

# Convolutions

IMAGE PROCESSING WITH KERAS IN PYTHON



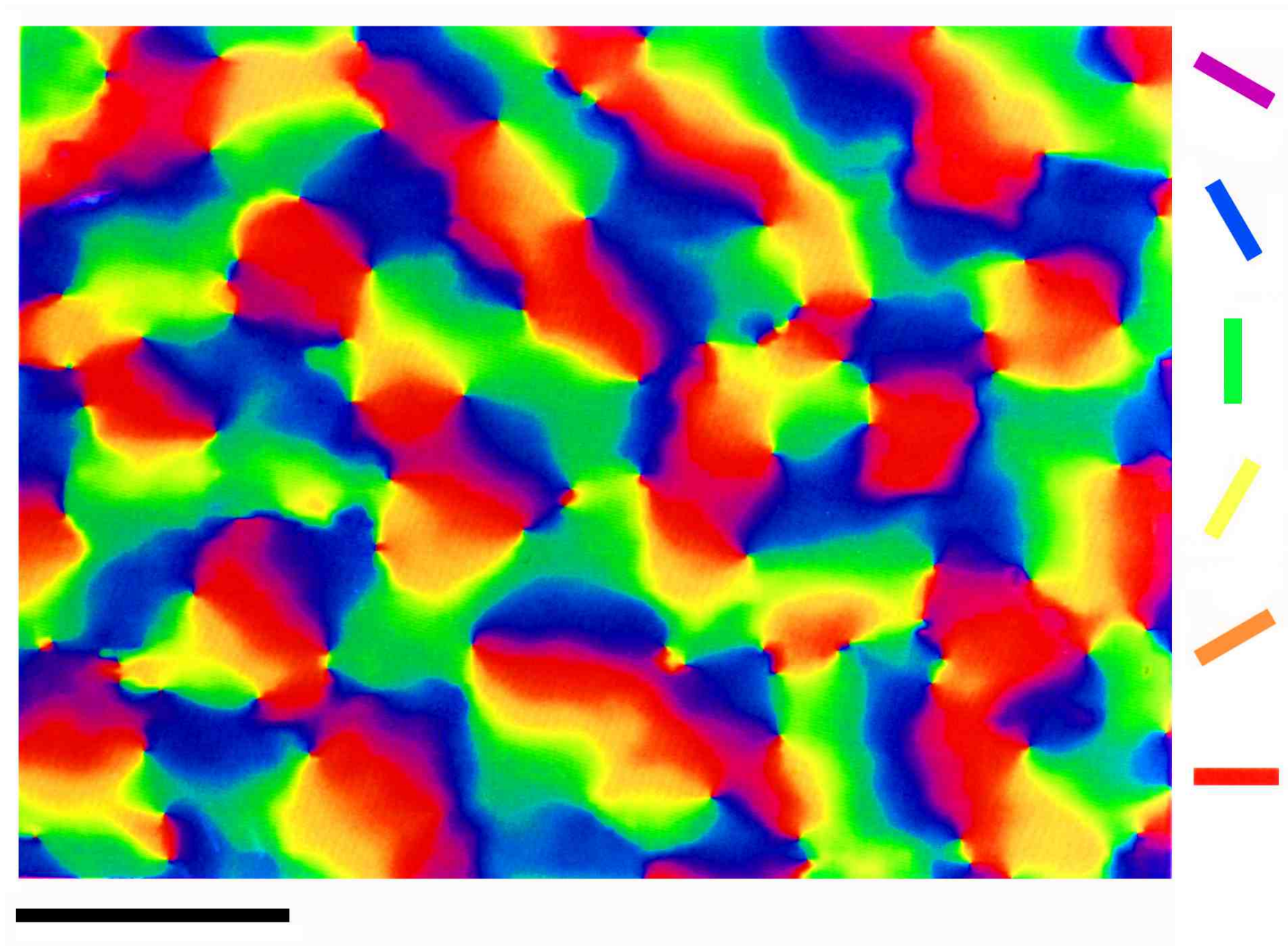
**Ariel Rokem**

Senior Data Scientist, University of  
Washington

# Using correlations in images

- Natural images contain spatial correlations
- For example, pixels along a contour or edge
- How can we use these correlations?

# Biological inspiration



# What is a convolution?

```
array = np.array([0, 0, 0, 0, 0, 1, 1, 1, 1, 1])
kernel = np.array([-1, 1])
conv = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0])

conv[0] = (kernel * array[0:2]).sum()
conv[1] = (kernel * array[1:3]).sum()
conv[2] = (kernel * array[2:4]).sum()
...

for ii in range(8):
    conv[ii] = (kernel * array[ii:ii+2]).sum()
conv
```

```
array([0, 0, 0, 0, 1, 0, 0, 0, 0])
```

