



# Padding, Strides and Pooling


Pre-work: Computer Vision

This file is meant for personal use by bpidugu@hotmail.com only.  
Sharing or publishing the contents in part or full is liable for legal action.



# Agenda

- Drawbacks of Convolutions
- Padding and Strides
- Pooling

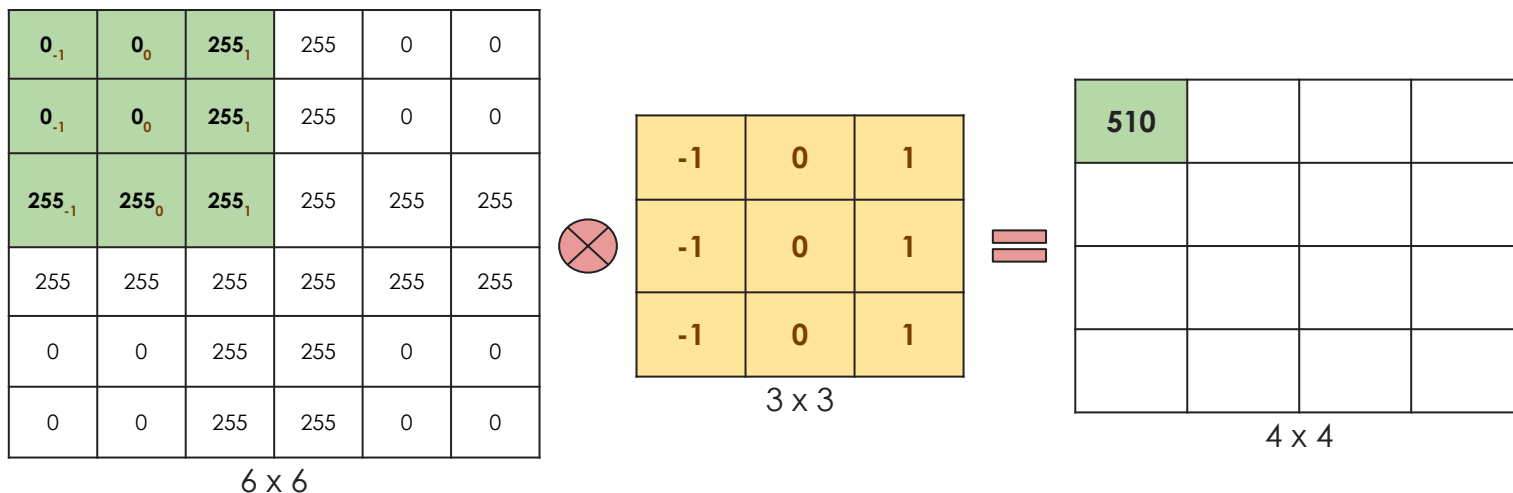


# Drawbacks of Convolutions

This file is meant for personal use by bpidugu@hotmail.com only.  
Sharing or publishing the contents in part or full is liable for legal action.

# Recap - The Convolution operation

- We have now understood **the mathematical working of the Convolution Operation**, which is a sum of the element-wise products between the input and the filter. This is carried out through a sliding mechanism in order to generate each number in the output feature map.

