How to Calculate the Output Shape of the CNN

Most of the NN platforms out there do not require knowing the input shape after each conv layer. However, understanding how CNN shrinks the image is very important.

The goal of this mini-project is to create a function that calculates the **output shape** of the image at each stage in the conv layers.

Concept

There are a few assumption: filter is a square matrix, strides is consistent in both direction, and input has even dimention. I make these assumptions just to keep things simpler. It does not affect the calculation too much.

- **Input** (x, y): can be rbg $(x, y, 3)$ or gray scale $(x, y, 1)$
- **Filter (kernel)** (f, f): Operation is sum of the dot product. 6×6 image when apply 3×3 filter would result 4×4 (with stride 1). More general, if a m vector sliding on a n vector, it would result m-n+1 (counting the initial position as a step). Thus,

New image size after filter:

$$(x - f + 1, y - f + 1)$$

Note: Some source argues that filter must be odd because avoiding distortion on the output. I say it is odd because even filter cause conflict with padding. You will see why in a moment.

- **Padding (p)**: simply just add p tiles to each size of x and y to avoid missing the pixel information from the borders

New size after applying filter(f) with padding(p):

$$(x - f + 1 + 2p, y - f + 1 + 2p)$$

Note Set padding = "same" means output is same as the input size, which is:

$$x - f + 1 + 2p = x$$
$$p = \frac{f - 1}{2}$$

- **Stride (s)**: If we don't set stride carefully then we'd have issues with spacing. Normally, practitioners will increase the stride if they want receptive fields to overlap less and if they want smaller spatial dimensions.

Hence, stride should satisfies the condition:

$$\{s : s | x + 2p - f\}$$

Thus, new size after applying filter(f) with padding(p) and stride(s)

$$(\frac{x + 2p - f}{s} + 1, \frac{y + 2p - f}{s} + 1)$$

- **Max pooling**: Like filter but with the different operation (max, mean, mode,...). It is also called downsampling. Max pooling is more popular than average pooling because we are interested in the most important characteristics. Note that max-pooling it is almost always 2×2 pixels applied with a stride of 2 pixels. So if the input is odd then we will have a minor spacing problem.

New size after everything:

$$\left(\frac{\frac{x+2p-f}{s} + 1}{2}, \frac{\frac{y+2p-f}{s} + 1}{2} \right)$$

Note: $x + 2p - f$ is odd so $\frac{x+2p-f}{s}$ is also odd.

Apply

In [185]:
```
# Everything that we discuss so far can be summarize to the following function
def Output_calculator(x,y,f=1,s=1,p=0,max_p=False):
    if p=='same':
        p=(f-1)/2
    if (x+2*p-f) % s !=0:
        print("Invalid stride")
        return 0,0
    if max_p==False:
        mp=1
    else:
        mp=2
        f=1

    x=((x+2*p-f)/(s)+1)/mp
    y=((y+2*p-f)/(s)+1)/mp
    print(x,y)
    return x,y
```

In [186]:
```
# Let's try to caculate the parameter of the following model
from keras import layers
from keras import models

model = models.Sequential()
model.add(layers.Conv2D(32,kernel_size=(5,5),strides=(1,1),
                        padding='same', activation='relu',
                        input_shape=(128, 128, 3)))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))

model.add(layers.Conv2D(128, (3, 3), padding='same', activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
```

In [187]:
```
# Using the function that we created above
x,y=Output_calculator(x=128,y=128,f=5,s=1,p='same')
x,y=Output_calculator(x=x,y=y,max_p=True)
print("--------------")
x,y=Output_calculator(x=x,y=y,f=3)
x,y=Output_calculator(x=x,y=y,max_p=True)
print("--------------")
x,y=Output_calculator(x=x,y=y,f=3,p='same')
x,y=Output_calculator(x=x,y=y,max_p=True)
print("--------------")
```

```
128.0 128.0
64.0 64.0
--------------
62.0 62.0
31.0 31.0
--------------
31.0 31.0
15.5 15.5
--------------
```

In [188]:
```
# Let's compare it with the summary function
model.summary()
```

```
Model: "sequential_31"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_77 (Conv2D)           (None, 128, 128, 32)      2432
_________________________________________________________________
max_pooling2d_75 (MaxPooling (None, 64, 64, 32)        0
_________________________________________________________________
conv2d_78 (Conv2D)           (None, 62, 62, 64)        18496
_________________________________________________________________
max_pooling2d_76 (MaxPooling (None, 31, 31, 64)        0
_________________________________________________________________
conv2d_79 (Conv2D)           (None, 31, 31, 128)       73856
_________________________________________________________________
max_pooling2d_77 (MaxPooling (None, 15, 15, 128)       0
=================================================================
Total params: 94,784
Trainable params: 94,784
Non-trainable params: 0
_________________________________________________________________
```