CNN | Introduction to Padding

During convolution, the size of the output feature map is determined by the size of the input feature map, the size of the kernel, and the stride. if we simply apply the kernel on the input feature map, then the output feature map will be smaller than the input. This can result in the loss of information at the borders of the input feature map. In Order to preserve the border information we use padding.

What Is Padding

padding is a technique used to preserve the spatial dimensions of the input image after convolution operations on a feature map. Padding involves adding extra pixels around the border of the input feature map before convolution.

This can be done in two ways:

- **Valid Padding**: In the valid padding, no padding is added to the input feature map, and the output feature map is smaller than the input feature map. This is useful when we want to reduce the spatial dimensions of the feature maps.
- **Same Padding**: In the 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. This is useful when we want to preserve the spatial dimensions of the feature maps.

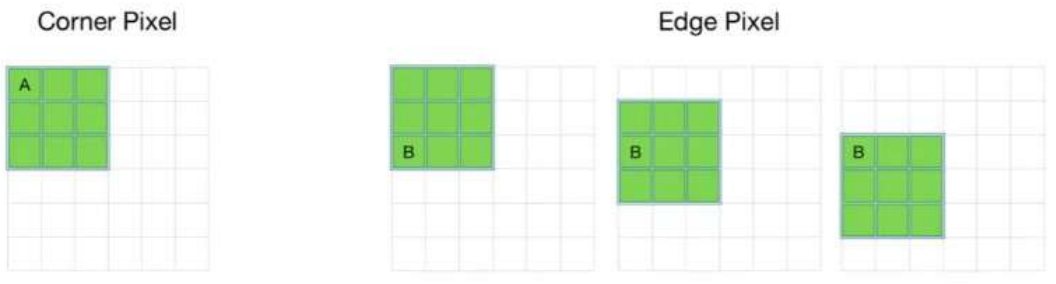
The number of pixels to be added for padding can be calculated based on the size of the kernel and the desired output of the feature map size. The most common padding value is zero-padding, which involves adding zeros to the borders of the input feature map.

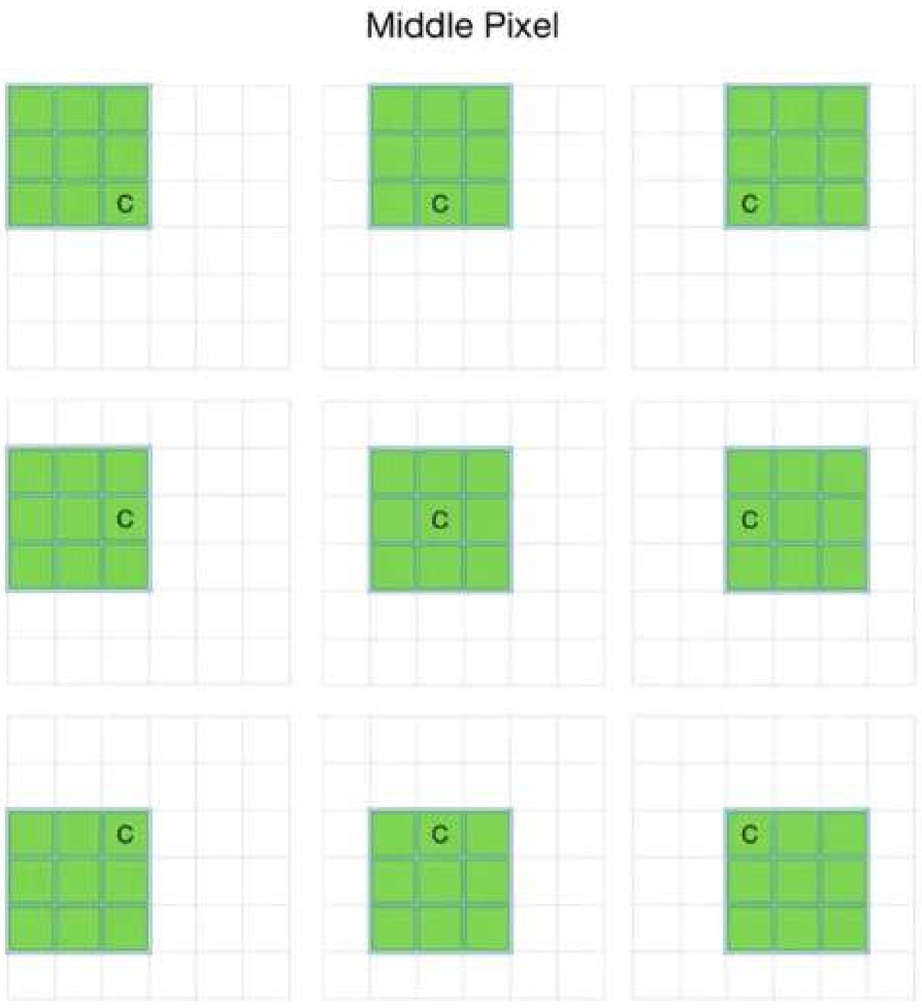
Padding can help in reducing the loss of information at the borders of the input feature map and can improve the performance of the model. However, it also increases the computational cost of the convolution operation. Overall, padding is an important technique in CNNs that helps in preserving the spatial dimensions of the feature maps and can improve the performance of the model.

