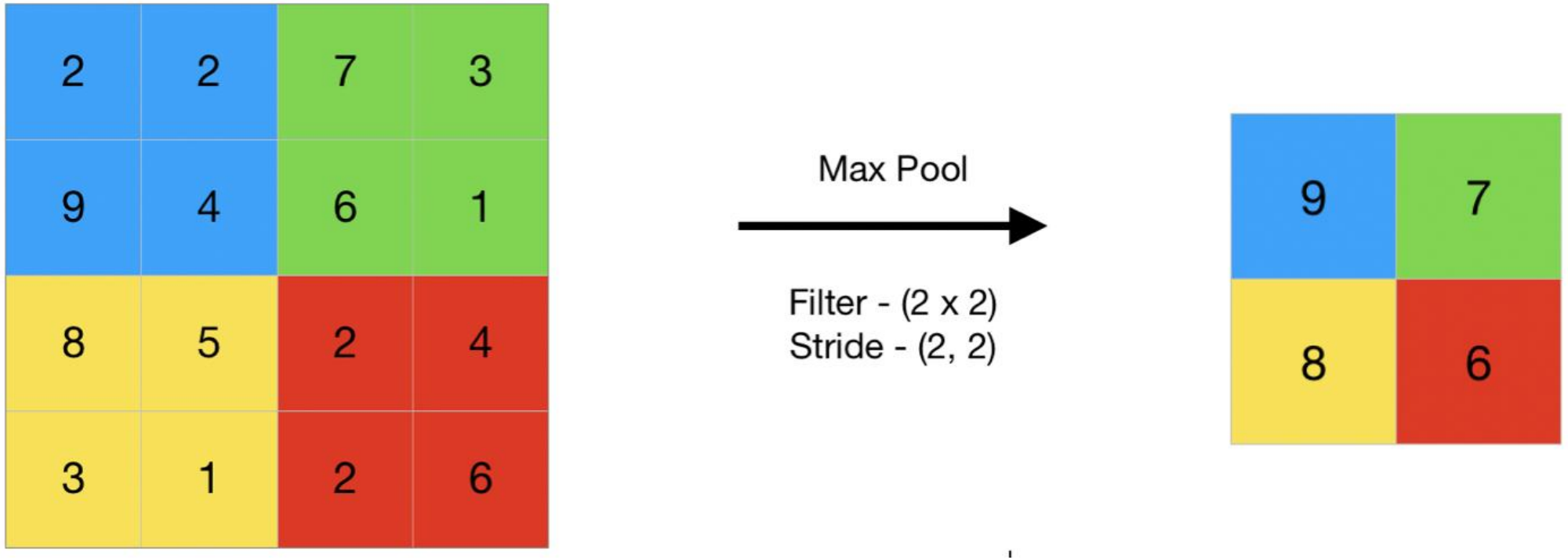
Convolutional Neural Networks

Pooling layers

Pooling layers

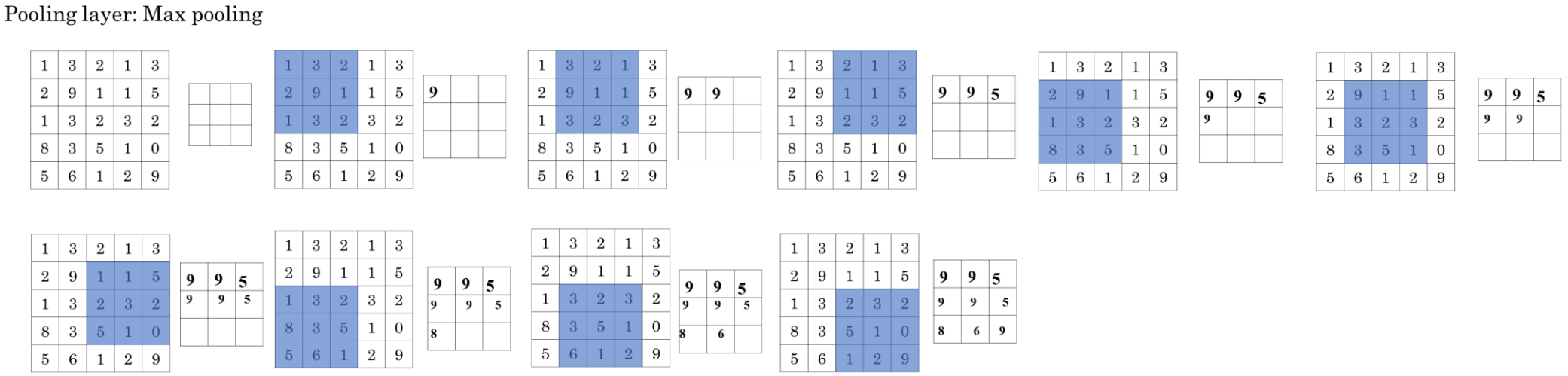
- Pooling layers are used in Convolutional Neural Networks (ConvNets) to reduce the size of the representation, which can speed up computation and make some of the features more robust.
- One common type of pooling is max pooling, where the input is divided into non-overlapping regions, and the maximum value of each region is taken to form the output. The hyperparameters of max pooling are the filter size (f) and the stride (s). The filter size determines the size of the regions that are pooled, and the stride determines the step size when moving to the next region. Max pooling has no parameters to learn, and the computation is fixed once the hyperparameters are set.
- Average pooling is another type of pooling where the average value within a pooling window is taken.