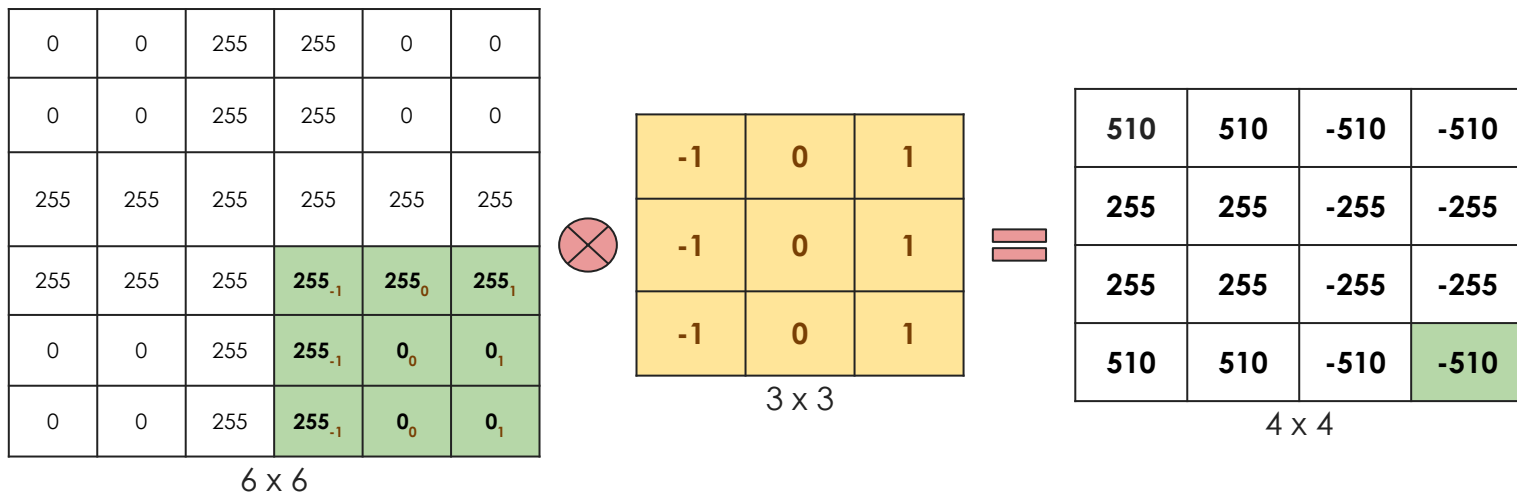


$$(0 \times -1) + (0 \times 0) + (255 \times 1) + (0 \times -1) + (0 \times 0) + (255 \times 1) + (255 \times -1) + (255 \times 0) + (255 \times 1) = 510$$

This file is meant for personal use by bpidugu@hotmail.com only.  
Sharing or publishing the contents in part or full is liable for legal action.

# Recap - The Convolution operation

- We have now understood **the mathematical working of the Convolution operation**, which is a sum of the element-wise products between the input and the filter. This is carried out through a sliding mechanism in order to generate each number in the output feature map.



$$(255 \times -1) + (255 \times 0) + (255 \times 1) + (255 \times -1) + (0 \times 0) + (0 \times 1) + (255 \times -1) + (0 \times 0) + (0 \times 1) = -510$$

# Drawbacks of Convolutions

- This mechanism of the vanilla Convolution Operation has **three significant drawbacks**:
- The first drawback** of Convolution is that **it under-utilizes pixels at the edge of the image**, especially the pixels at the corners, or close to the corners in relatively large images. These pixels may only be utilized a small number of times by the sliding filter, in comparison to pixels closer to the center of the image.  
**For example:** The 0 pixels circled in the corners would only be utilized once by the sliding filter in all its movement across the rows and columns of the image.

0 <sub>-1</sub>	0 <sub>0</sub>	255 <sub>1</sub>	255	0	0	0	0	255	255	0	0
0 <sub>-1</sub>	0 <sub>0</sub>	255 <sub>1</sub>	255	0	0	0	0	255	255	0	0
255 <sub>-1</sub>	255 <sub>0</sub>	255 <sub>1</sub>	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
0	0	255	255	0	0	0	0	255	255	0	0
0	0	255	255	0	0	0	0	255	255	0	0

# Drawbacks of Convolutions

- This drawback of convolutions is a **significant disadvantage** for images where there is **important information in the corners or edges of the image**.

**For example:** In this cropped image of the duck, the vanilla Convolution Operation may perform poorly in classifying the image as a duck, because the crucial information about the shape of the duck's head, which is needed in classifying this image as that of a duck, is only in the corner of the image. Such regions, as we have seen, are poorly utilized by the vanilla convolution operation.