Problem With Convolution Layers Without Padding

- For a grayscale (n x n) image and (f x f) filter/kernel, the dimensions of the image resulting from a convolution operation is **(n – f + 1) x (n – f + 1)**. For example, for an (8 x 8) image and (3 x 3) filter, the output resulting after the convolution operation would be of size (6 x 6). Thus, the image shrinks every time a convolution operation is performed.
- This places an upper limit to the number of times such an operation could be performed before the image reduces to nothing thereby precluding us from building deeper networks.
- Also, the pixels on the corners and the edges are used much less than those in the middle.

  For example,



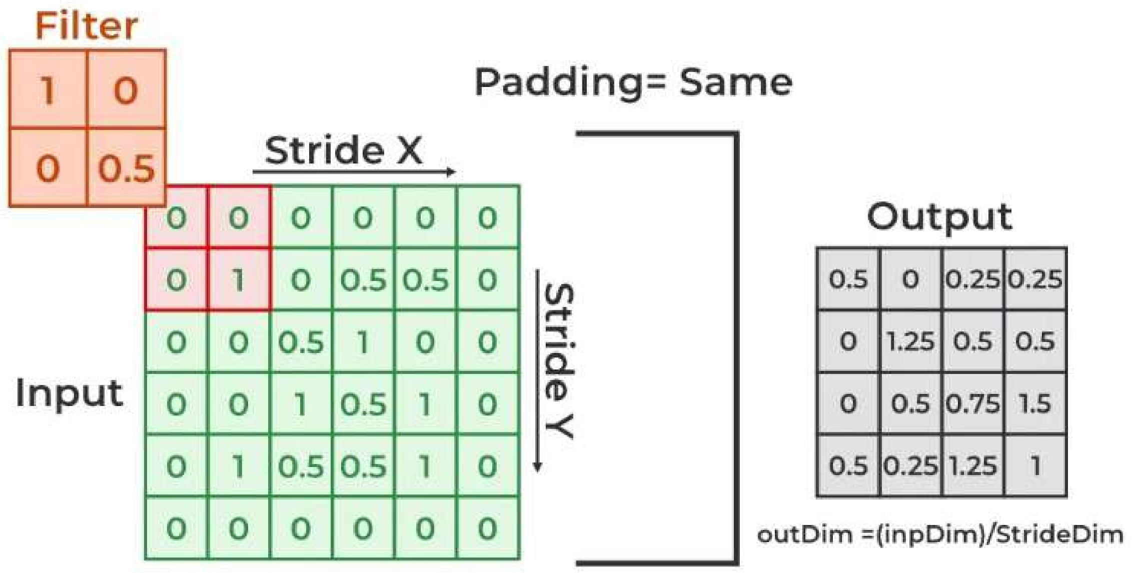


Padding in convulational neural network

- Clearly, pixel A is touched in just one convolution operation and pixel B is touched in 3 convolution operations, while pixel C is touched in 9 convolution operations. In general, pixels in the middle are used more often than pixels on corners and edges. Consequently, the information on the borders of images is not preserved as well as the information in the middle.

Effect Of Padding On Input Images

Padding is simply a process of adding layers of zeros to our input images so as to avoid the problems mentioned above through the following changes to the input image.



padding in convolutional network

Padding prevents the shrinking of the input image.

p *= number of layers of zeros added to the border of the image,*

then ***(n x n)*** *image* **—> *(n + 2p) x (n + 2p)*** *image after padding.*

(n + 2p) x (n + 2p) * (f x f)   —–> outputs (n + 2p – f + 1) x (n + 2p – f + 1) *images*

For example, by adding one layer of padding to an (8 x 8) image and using a (3 x 3) filter we would get an (8 x 8) output after performing a convolution operation.

This increases the contribution of the pixels at the border of the original image by bringing them into the middle of the padded image. Thus, information on the borders is preserved as well as the information in the middle of the image.

Types of Padding

Valid Padding: It implies no padding at all. The input image is left in its valid/unaltered shape. So

where, nxn is the dimension of input image

fxf is kernel size

n-f+1 is output image size

** represents a convolution operation.*

Same Padding: In this case, we add 'p' padding layers such that the output image has the same dimensions as the input image.
So, ***[(n + 2p) x (n + 2p) image] * [(f x f) filter] —> [(n x n) image]***
which gives **p = (f – 1) / 2** (because n + 2p – f + 1 = n).