Pooling layers: Max pooling

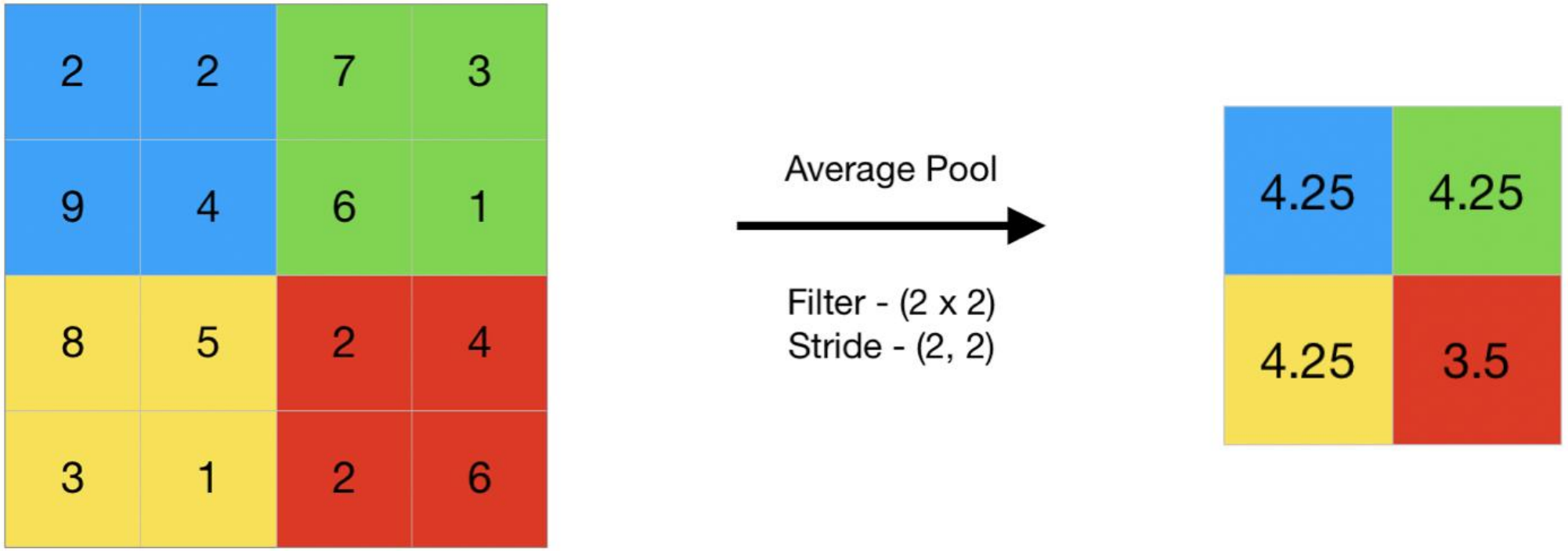


Pooling layers: Max pooling

- Consider an example with a 5x5 input, filter =3 and stride = 1



Pooling layers: Average pooling



Summary of pooling

Hyperparameters:

f : filter size

s : stride

For a feature map having dimensions $\mathbf{n_h}$ **x** $\mathbf{n_w}$ **x** $\mathbf{n_c}$, the dimensions of the output volume size obtained after a pooling layer with filter size **f** and stride **s** is:

$$\left(\frac{n_H - f}{s} + 1\right) \times \left(\frac{n_W - f}{s} + 1\right) \times n_C$$

Advantages of Pooling Layer

- Dimensionality reduction, reducing computational cost, and preventing overfitting.
- Contribute to achieving translation invariance, ensuring consistent feature detection regardless of object position.
- Aids in feature selection, with max pooling emphasizing salient features and average pooling preserving more information.