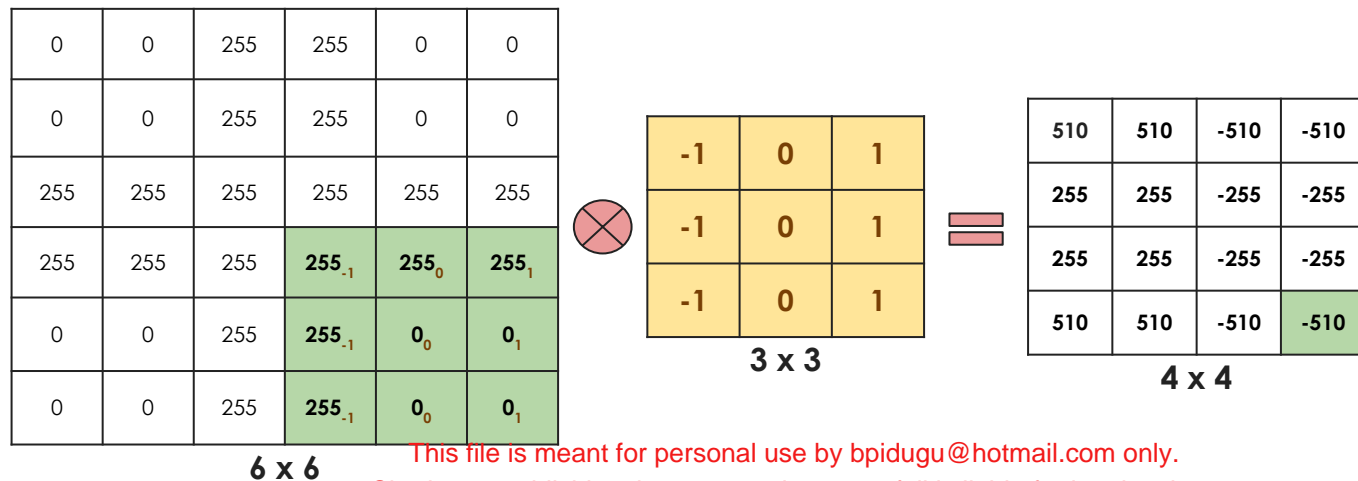


This file is meant for personal use by bpidugu@hotmail.com only.  
Sharing or publishing the contents in part or full is liable for legal action.

# Drawbacks of Convolutions

- **The second drawback** of Convolution has to do with the dimensionality of the convolution process - As we see in the example below, **the 3 x 3 convolution filter decreases the dimensions** of the image from **6 x 6 in the input**, to **4 x 4 in the output**.
- The generalized formula is that for an input **(n,n)** and filter size **(f,f)**, the output feature map will have dimensions **(n-f+1 , n-f+1)**, which is a reduced size in comparison to the image input size.

For example:



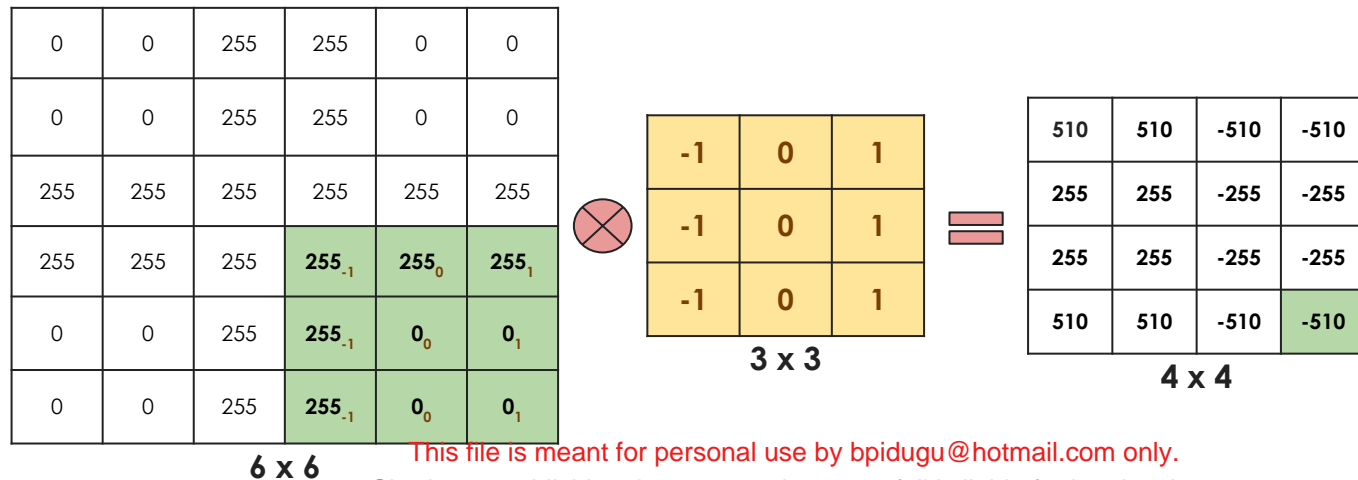
This file is meant for personal use by bpidugu@hotmail.com only.  
Sharing or publishing the contents in part or full is liable for legal action.