# Convolution in one dimension

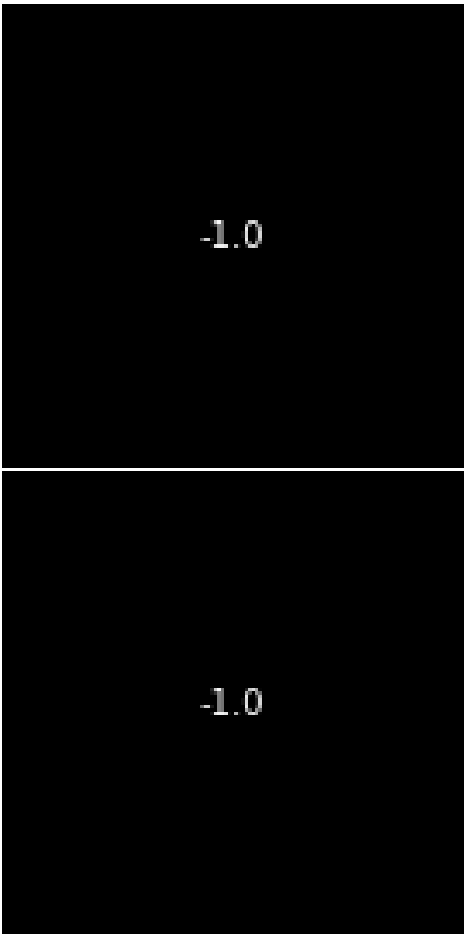
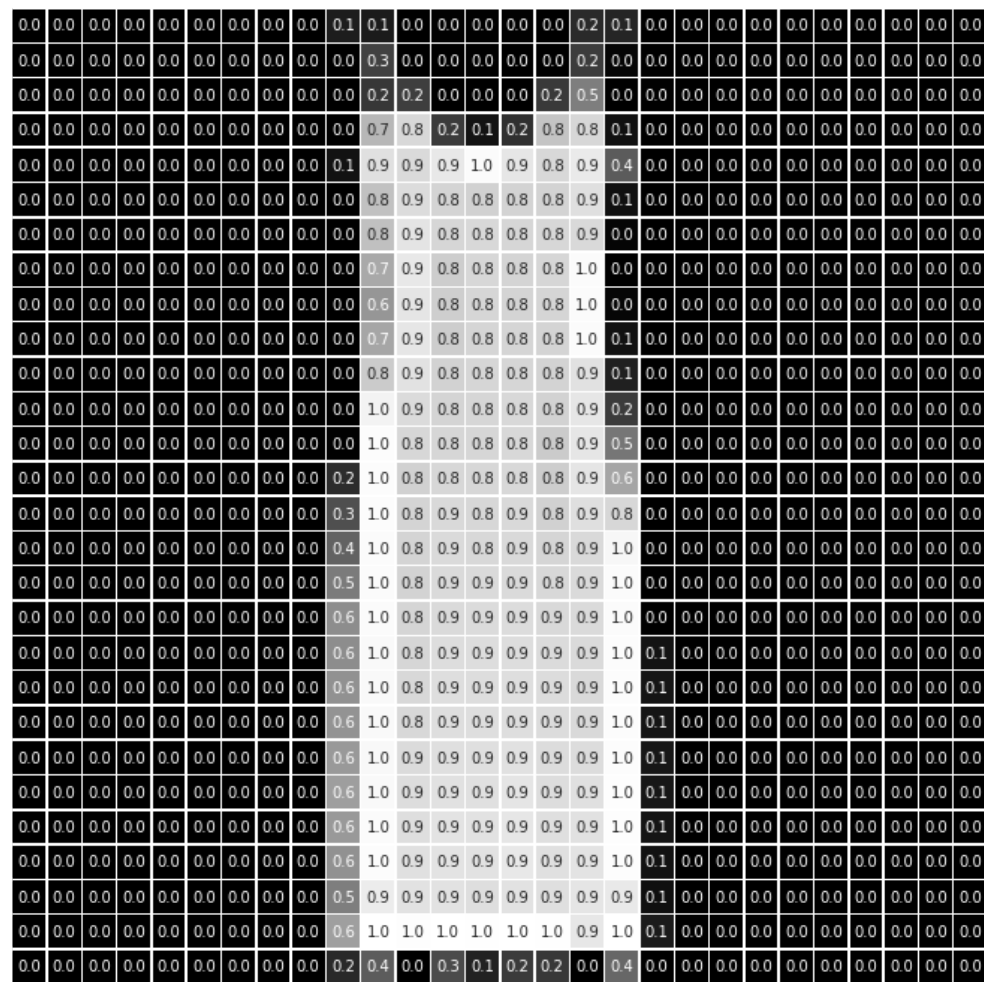
```
array = np.array([0, 0, 1, 1, 0, 0, 1, 1, 0, 0])
kernel = np.array([-1, 1])

conv = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0])
for ii in range(8):
    conv[ii] = (kernel * array[ii:ii+2]).sum()

conv
```

```
array([ 0,  1,  0, -1,  0,  1,  0, -1,  0])
```