Disadvantages of Pooling Layer

- Information loss: Pooling Layer discards some information from the input feature maps, which can be important for the final classification or regression task.
- Over-smoothing: Pooling layers can also cause over-smoothing of the feature maps, which can result in the loss of some fine-grained details that are important for the final classification or regression task.
- Hyperparameter tuning: Pooling layers also introduce hyperparameters such as the size of the pooling regions and the stride, which need to be tuned in order to achieve optimal performance. This can be time-consuming and requires some expertise in model building.

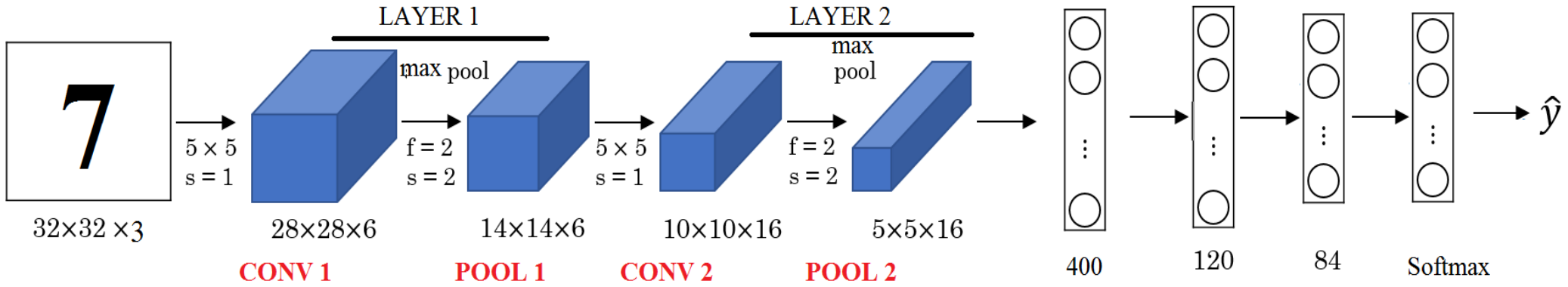
FPT Education

FPT UNIVERSITY

Convolutional neural network example

Convolutional neural network example

- Consider an example of building a convolutional neural network (CNN) for handwritten digit recognition. The input image is 32x32x3, and the CNN consists of several layers, including convolutions, pooling, and fully connected layers.



Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072	0
CONV1 (f=5, s=1)	(28,28,6)	4,704	456
POOL1	(14,14,6)	1,176	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	2,416
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,120
FC4	(84,1)	84	10,164
Softmax	(10,1)	10	850

Summarization

- CNNs, tailored for grid-like data, excel in tasks like image recognition by learning spatial hierarchies of features. In computer vision, they interpret visual data for tasks like image recognition and object detection.
- Edge detection, a key image processing technique, highlights object boundaries. Algorithms like Sobel or Canny enhance edge detection.
- Padding adds border pixels to maintain spatial information during convolutions.
- Strided convolutions skip pixels, reducing output volume dimensions.
- CNNs handle multi-dimensional data, like RGB images, with convolutions across all dimensions. A CNN layer involves filters applying convolutions followed by ReLU activation.
- A basic CNN structure includes convolutional layers, activation functions, pooling layers, and fully connected layers.
- Pooling layers reduce spatial dimensions, retaining vital information.
- In a complete CNN, multiple convolutional, pooling, and fully connected layers form a deep learning architecture for tasks like image classification.

Question

- 1. What kind of edges does the filter detect when applied to a grayscale image?
- 2. If a 300×300 RGB image is input to a fully connected layer with 100 neurons, how many parameters are in this hidden layer (including bias)?
- 3. If you apply 100 convolutional filters of size 5×5 to a 300×300 RGB image, how many parameters (including biases) does this layer have?
- 4. What is the output volume when convolving a 63×63×16 input with 32 filters of size 7×7, stride 2, and no padding?
- 5. If a 15×15×8 input volume is padded with 2 pixels, what is the resulting volume size?
- 6. What padding should you use to perform a “same” convolution on a 63×63×16 input with 7×7 filters and stride 1?
- 7. What is the output volume when applying 2×2 max pooling with stride 2 to a 32×32×16 input?
- 8. True or false: Pooling layers do not affect backpropagation because they have no parameters.
- 9. Name two advantages of parameter sharing in ConvNets.
- 10. What does “sparsity of connections” mean in the context of convolutional layers?
- 11. How many parameters are there in a convolutional layer with 128 filters of size 3×3 applied to a 256×256 grayscale image (including bias)?
- 13. What is the output volume when a 127×127×16 input is convolved with 32 filters of size 5×5, stride 2, and no padding?
- 15. What is the size of the volume after padding a 31×31×32 input with pad = 1?
- 17. What are two hyperparameters of a pooling layer?
- 19. List two benefits of using convolutional layers.
- 20. When a convolution is done with stride 1, on which input pixels does a specific output pixel depend?