# Drawbacks of Convolutions

- While it may not make a huge difference for one convolution layer, **with multiple repeated convolutions** layer after layer (as is usually the case in Deep Convolutional Neural Networks), **this process may end up shrinking the image by a large amount**, and feature extraction might become more difficult with such a severely reduced image size.
- Hence, **we need a way to put a control** on whether we want to shrink the image after convolution or not, since it may not always be a desirable outcome for the output feature map.

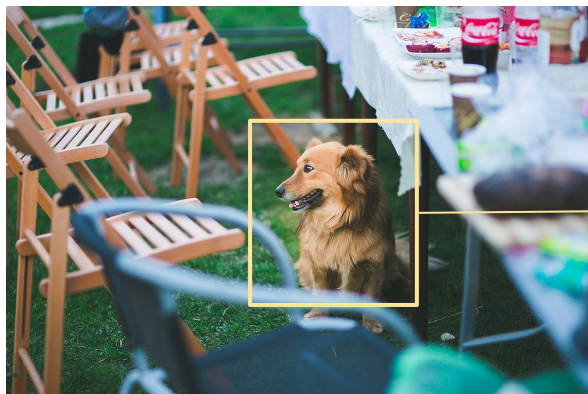
For example:



# Drawbacks of Convolutions

- **The third drawback** of convolution is that **the convolution operation records the precise position of each of the features in the input.** However, this would mean that just a small movement in the position of a feature would give us a different output.
- That may not always be ideal, and sometimes it may just be required to detect only the presence of an object in the image and not exactly where it is present. In that case we do not need exact positions, but just need to validate the presence of the object in the whole image.

**For example:**



Detecting and validating the presence of a dog in an image.

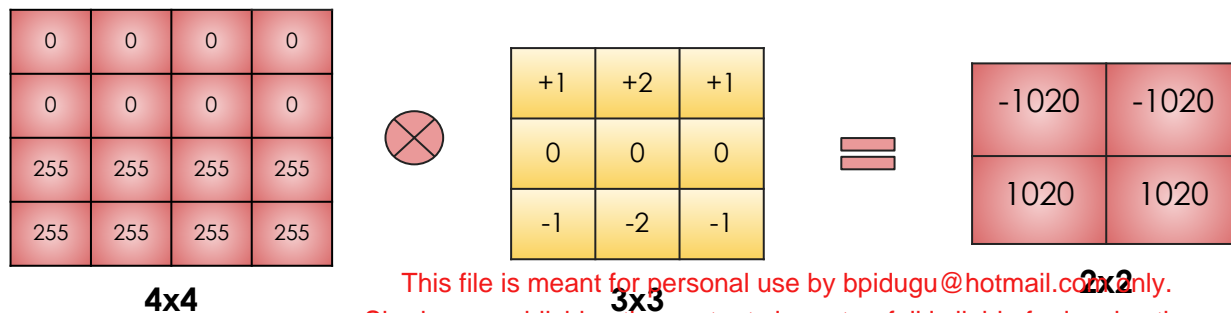


# Padding and Strides

This file is meant for personal use by bpidugu@hotmail.com only.  
Sharing or publishing the contents in part or full is liable for legal action.

