**Understanding the Convolutional Filter Operation in CNN’s.**

[[Frederik vl](https://medium.com/@frederik.vl?source=post_page-----c4aad26cf32--------------------------------)](https://medium.com/@frederik.vl?source=post_page-----c4aad26cf32--------------------------------)

[[Advanced Deep Learning](https://medium.com/advanced-deep-learning?source=post_page-----c4aad26cf32--------------------------------)](https://medium.com/advanced-deep-learning?source=post_page-----c4aad26cf32--------------------------------)

[Frederik vl](https://medium.com/@frederik.vl?source=post_page-----c4aad26cf32--------------------------------)

·

[Follow](https://medium.com/m/signin?actionUrl=https%3A%2F%2Fmedium.com%2F_%2Fsubscribe%2Fuser%2F3b95e44a5f7c&operation=register&redirect=https%3A%2F%2Fmedium.com%2Fadvanced-deep-learning%2Fcnn-operation-with-2-kernels-resulting-in-2-feature-mapsunderstanding-the-convolutional-filter-c4aad26cf32&user=Frederik+vl&userId=3b95e44a5f7c&source=post_page-3b95e44a5f7c----c4aad26cf32---------------------post_header-----------)

Published in

[Advanced Deep Learning](https://medium.com/advanced-deep-learning?source=post_page-----c4aad26cf32--------------------------------)

·

5 min read

·

Aug 19

126

2

Convolutional filters, also called kernels are designed to detect specific patterns or features in the input data. For example, in image processing, filters might be designed to detect edges, corners, or textures. In deep learning, the weights of these filters are learned automatically through training on large datasets.

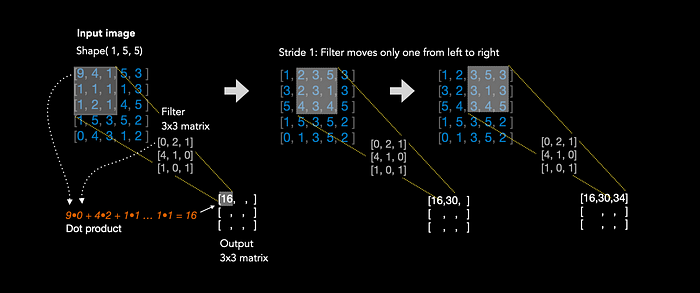


Figure 1: The kernel operation

Figure 1 illustrates the convolutional operation. We have one input matrix of shape (1,5,5), which could be a very small black and white image for example. The pixel values of an image are the features, because each pixel value represents the intensity of light at a given location. We want to learn the relationship between those features in an image in order to detect objects, faces etc. In order to do that, we apply kernels, which are small weight matrices, in this case of size (3,3) to the input. We start in the top left corner and perform the dot product between the overlapping input section and the kernel. In this case: 9•0 + 4•2 + 1•1 + 1•4 + 1•1 + 1•0 + 1•1 + 2•0 + 1•1 = 16. Now, we are done with the first 3x3 pixel values and we move the kernel one to the right and start again, resulting in a value of 30. We repeat the whole operation until the kernel covered every pixel of the input image.

The output matrix is called feature map. Feature maps are not directly interpretable by humans, but they serve as a way for the network to capture and encode the presence of specific features in the input data. As the network learns from data during training, it adapts its filters to recognize features that are relevant to the task at hand, such as classifying objects in images.

**Convolutional operation on coloured RGB images**

An coloured Image in RGB format has the shape (width, height, colour channels), which is sometimes represented in a different order: (colour channels, width, height). Thus, the input image now comes with three colour matrices. The following section explains how the kernel operation is performed in the case of RGB images.

Imagine we apply one convolutional layer with 2 filters of size (3,3) on a coloured input image with 5x5 pixels.

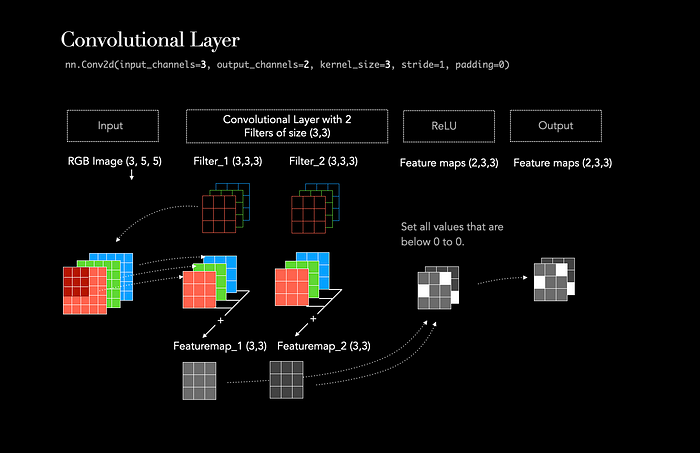
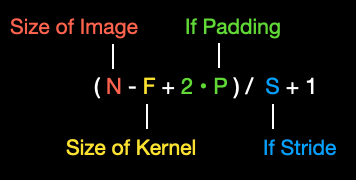


Figure 2: Convolutional Kernel Operation in ML

The illustration above demonstrates the process of applying 2 filters/kernels to one image, resulting in 2 feature maps.

