from neurons in the lower layers having a smaller receptive field. CNN learns a hierarchy of increasingly complex features from the input data as it flows through the network.

In CNN, the neurons (or filters) in the convolutional layer are not all connected to the pixels in the input image as we have in the dense multilayer perceptron. Hence, a CNN is also called a sparse neural network.

A distinct advantage of CNN over MLP is the reduced number of weights needed for training the network.

Convolutional neural networks are composed of three fundamental types of layers:

- Convolutional layer
- Pooling layer
- Fully connected layer

The Convolutional Layer

The convolution layer is made up of filters and feature maps. A filter is passed over the input image pixels to capture a specific set of features in a process called convolution (see Figure 35-5). The output of a filter is called a feature map.

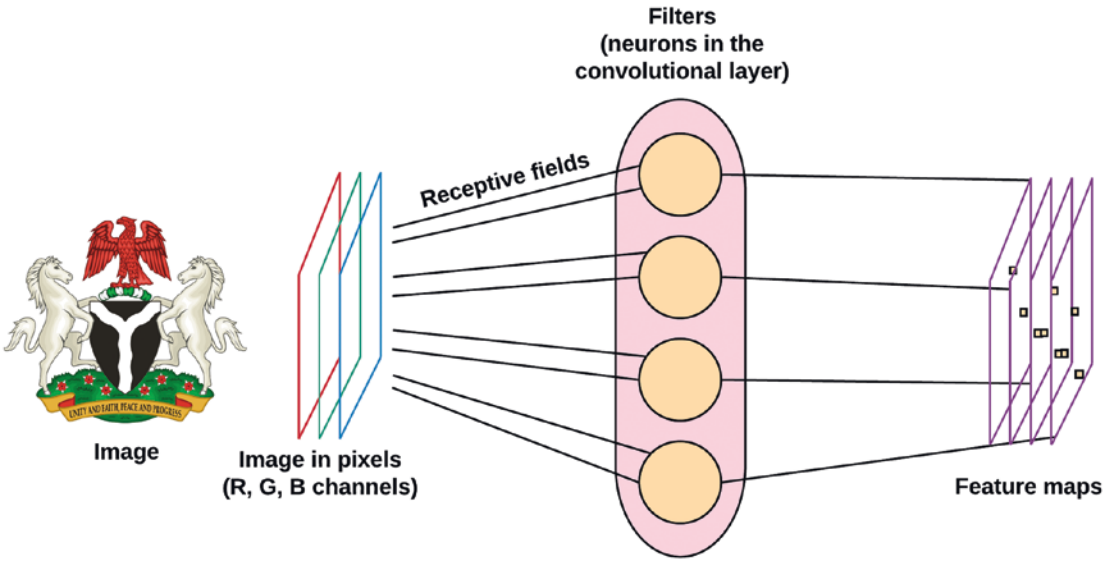

Figure 35-5. *The convolution process*

Convolution

Convolution is the process by which a function is applied to a matrix to extract specific information from the matrix. The function is implemented as a sliding window through the matrix, and it is more popularly called a convolutional filter or a kernel. Both terms are used interchangeably in the literature. The image in Figure 35-6 illustrates a filter sliding through a matrix to extract information from it.