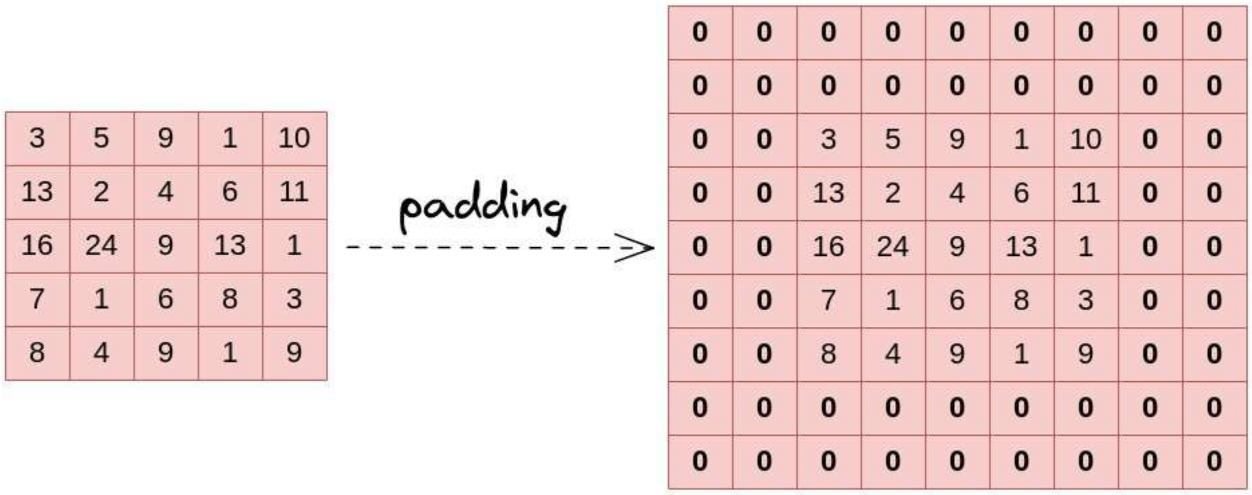
So, if we use a (3 x 3) filter on an input image to get the output with the same dimensions. the 1 layer of zeros must be added to the borders for the same padding. Similarly, if (5 x 5) filter is used 2 layers of zeros must be appended to the border of the image.

Padding

In machine learning and in neural networks, convolutional filters are applied to images in order to extract useful information from them, such as patterns, edges, and corners. Since an image convolves with a filter, the output's dimension is (n-f+1) * (n-f+1).

Due to the convolutional operation, the image shrinks, especially if the neural network is quite deep, with many layers. Furthermore, in contrast to the center pixels, which are engaged in numerous convolutional areas, corner pixels are employed less often and are not as much involved when applying a convolutional operation to a matrix. This causes the edges of the input picture to lose relevant information. Padding comes to deal with these issues.

Padding is a technique employed that guarantees that the input data has the exact size and shape that the model anticipates after every convolutional operation at each stage of the deep learning model. So this is ensured by adding additional elements to the input data, such as zeros, to make it conform to the model's specifications:



3. Different Types of Padding

There are three main types of padding used in machine learning: same padding, valid padding, and causal padding, with the most generally used being the same padding.

Same Padding: This type of padding adds an equal number of elements ( ), typically zeros, to the top, bottom, left, and right sides of the input image so that the output size is the same as the input size.

Valid Padding: This kind of padding computes the input matrix's valid elements rather than adding new ones. Therefore, the output size is less than the input size. Valid padding denotes that there is no padding and that all of the proportions are accurate, resulting in a filter that completely covers the input picture.

Causal Padding: This type of padding is used in sequence-to-sequence models and time series forecasting and basically works with the one-dimensional convolutional layers. Causal padding adds elements to the start of the data, which also aids in forecasting the values of early time steps.

Note that the choice of padding can significantly impact the model's performance and accuracy.

4. Why Is Padding Used?

Padding is one of the most crucial machine learning techniques since it allows for the modification of input size in order to be compatible with the neural network's requirements. It adds elements to the input matrix before any convolutional filter is applied, and thus, it aids in preventing any information loss, particularly from the edges of the image. Moreover, padding helps to prevent the image's compression after every convolutional operation.

Lastly, it aids in maintaining the spatial relationship and characteristics of the input data and can significantly impact the model's overall performance.

5. Advantages and Disadvantages

Padding offers several advantages along with some disadvantages as well.

One of its important benefits is that padding has been shown to improve the model's performance. It helps keep important aspects of the image that would get lost and thus reveals useful relationships that may play significant roles in the output. Furthermore, it aids in maintaining consistent input length by adding extra elements, and it simplifies the data processing in a way that there is no need for additional pre-processing steps to handle variable-length input data.

On the other hand, padding increases the computational complexity of the model as more operations and processing steps are required. In addition, it adds extra elements, and thus the computational cost is increased. Lastly, in some cases, padding has seemed to contribute to over-fitting.