# Padding

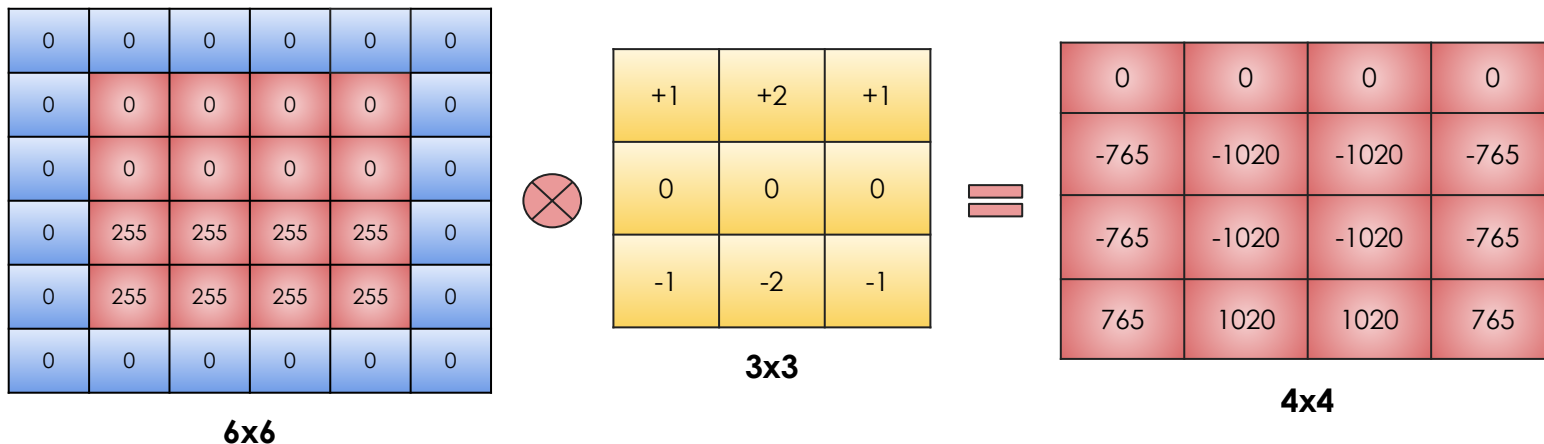
- The first two previously-mentioned drawbacks of convolutions can both be addressed through a single technique: **Padding**.
- Padding is a simple trick to add extra pixels to the edges or boundaries** of the image, so that the effective size of the image gets increased. This increase in effective size of the image solves both of the first two drawbacks of convolution in the following ways:
  - It solves the problem of under-utilized edge pixels**, because now the pixels originally at the edge are closer to the center and will be utilized more often by the convolution filter.
  - It also solves the shrinking problem of convolutions**, because padding creates an enlarged image, and shrinking the enlarged size can give an output with the original size.
- For example:** In the below visual, we observe that the input image, with a size of 4x4, shrinks to an output size of 2x2, when a 3x3 convolution filter is applied on it.



This file is meant for personal use by bpidugu@hotmail.com only.  
Sharing or publishing the contents in part or full is liable for legal action.

# Padding

- The disadvantage of shrinking exhibited by the convolution operation, can be addressed using padding as shown below. Here we pad the input image with zeros (0) along the boundaries, to effectively convert it from a 4x4 image to a 6x6 input image. When a 3x3 filter is convolved over a 6x6 image, the output is a 4x4 matrix which is the same size as the original 4x4 input.
- Thus, **padding effectively helps us get the same output size as the input, and avoid shrinkage.**

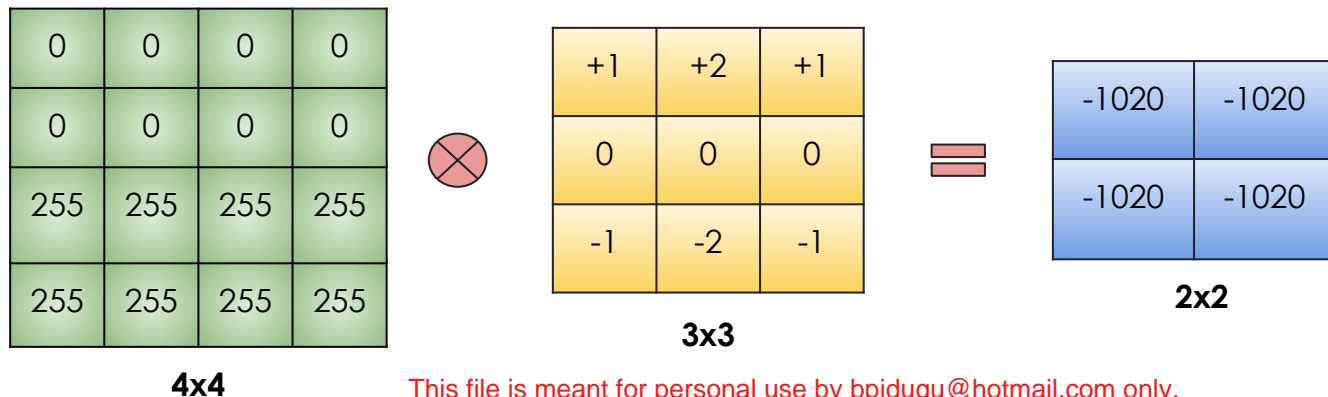


# Types of Padding

In the computer vision literature, two types of padding are described:

