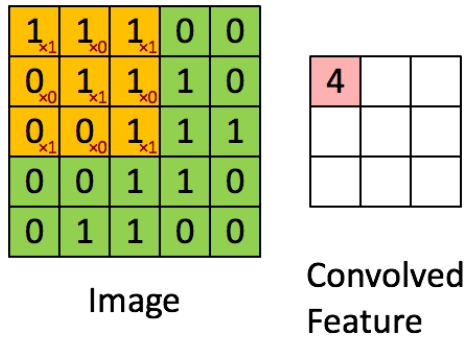
Deep Learning - Udacity



Just as with neural networks, we create a CNN in Keras by first creating a `Sequential` model.

We add layers to the network by using the `.add()` method.

Copy and paste the following code into a Python executable named `conv-dims.py`:

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

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

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.

Run `python path/to/conv-dims.py` and look at the output. It should appear as follows:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 100, 100, 16)      80
=================================================================
Total params: 80
Trainable params: 80
Non-trainable params: 0
_________________________________________________________________
```

Do the dimensions of the convolutional layer line up with your expectations?

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`.

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`.

Formula: Shape of a Convolutional Layer

The shape of a convolutional layer depends on the supplied values of `kernel_size`, `input_shape`, `padding`, and `stride`. Let's define a few variables:

- `K` - the number of filters in the convolutional layer
- `F` - the height and width of the convolutional filters
- `S` - the stride of the convolution
- `H_in` - the height of the previous layer
- `W_in` - the width of the previous layer

Notice that `K` = `filters`, `F` = `kernel_size`, and `S` = `stride`. Likewise, `H_in` and `W_in` are the first and second value of the `input_shape` tuple, respectively.

The **depth** of the convolutional layer will always equal the number of filters `K`.

If `padding = 'same'`, then the spatial dimensions of the convolutional layer are the following:

- **height** = ceil(float(`H_in`) / float(`S`))
- **width** = ceil(float(`W_in`) / float(`S`))

If `padding = 'valid'`, then the spatial dimensions of the convolutional layer are the following:

- **height** = ceil(float(`H_in` - `F` + 1) / float(`S`))
- **width** = ceil(float(`W_in` - `F` + 1) / float(`S`))

Please change the `conv-dims.py` file, so that it appears as follows:

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

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

Run `python path/to/conv-dims.py`, and use the output to answer the questions below.

Question 1 of 3

How many parameters does the convolutional layer have?

-
-
-
-

Question 2 of 3

What is the depth of the convolutional layer?

-
-
-
-

Question 3 of 3

What is the width of the convolutional layer?