# Image convolution

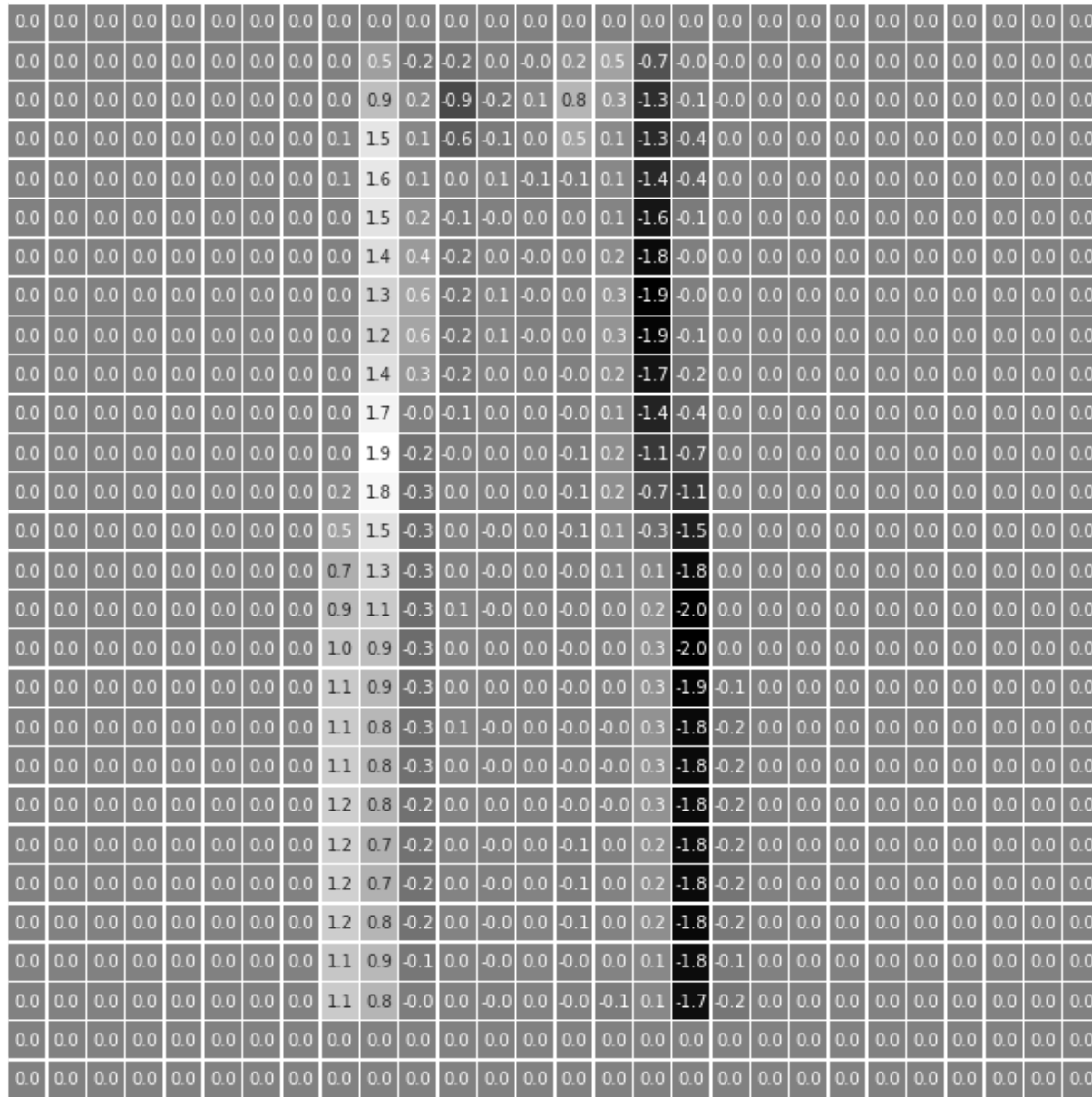


10

10



# Image convolution

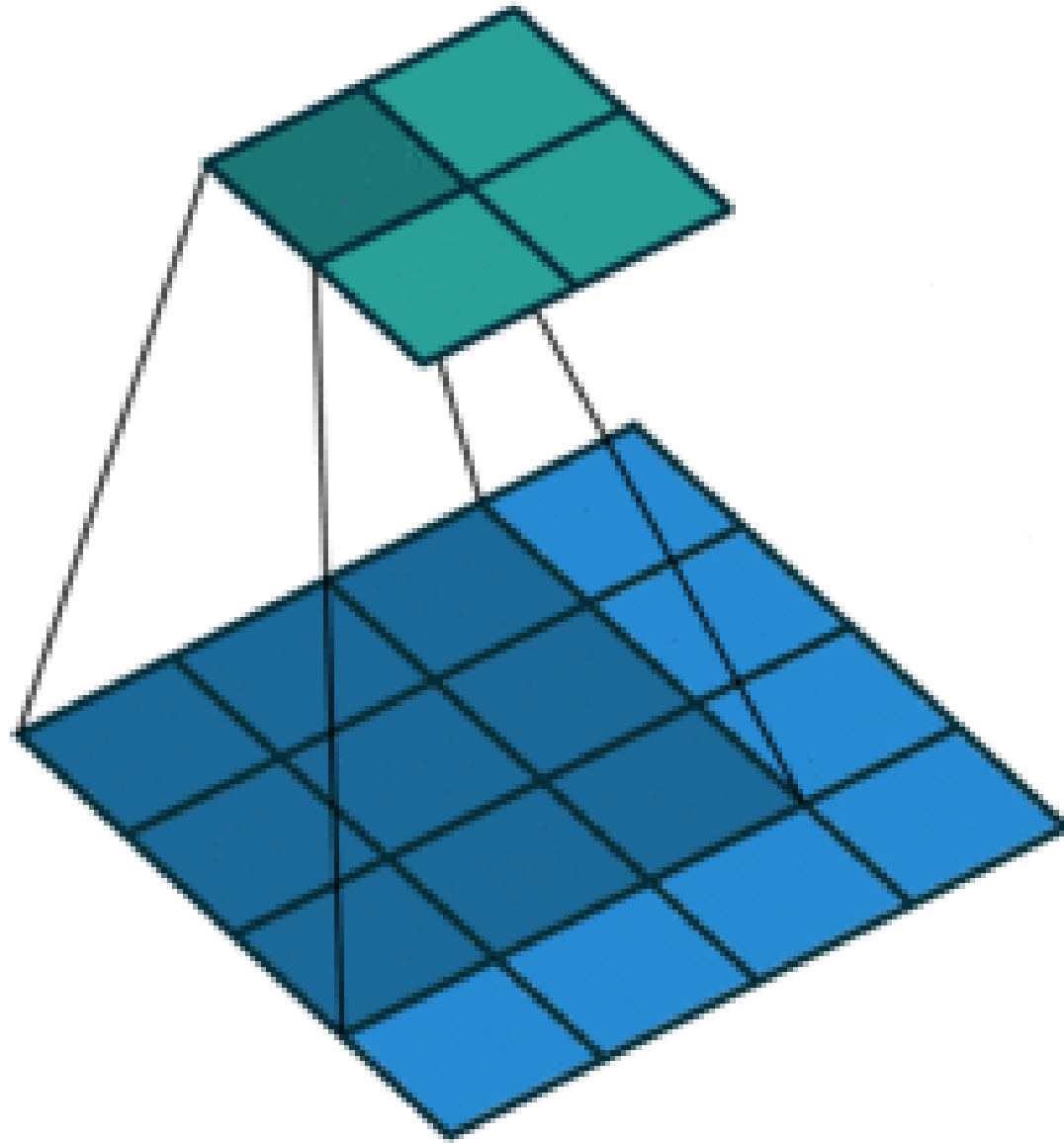


# Two-dimensional convolution

```
kernel = np.array([[ -1,  1],  
                   [ -1,  1]])  
  
conv = np.zeros((27, 27))  
  
for ii in range(27):  
    for jj in range(27):  
        window = image[ii:ii+2, jj:jj+2]  
        conv[ii, jj] = np.sum(window * kernel)
```