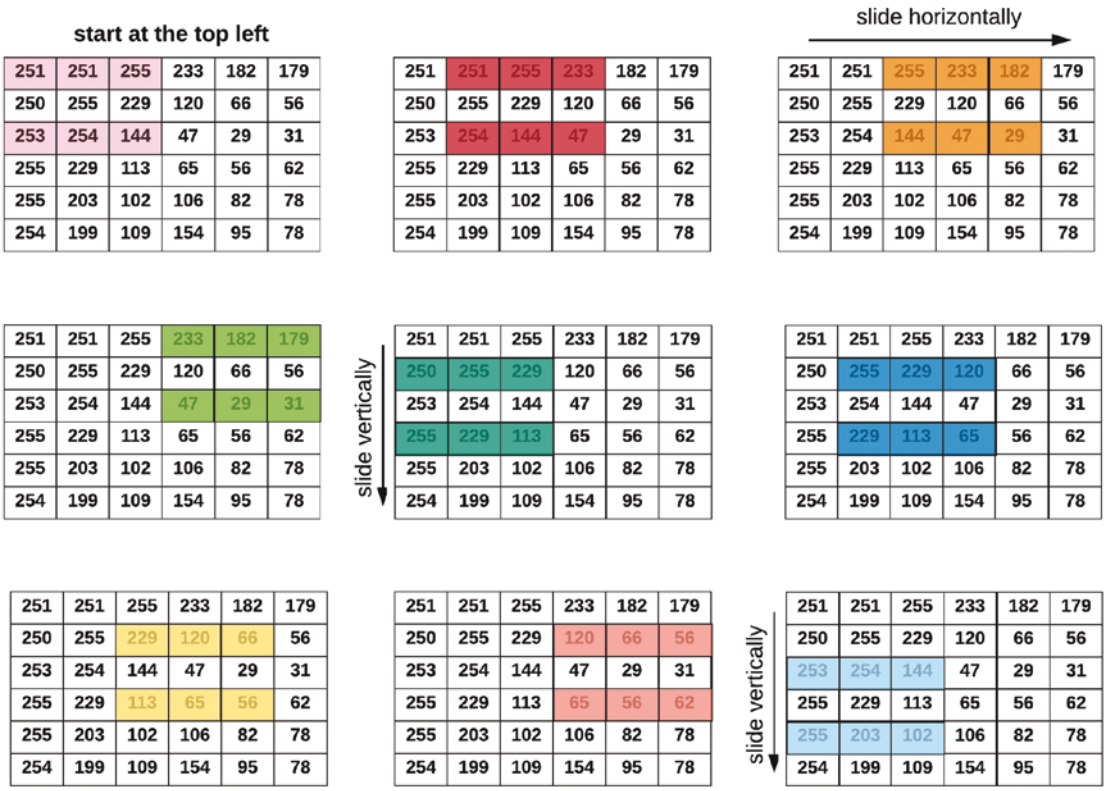

Figure 35-6. *Sliding window of a filter through a matrix*

Filters are neurons in the convolutional layer. They are assigned weights and are applied as a sliding window through the matrix. The output of a filter is a feature map. Filters which are basically neurons also have a non-linear activation function.

The inputs into a filter can be the matrix of the image pixels if the filter is at the input layer, or it can be the feature maps of a previous convolutional layer if the filter is applied at a deeper layer in the network.

Filters are assigned a fixed square block for its input size. This input size can also be seen as the local receptive field of the filter. A common input size for filters is a 3 x 3 square patch as illustrated in Figure 35-7; other standard sizes include a 5 x 5 or 7 x 7 filter for extracting features from images. It is also a best practice to use more filters at deeper layers of the network and fewer filter at the input layer.

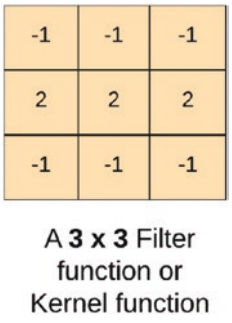

Figure 35-7. *An example of a 3 x 3 filter kernel*

Observe that each cell in the filter has an associated weight or value. These values are used to multiply its associated pixel intensities and then sum up their results, which are filled in the appropriate cell of the convolutional result. This procedure is illustrated in Figure 35-8.

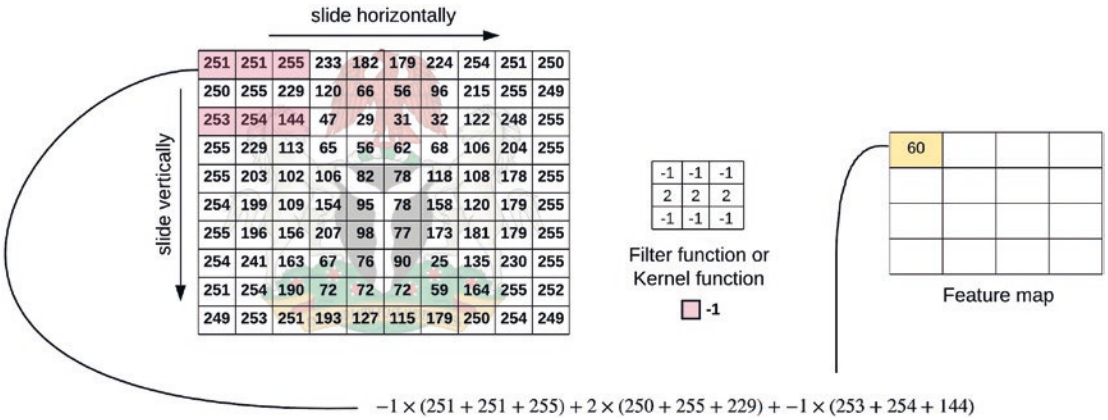

Figure 35-8. *Sliding a convolutional filter across an image matrix to extract features*

The weights on the filter determine the filter operation and consequently the type of features that are extracted from the filter inputs. Different filters are responsible for edge detection, line detection, and so on. See Figure 35-9.

