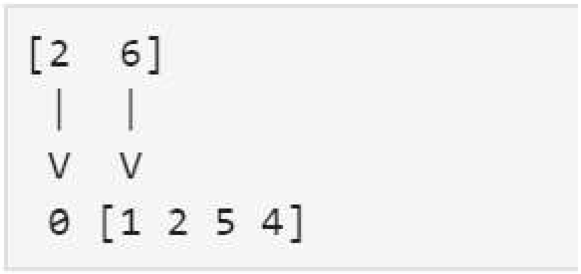
Types of padding in convolution layer

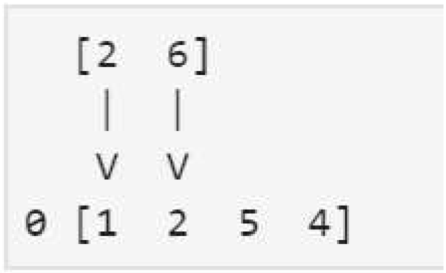
Let’s discuss padding and its types in convolution layers. In convolution layer we have kernels and to make the final filter more informative we use padding in image matrix or any kind of input array. We have three types of padding that are as follows.

1. **Padding Full 😯* Let’s assume a kernel as a sliding window. We have to come with the solution of padding zeros on the input array. This is a very famous implementation and will be easier to show how it works with a simple example, consider x as a filter and h as an input array. x[i] = [6, 2] h[i] = [1, 2, 5, 4] Using the zero padding, we can calculate the convolution. You have to invert the filter x, otherwise the operation would be cross-correlation. First step, (now with zero padding):



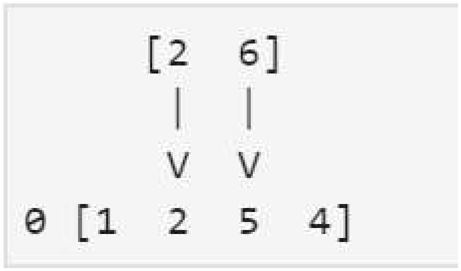
```
[2  6]
 |  |
 V  V
 0 [1 2 5 4]
```

= 2 * 0 + 6 * 1 = 6 Second step:



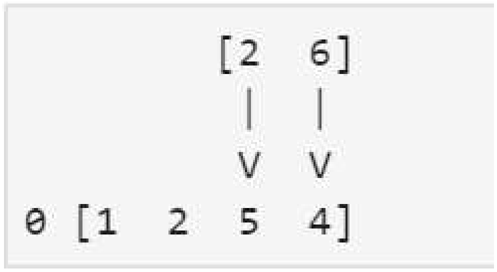
```
   [2  6]
    |  |
    V  V
0 [1  2  5  4]
```

= 2 * 1 + 6 * 2 = 14 Third step:



```
      [2  6]
       |  |
       V  V
0 [1  2  5  4]
```

= 2 * 2 + 6 * 5 = 34 Fourth step:



```
         [2  6]
          |  |
          V  V
0 [1  2  5  4]
```

= 2 * 5 + 6 * 4 = 34 Fifth step: = 2 * 4 + 6 * 0 = 8 The result of the convolution for this case, listing all the steps above, would be: Y = [6 14 34 34 8]

Python3

```
import  numpy as np

x  =  [``6``,  2``]

h  =  [``1``,  2``,  5``,  4``]

y  =  np.convolve(x, h, "full")

print``(y)
```

1. **Padding same 😗* In this type of padding, we only append zero to the left of the array and to the top of the 2D input matrix.

Python3

```
import  numpy as np

x  =  [``6``,  2``]

h  =  [``1``,  2``,  5``,  4``]

y  =  np.convolve(x, h, "same")

print``(y)
```

1. **Padding valid 😗* In this type of padding, we got the reduced output matrix as the size of the output array is reduced. We only applied the kernel when we had a compatible position on the h array, in some cases you want a dimensionality reduction.

Python3

```
import  numpy as np

x  =  [``6``,  2``]

h  =  [``1``,  2``,  5``,  4``]

y  =  np.convolve(x, h, "valid")

print``(y)
```

There are two types of padding in convolution layers:

Valid Padding: In valid padding, no padding is added to the input feature map, and the output feature map is smaller than the input feature map. The convolution operation is performed only on the valid pixels of the input feature map.

Same Padding: In same padding, padding is added to the input feature map such that the size of the output feature map is the same as the input feature map. The number of pixels to be added for padding can be calculated based on the size of the kernel and the desired output feature map size. The convolution operation is performed on the padded input feature map.

The most common padding value is zero-padding, which involves adding zeros to the borders of the input feature map. This helps in reducing the loss of information at the borders of the input feature map and can improve the performance of the model.

In addition to zero-padding, other types of padding, such as reflection padding and replication padding, can also be used. Reflection padding involves reflecting the input feature map along its borders, while replication padding involves replicating the pixels of the input feature map along its borders.

The choice of padding type depends on the specific requirements of the model and the task at hand. In general, same padding is preferred when we want to preserve the spatial dimensions of the feature maps, while valid padding is preferred when we want to reduce the spatial dimensions of the feature maps.

Overall, padding is an important technique in convolutional neural networks that helps in preserving the spatial dimensions of the feature maps and can improve the performance of the model.

Advantages of padding in convolution layers:

1. Preserves spatial information: Padding helps in preserving the spatial dimensions of the feature maps, which can be important in many image processing tasks. Without padding, the spatial dimensions of the feature maps would be reduced after each convolution operation, which could result in the loss of important information at the borders of the input feature map.
2. Improved model performance: Padding can help in improving the performance of the model by reducing the loss of information at the borders of the input feature map. This can result in better accuracy and higher prediction scores.
3. Flexibility: The use of padding provides flexibility in choosing the size of the kernel and the stride. It can allow for the use of larger kernels and/or strides without losing important spatial information.

Disadvantages of padding in convolution layers:

1. Increased computational cost: Padding involves adding extra pixels to the input feature map, which increases the computational cost of the convolution operation. This can result in longer training times and slower inference times.

2. Increased memory usage: The use of padding can increase the memory usage of the model, which can be an issue in resource-constrained environments.
3. Overall, the advantages of padding in convolution layers outweigh the disadvantages in most cases, as it can help in preserving spatial information, improving model performance, and providing flexibility in choosing kernel size and stride. However, the increased computational cost and memory usage should be taken into consideration when designing a convolutional neural network.