# Convolution



# Let's practice!

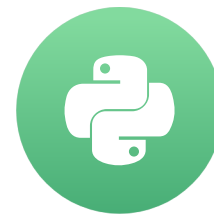
IMAGE PROCESSING WITH KERAS IN PYTHON

# Implementing convolutions in Keras

IMAGE PROCESSING WITH KERAS IN PYTHON

**Ariel Rokem**

Senior Data Scientist, University of  
Washington



# Keras Convolution layer

```
from keras.layers import Conv2D  
Conv2D(10, kernel_size=3, activation='relu')
```

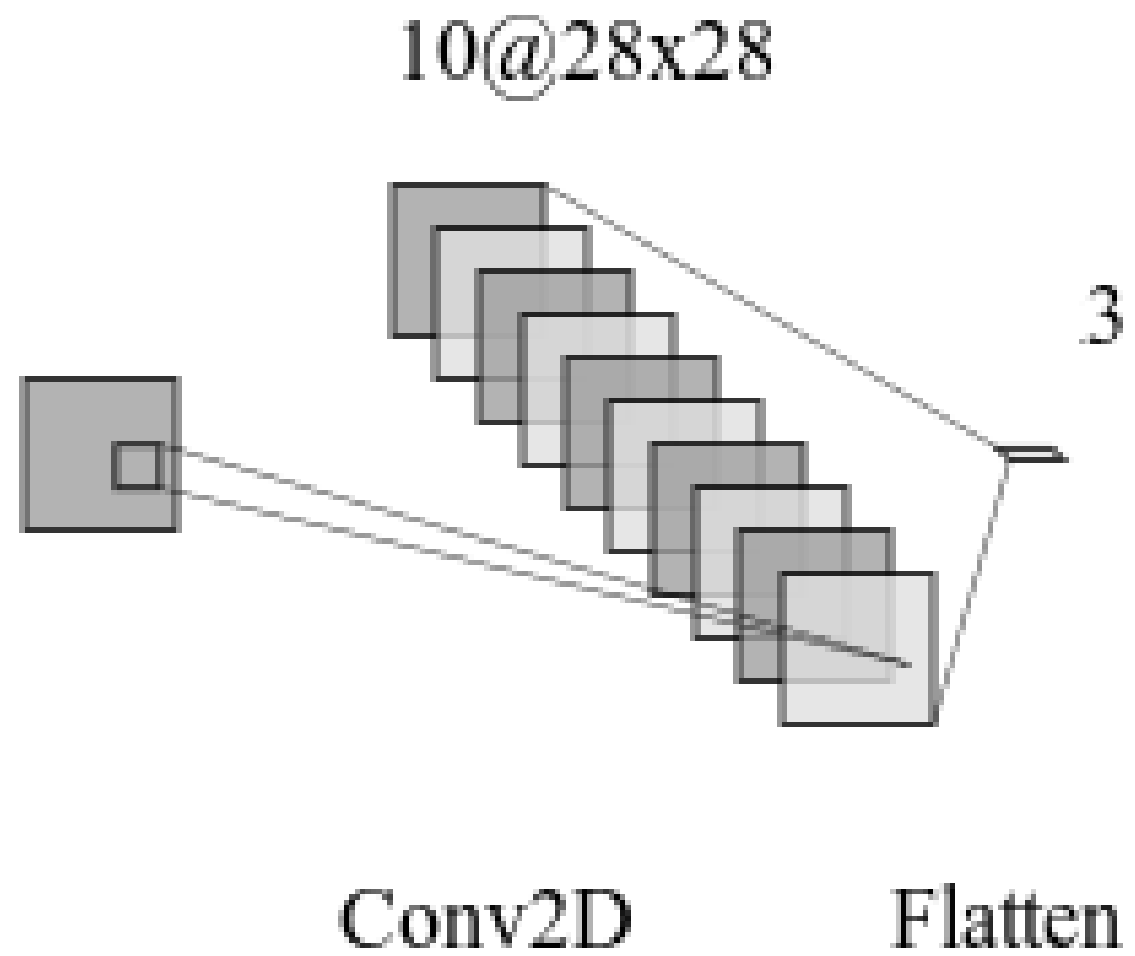
# Integrating convolution layers into a network

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten

model = Sequential()
model.add(Conv2D(10, kernel_size=3, activation='relu',
                input_shape=(img_rows, img_cols, 1)))

model.add(Flatten())
model.add(Dense(3, activation='softmax'))
```

# Our CNN



# Fitting a CNN

```
model.compile(optimizer='adam',  
              loss='categorical_crossentropy',  
              metrics=['accuracy'])  
  
train_data.shape
```

```
(50, 28, 28, 1)
```

```
model.fit(train_data, train_labels, validation_split=0.2,  
          epochs=3)  
  