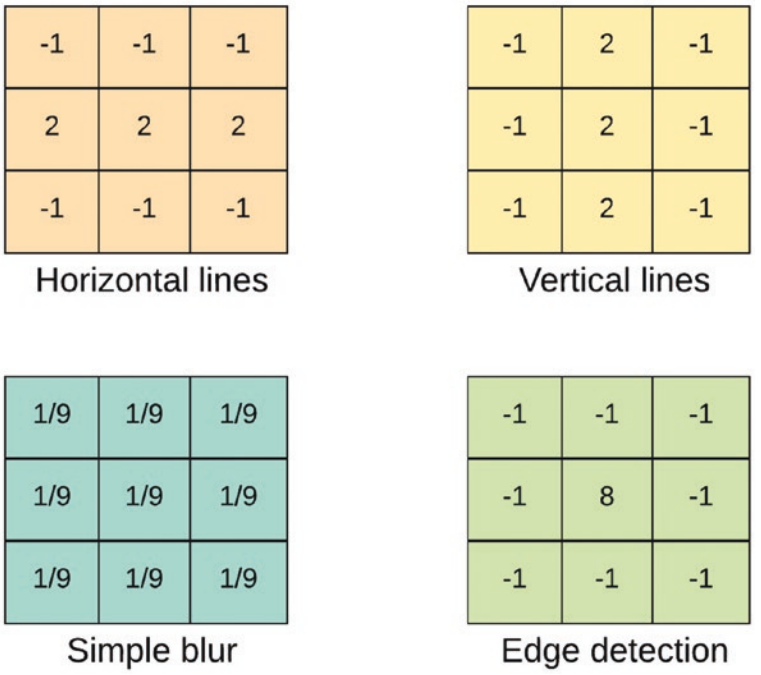

Figure 35-9. *Filter types*

Key considerations to make when designing a convolutional layer are

- The filter size
- The stride of the filter
- The padding for the layer input

The *stride* of the filter determines how many pixel steps the filter makes when moving from one image activation to another. It is typical to use a stride of 1, although this could be increased for large images. See Figure 35-10.

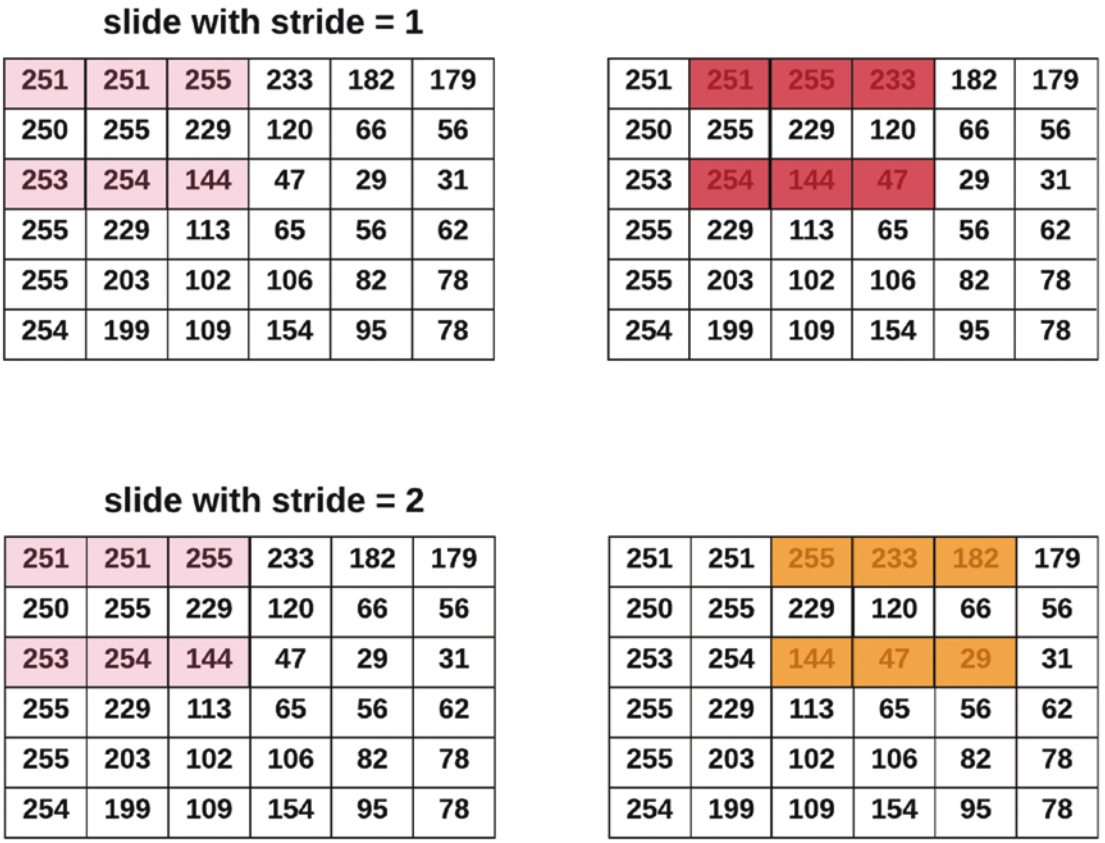

Figure 35-10. *An illustration of stride width*

Sometimes the choice of our filter size and the selected stride may not evenly divide up the size of the input to the filter. So to avoid losing pixel information since we don't slide past the edge of the image, a technique called *zero padding* is employed to pad the borders of the image pixels with a defined layer of zeros. This allows the filter to stride evenly through all the pixels in the image by including the zeros in the convolution. See Figure 35-11.