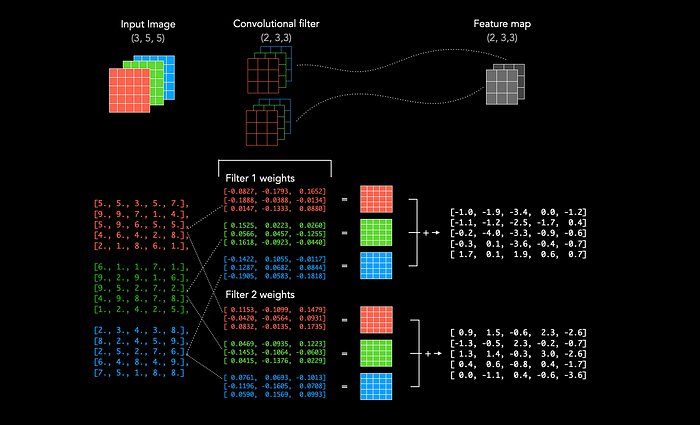
First, each filter is initialized with a random weight matrix for each input channel. In our case, the input image has 3 input channels (Red, Green, Blue), thus each filter comes with three weight matrices. During the operation, all three weight matrices, of each filter, are applied to all three corresponding color channels. This results in three new matrices of size 3x3, called feature maps. The new size of each feature map is calculated through:



Those three feature maps are now added up element-wise, resulting in one single feature map for the first filter. Because we have initalised a second filter, the same process is repeated, but this time with different weights. This combination enhances the network’s ability to recognize complex patterns that may involve interactions between different color channels. Each feature map highlights patterns, edges, or textures. The weights of each filter are learned through backpropagation.

#Same Opertion in Pytorch:  
nn.Conv2d(input\_channels=3, output\_channels=2, kernel\_size=3, stride=1, padding=0)

For better clarity, let’s provide an example with random numbers:

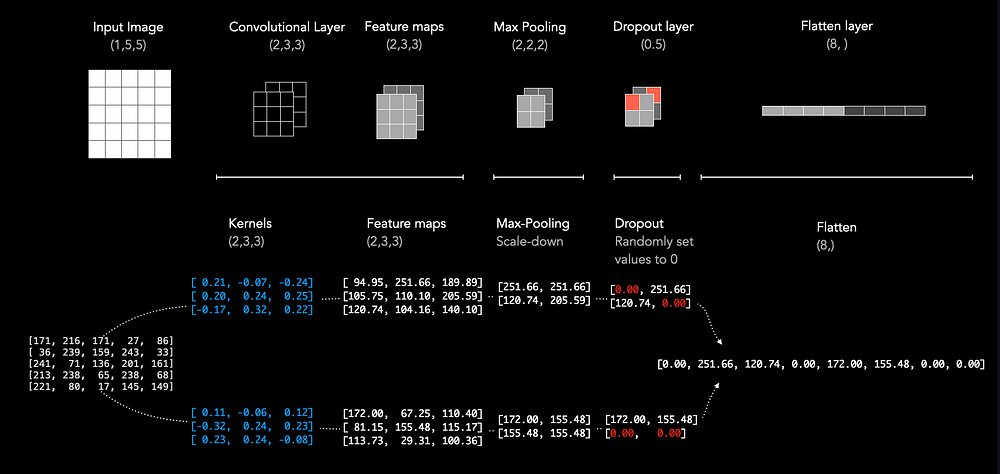


**Understanding the Data Forward-Pass**

Let’s illustrate the data flow in PyTorch or TensorFlow as it passes through the following layers:

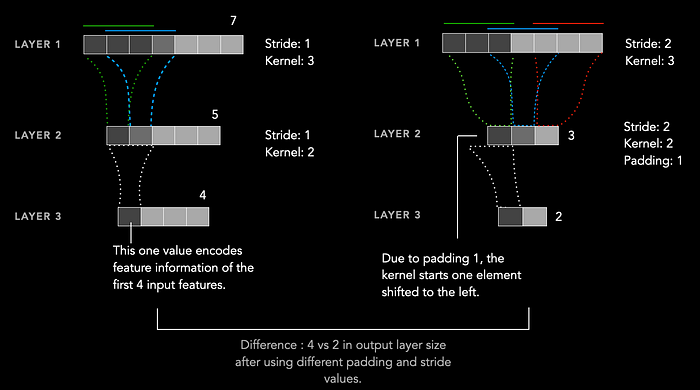
1. Convolutional Layer (2 Kernels, Kernel-size of 3, ReLU activation)
2. Max Pooling Layer
3. Dropout Layer

The illustration below demonstrates how the pixel values of a black and white input image of size (5,5) are transformed through these layers. Initially, both weight matrices (depicted in blue) of the kernels are randomly initialized and applied to the input image. This results in two feature maps, each of size (3,3). The Max Pooling layer then reduces the size of the feature maps to a matrix of (2,2). Finally, the Dropout layer randomly sets each value with a specified probability to zero. In this case, the dropout rate is 50%. To proceed with a fully connected layer, we must flatten the matrices and concatenate them.



**Understanding the Receptive Field of Convolutional Layer**

For large inputs, we need many layers to understand the whole input. We can downsample the features by using stride, kernel\_size and max\_pooling. They increase the receptive field. The receptive field essentially expresses how much information a later layer contains of the first input layer. Consider the example of a 1D array of length 7, where we apply a 1D kernel of size 3. On the left we see that the 1D array length decreases from 7 to 5 in the second layer due to the convolutional operation. The first cell of layer 2 now represents information of the first 3 cells of the first layer.



Receptive Field of CNNs

[Convolutional Network](https://medium.com/tag/convolutional-network?source=post_page-----c4aad26cf32---------------convolutional_network-----------------)

[Cnn](https://medium.com/tag/cnn?source=post_page-----c4aad26cf32---------------cnn-----------------)

[Machine Learning](https://medium.com/tag/machine-learning?source=post_page-----c4aad26cf32---------------machine_learning-----------------)

[Filters](https://medium.com/tag/filters?source=post_page-----c4aad26cf32---------------filters-----------------)

[Kernel](https://medium.com/tag/kernel?source=post_page-----c4aad26cf32---------------kernel-----------------)