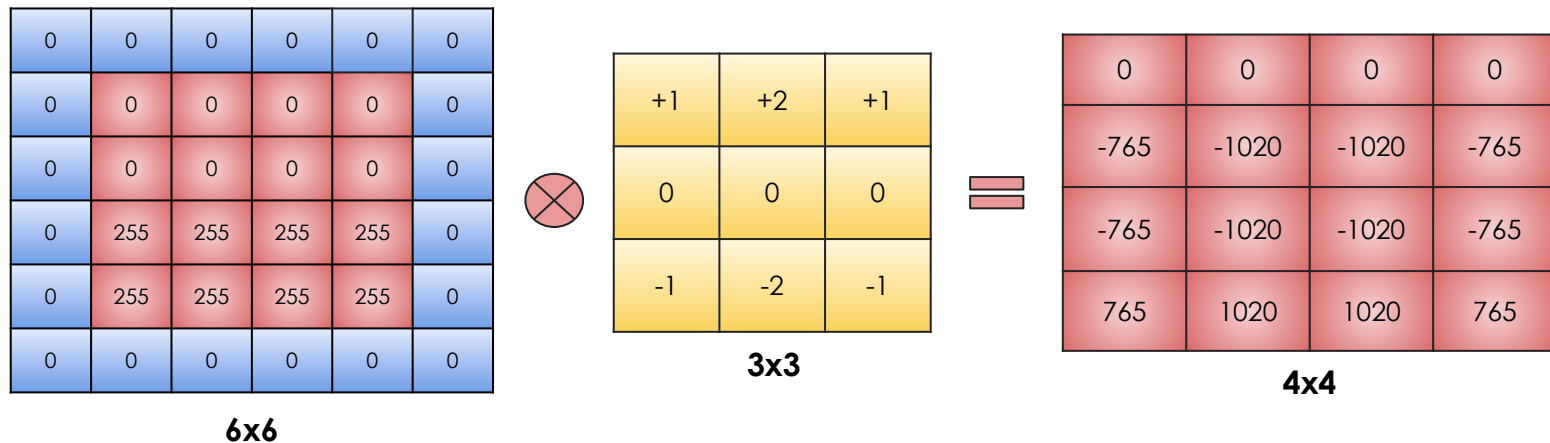
1. **Valid Padding:** This is the least used method as this essentially means **no padding**. Here we do not perform any padding operation on the array.
2. **Same Padding:** It pads the input size in-order to make **the output size same as the input size**. One of the basic type of same padding is called Zero Padding. In this, we add/pad zeros to all the edges of an image to prevent the loss of the original pixel values of the image.

**Example 1: Valid Padding** - We observe below that the output size of the array decreases to 2x2 when a 4x4 input image is convolved using a 3x3 filter without any padding.



# Valid and Same Padding

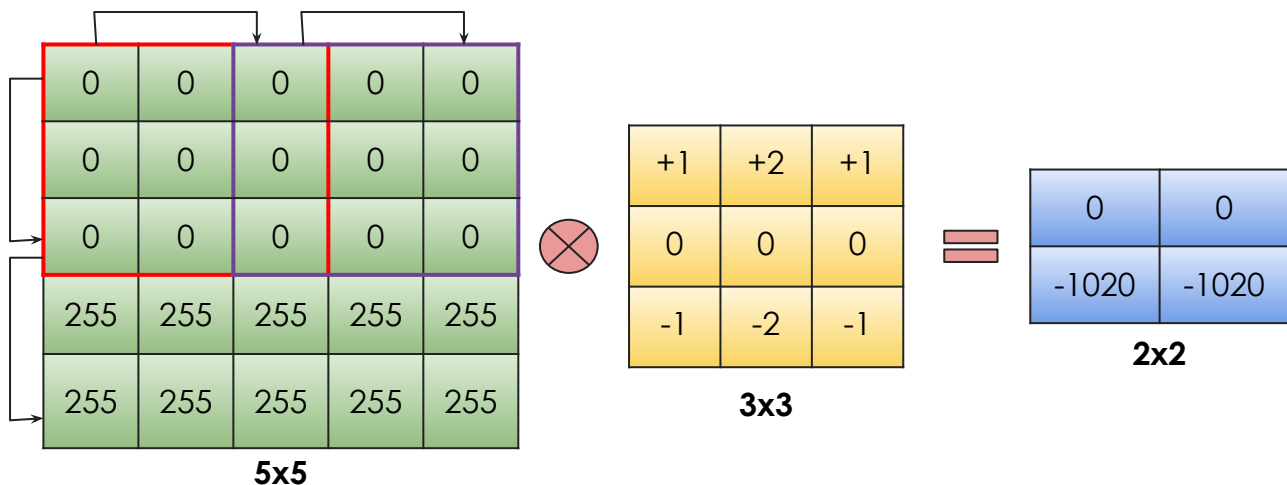
- Example 2: Same Padding** - We observe that **the shape of the output is now the same as that of the input image** when we apply zero padding on the input image. Thus padding helps in maintaining the output shape of the image to the same as the input even after convolution.