without zero padding

251	251	255	233	182	179
250	255	229	120	66	56
253	254	144	47	29	31
255	229	113	65	56	62
255	203	102	106	82	78
254	199	109	154	95	78

with zero padding

0	0	0	0	0	0	0	0
0	251	251	255	233	182	179	0
0	250	255	229	120	66	56	0
0	253	254	144	47	29	31	0
0	255	229	113	65	56	62	0
0	255	203	102	106	82	78	0
0	254	199	109	154	95	78	0
0	0	0	0	0	0	0	0

Figure 35-11. *An illustration of zero padding*

Feature Maps

Feature maps are the outputs of a filter in a convolutional layer. Feature maps bring to the fore certain patterns of the input image such as horizontal lines, vertical lines, edges, and so on. These feature maps of the various neurons stacked together are what forms a convolutional neural layer and enable the layer to learn complex patterns and features of an image.

Moving deeper across the convolutional neural network, the inputs to a deeper convolutional layer are the feature maps of the previous layer. See Figure 35-12.

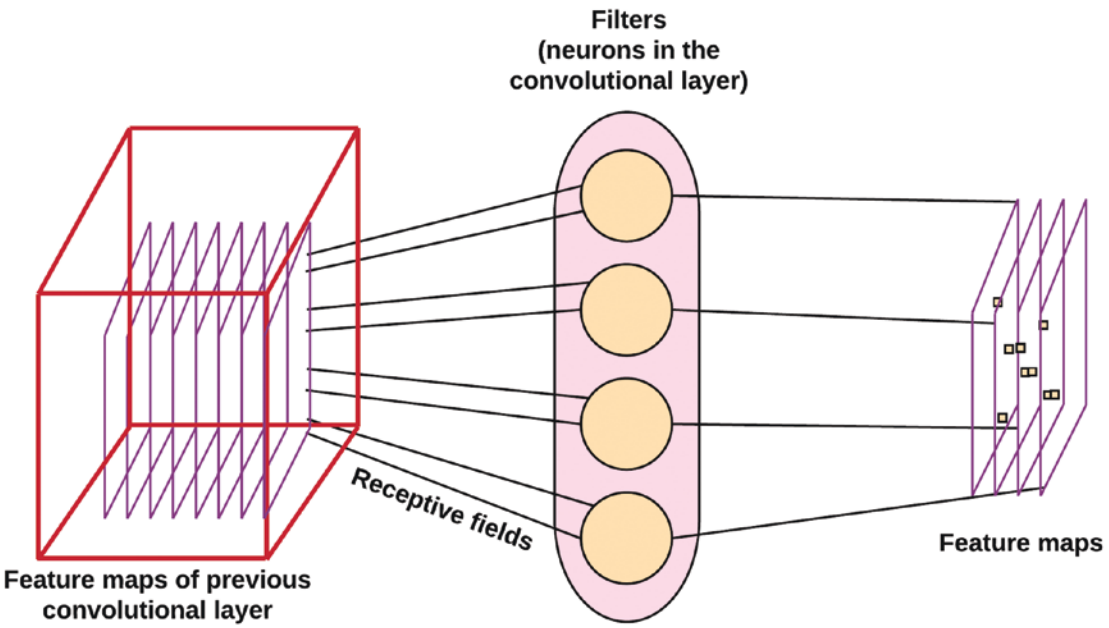

Figure 35-12. *Feature maps as inputs to a convolutional layer*

The Pooling Layer

Pooling layers typically follow one or more convolutional layers. The goal of the pooling layer is to reduce or downsample the feature map of the convolutional layer. The pooling layer summarizes the image features learned in the previous network layers. By doing so, it also helps prevent the network from overfitting. Moreso, the reduction in the input size also bodes well for processing and memory costs when training the network.

The pooling layer can be seen as an aggregation function that consolidates learned features and extracts the essential features from previous layers. It does not conduct any multiplicative transformation on the input feature maps as seen in the convolutional layer.

The aggregation functions carried out by the pooling layer include max, sum, and average. The most frequently used aggregation function in practice is the max and is commonly called the MaxPool.

The aggregation functions of the pooling layer serve as the layers' filters. Just like the filters of the convolutional layer, they have a receptive field (although smaller in size than that of the convolutional layer) and a stride width. Howbeit, the filters which are the neurons of the pooling layer have no weight or biases. A typical size for the pooling filter is a 2 x 2 matrix as shown in Figure 35-13.