Dimensionality

In [1]:

```
from keras.models import Sequential
from keras.layers import Conv2D
```

```
Using TensorFlow backend.
```

Formula: Number of Parameters in a Convolutional Layer

The number of parameters in a convolutional layer depends on the supplied values of filters, kernel_size, and input_shape. Let's define a few variables:

- K - the number of filters in the convolutional layer
- F - the height and width of the convolutional filters
- D_in - the depth of the previous layer

Notice that K = filters, and F = kernel_size. Likewise, D_in is the last value in the input_shape tuple.

Since there are **F*F*D_in** weights per filter, and the convolutional layer is composed of K filters, the total number of weights in the convolutional layer is **K*F*F*D_in**. Since there is one bias term per filter, the convolutional layer has K biases. Thus, the number of parameters in the convolutional layer is given by **K*F*F*D_in + K**.

In [4]:

```
model = Sequential()
model.add(Conv2D(filters=16, kernel_size=2, strides=2, padding='valid',
    activation='relu', input_shape=(200, 200, 1)))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 100, 100, 16)      80
=================================================================
Total params: 80.0
Trainable params: 80
Non-trainable params: 0.0
_________________________________________________________________
```

We will not train this CNN; instead, we'll use the executable to study how the dimensionality of the convolutional layer changes, as a function of the supplied arguments.
Feel free to change the values assigned to the arguments (filters, kernel_size, etc) in your conv-dims.py file.

Take note of how the **number of parameters** in the convolutional layer changes. This corresponds to the value under Param # in the printed output. In the figure above, the convolutional layer has 80 parameters.

Also notice how the **shape** of the convolutional layer changes. This corresponds to the value under Output Shape in the printed output. In the figure above, None corresponds to the batch size, and the convolutional layer has a height of 100, width of 100, and depth of 16.

QUESTION 1 OF 3

How many parameters does the convolutional layer have?

In [13]:

```
model1 = Sequential()
model1.add(Conv2D(filters=32, kernel_size=3, strides=2, padding='same',
    activation='relu', input_shape=(128, 128, 3)))
model1.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_8 (Conv2D)            (None, 64, 64, 32)        896
=================================================================
Total params: 896.0
Trainable params: 896
Non-trainable params: 0.0
_________________________________________________________________
```

Correct! The number of parameters is (32 x 3 x 3 x 3) + 32 = 896.

QUESTION 2 OF 3

What is the depth of the convolutional layer?

In [7]:

```
print("The depth of convolutional layer equals to the number of the filters - 32")
```

```
The depth of convolutional layer equals to the number of the filters - 32
```

QUESTION 3 OF 3

What is the width of the convolutional layer?

In [14]:

```
print("The width of convolutional layer equals to the input layer width devided
by strides - 128/2 = 64")
```

```
The width of convolutional layer equals to the input layer width devided by
strides - 128/2 = 64
```

Checking the Dimensionality of Max Pooling Layers

Example

Say I'm constructing a CNN, and I'd like to reduce the dimensionality of a convolutional layer by following it with a max pooling layer. Say the convolutional layer has size (100, 100, 15), and I'd like the max pooling layer to have size (50, 50, 15). I can do this by using a 2x2 window in my max pooling layer, with a stride of 2, which could be constructed in the following line of code:

```
    MaxPooling2D(pool_size=2, strides=2)
```

In [2]:

```
model = Sequential()
model.add(MaxPooling2D(pool_size=2, strides=2, input_shape=(100, 100, 15)))
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
max_pooling2d_1 (MaxPooling2 (None, 50, 50, 15)        0
=================================================================
Total params: 0.0
Trainable params: 0.0
Non-trainable params: 0.0
_________________________________________________________________
```

Feel free to change the arguments in your pool-dims.py file, and check how the shape of the max pooling layer changes.

In [3]:

```
model1 = Sequential()
model1.add(MaxPooling2D(pool_size=2, strides=2, input_shape=(200, 200, 16)))
model1.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
max_pooling2d_2 (MaxPooling2 (None, 100, 100, 16)      0
=================================================================
Total params: 0.0
Trainable params: 0.0
Non-trainable params: 0.0
_________________________________________________________________
```