- Padding with zeros converts the 4x4 input to 6x6.
- This causes the output to be 4x4, the same as the original input size.

# Strides

- The third drawback of convolutions, however, about the operation being too precise, may be addressed through another idea called **Strides**.
- We observe during convolution that we slide from the top-most corner to the bottom-most corner **by a shift - this shift is called the Stride**.





# Mathematical interpretation of Padding and Strides

- Let us understand the mathematical interpretation of padding and strides using some examples:

## Example 1: Using only Padding:

Size of the input image (**nxn**): **4x4**

Size of the filter (**fxf**): **3x3**

Padding (**p**): **1**

The **shape of the output** can be calculated using the formula:  **$n+2p-f+1$**

So according to our example, the output shape would be:

$$4+(2*1)-3+1 = 4$$

## Example 2: Using both Padding and Strides:

Size of input image (**nxn**): **5x5**

Size of the filter (**fxf**): **3x3**

Padding (**p**): **0**

Stride (**s**): **2**

The **shape of the output** can be calculated using the formula:  **$\text{int}((n+2p-f)/s+1)$**

So according to our example, the output shape would be:

$$\text{int}((5+(2*0)-3)/2+1) = 2$$

**Note:** `int()` in Python rounds the number down to its nearest integer value. `int(3.6)=3`



# Pooling

This file is meant for personal use by bpidugu@hotmail.com only.  
Sharing or publishing the contents in part or full is liable for legal action.

# Pooling

- There is one more method that helps in **eliminating the unwanted information from an image**, which is more common, similar, and robust to small changes in the location of features in the input. This method is called **Pooling**.
- There are usually two common mechanisms used in Pooling:
- **Max Pooling:**
  - Max Pooling breaks the convolutional layer output into smaller patches. It is often, for example, a series of 2x2 pixel areas. So it is as if, in this example, there is a 2x2 filter sliding across the image with a stride of 2.
  - Max Pooling just looks at the values in the patch and selects the maximum value from that patch. So, **it helps in feature selection** by removing the weak/dim features from a bright foreground and in the process, helps in downsampling or dimensionality reduction.

20	40	70	80
80	90	30	60
35	15	70	80
80	90	90	10

4x4

90	80
90	90

2x2

**Max Pooling** with  
stride 2 and pool  
size 2

This file is meant for personal use by bpidugu@hotmail.com only.  
Sharing or publishing the contents in part or full is liable for legal action.

# Pooling

- **Average Pooling:**

- Average Pooling is exactly the same max pooling with one difference - we take the average of the values in the patch instead of the maximum value.

20	40	70	80
80	90	30	60
35	15	70	80
80	90	90	10

**4x4**



57.5	60
55	62.5

**2x2**

**Avg Pooling** with  
stride 2 and pool  
size 2

# Mathematical interpretation of Pooling

- Let us understand the mathematical interpretation of pooling using some examples:

<p><b>Example 1: Using only Pooling with no Padding or Strides:</b></p> <p>Size of the input image (<b>nxn</b>): <b>5x5</b> Pooling size (<b>fx</b><b>f</b>): <b>3x3</b></p>	<p><b>Example 2: Using Pooling with Padding and Strides:</b></p> <p>Size of the Input image (<b>nxn</b>): <b>5x5</b> Pooling size (<b>fx</b><b>f</b>): <b>3x3</b> Padding (<b>p</b>): <b>1</b> Stride (<b>s</b>): <b>2</b></p>
<p>The <b>shape of the output</b> can be calculated using the formula: <b><math>n-f+1</math></b></p> <p>So according to our example, the output shape would be: <math>5-3+1=3</math></p>	<p>The shape of the output can be calculated using the formula: <b><math>\text{int}((n+2p-f)/s+1)</math></b></p> <p>So according to our example, the output shape would be: <math>\text{int}((5+(2*1)-3)/2+1) = 3</math></p> <p><b>Note:</b> int() in Python rounds the number down to its nearest integer value. <math>\text{int}(3.6)=3</math></p>



# Thank You

This file is meant for personal use by bpidugu@hotmail.com only.  
Sharing or publishing the contents in part or full is liable for legal action.