model.evaluate(test_data, test_labels, epochs=3)
```



# Let's practice!

IMAGE PROCESSING WITH KERAS IN PYTHON

# Tweaking your convolutions

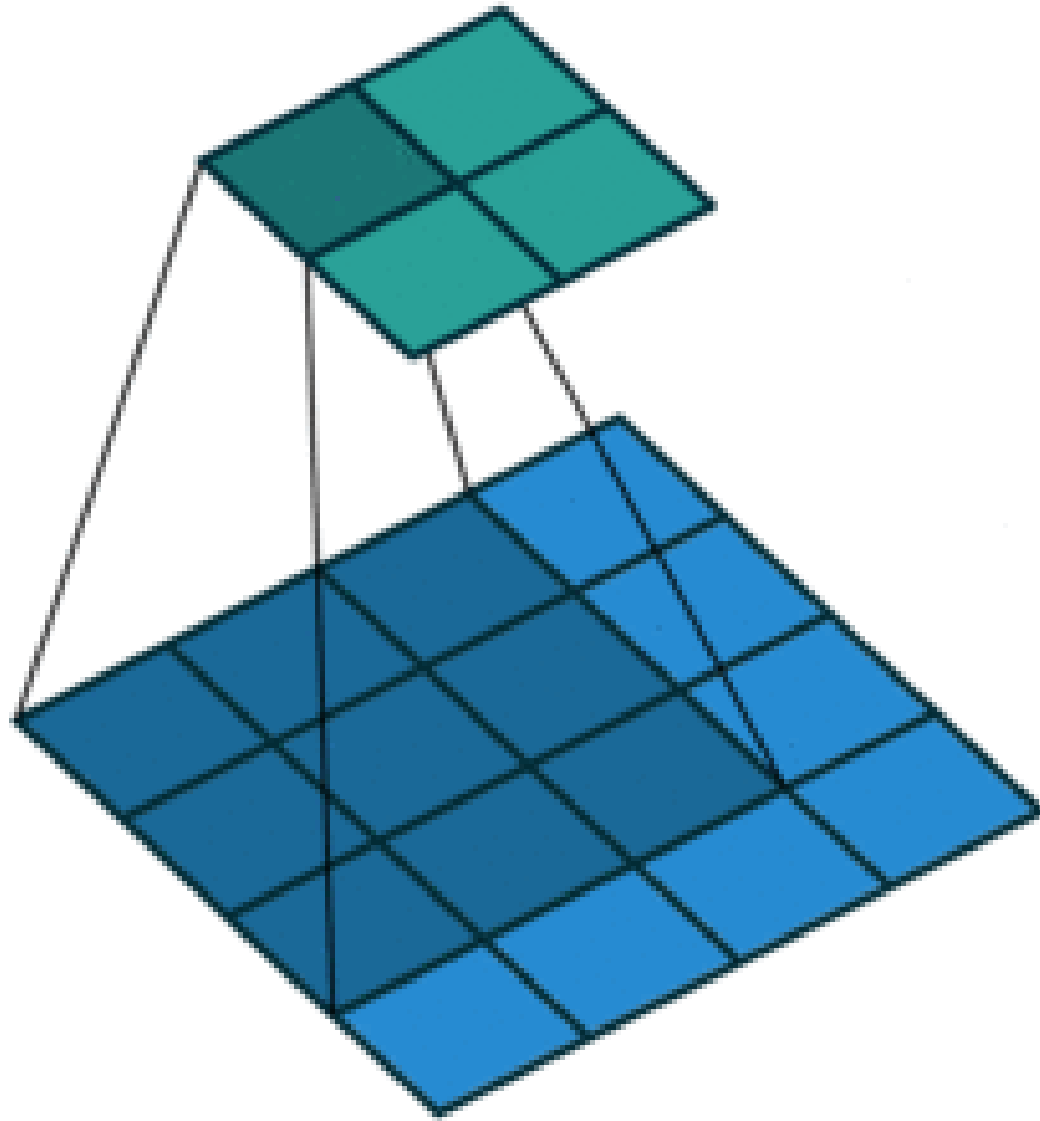
IMAGE PROCESSING WITH KERAS IN PYTHON



