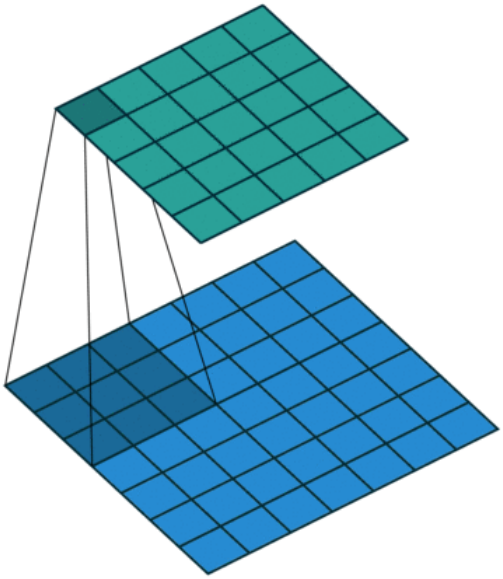

Convolution with 3x3 window and stride 1

Image source: http://iamaaditya.github.io/2016/03/one-by-one-convolution/

Convolutional Layers in Keras

To create a convolutional layer in Keras, you must first import the necessary module:

```
from keras.layers import Conv2D
```

Then, you can create a convolutional layer by using the following format:

```
Conv2D(filters, kernel_size, strides, padding, activation='relu', input_shape)
```

Arguments

You must pass the following arguments:

- `filters` - The number of filters.
- `kernel_size` - Number specifying both the height and width of the (square) convolution window.

- `strides` - The stride of the convolution. If you don't specify anything, `strides` is set to `1`.
- `padding` - One of `'valid'` or `'same'`. If you don't specify anything, `padding` is set to `'valid'`.
- `activation` - Typically `'relu'`. If you don't specify anything, no activation is applied. You are **strongly encouraged** to add a ReLU activation function to **every** convolutional layer in your networks.

NOTE: It is possible to represent both `kernel_size` and `strides` as either a number or a tuple.

When using your convolutional layer as the first layer (appearing after the input layer) in a model, you must provide an additional `input_shape` argument:

- `input_shape` - Tuple specifying the height, width, and depth (in that order) of the input.

NOTE: Do ***not*** include the `input_shape` argument if the convolutional layer is *not* the first layer in your network.

There are many other tunable arguments that you can set to change the behavior of your convolutional layers. To read more about these, we recommend perusing the official **documentation**.

Example #1

Say I'm constructing a CNN, and my input layer accepts grayscale images that are 200 by 200 pixels (corresponding to a 3D array with height 200, width 200, and depth 1). Then, say I'd like the next layer to be a convolutional layer with 16 filters, each with a width and height of 2. When performing the convolution, I'd like the filter to jump two pixels at a time. I also don't want the filter to extend outside of the image boundaries; in other words, I don't want to pad the image with zeros. Then, to construct this convolutional layer, I would use the following line of code:

```
Conv2D(filters=16, kernel_size=2, strides=2, activation='relu', input_shape=(200, 200, 1
))
```

constructed in Example 1 as input. Say I'd like my new layer to have 32 filters, each with a height and width of 3. When performing the convolution, I'd like the filter to jump 1 pixel at a time. I want the convolutional layer to see all regions of the previous layer, and so I don't mind if the filter hangs over the edge of the previous layer when it's performing the convolution. Then, to construct this convolutional layer, I would use the following line of code:

```
Conv2D(filters=32, kernel_size=3, padding='same', activation='relu')
```

Example #3

If you look up code online, it is also common to see convolutional layers in Keras in this format:

```
Conv2D(64, (2,2), activation='relu')
```

In this case, there are 64 filters, each with a size of 2x2, and the layer has a ReLU activation function. The other arguments in the layer use the default values, so the convolution uses a stride of 1, and the padding has been set to 'valid'.

NEXT