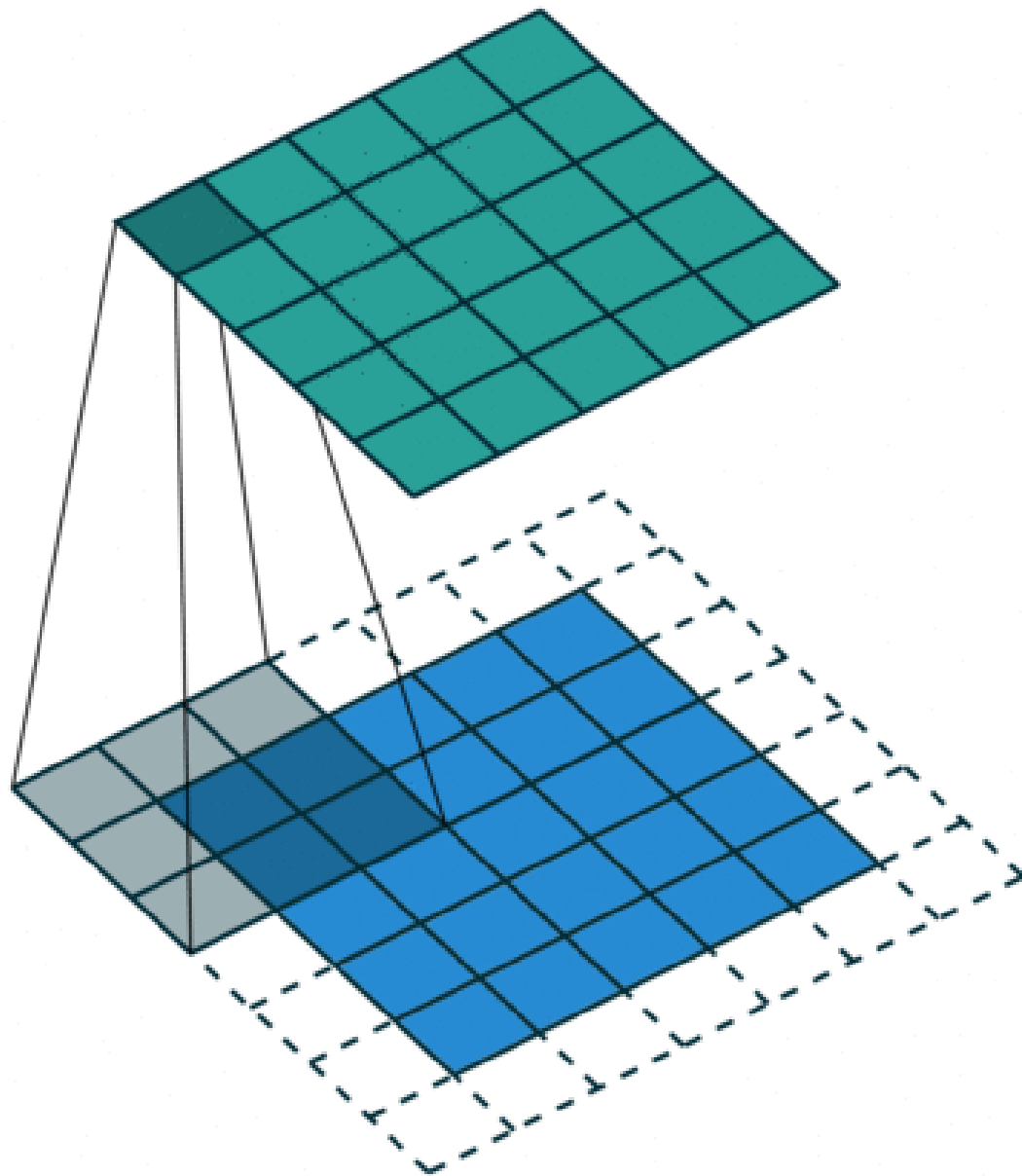
**Ariel Rokem**

Senior Data Scientist, University of  
Washington

# Convolution



# Convolution with zero padding



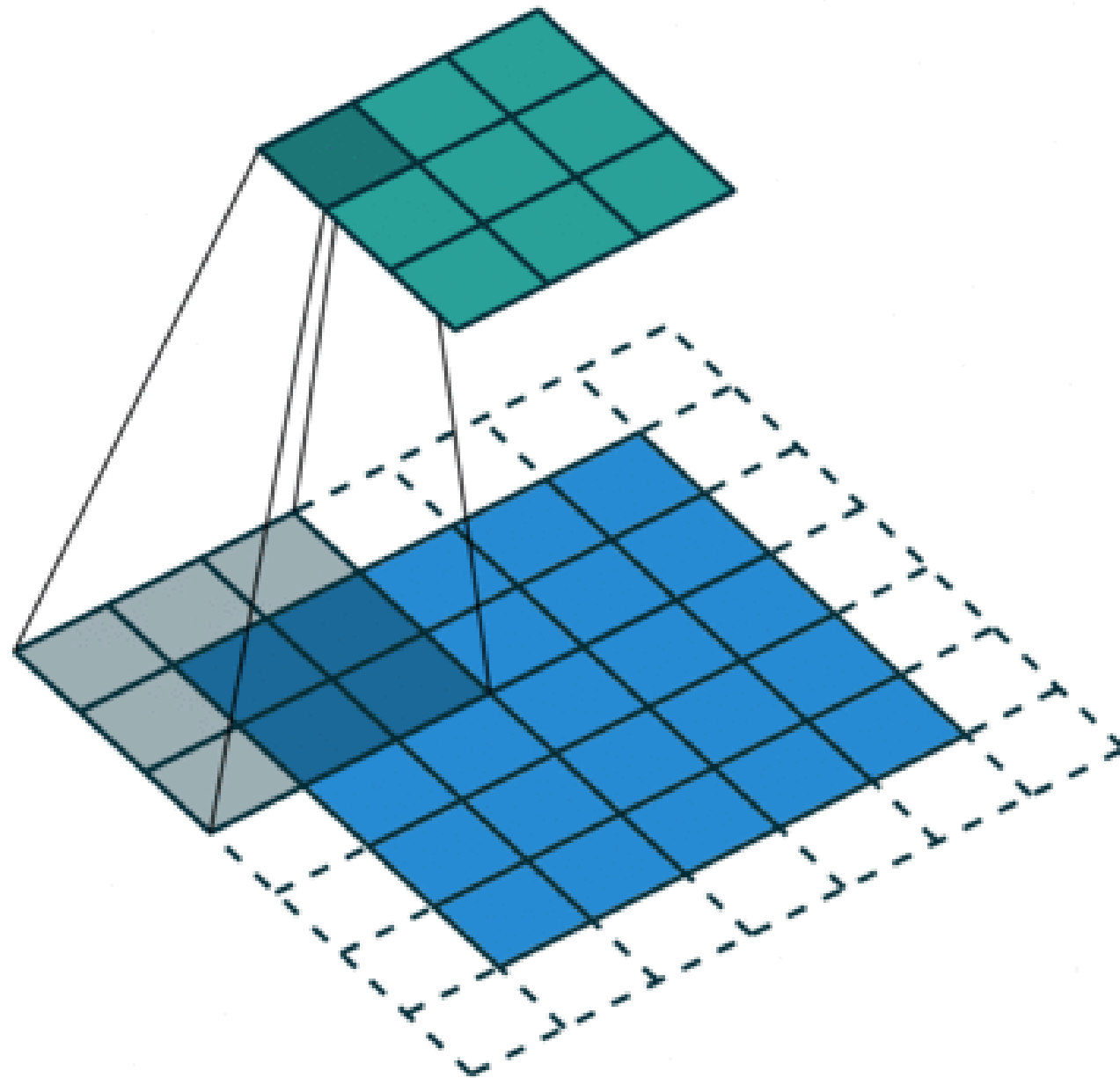
# Zero padding in Keras

```
model.add(Conv2D(10, kernel_size=3, activation='relu',  
                 input_shape=(img_rows, img_cols, 1)),  
            padding='valid')
```

# Zero padding in Keras

```
model.add(Conv2D(10, kernel_size=3, activation='relu',  
                 input_shape=(img_rows, img_cols, 1)),  
            padding='same')
```

# Strides





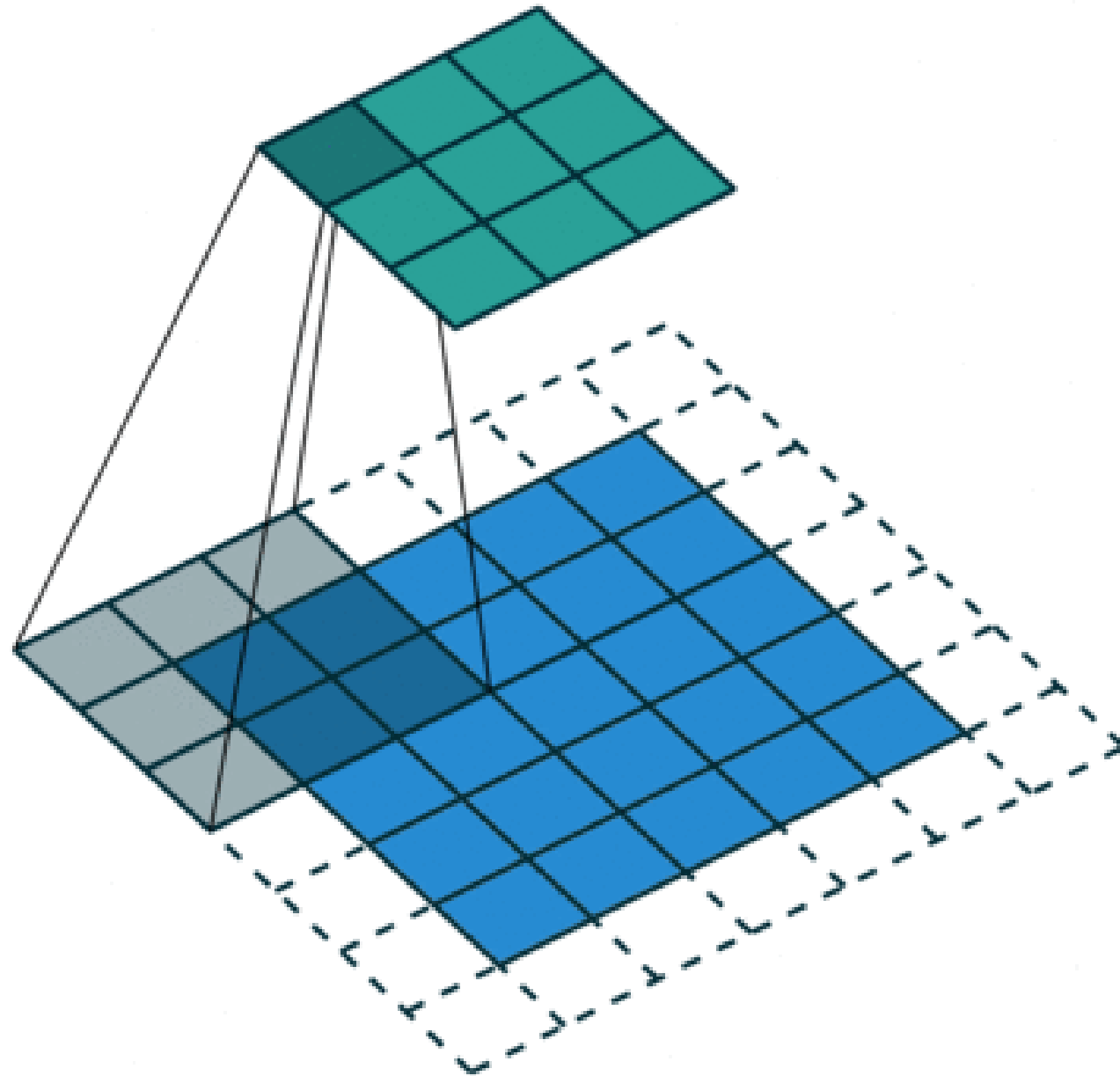
# Strides in Keras

```
model.add(Conv2D(10, kernel_size=3, activation='relu',  
                 input_shape=(img_rows, img_cols, 1)),  
            strides=1)
```

# Strides in Keras

```
model.add(Conv2D(10, kernel_size=3, activation='relu',  
                 input_shape=(img_rows, img_cols, 1)),  
            strides=2)
```

# Example



# Calculating the size of the output

$$O = ((I - K + 2P)/S) + 1$$

where

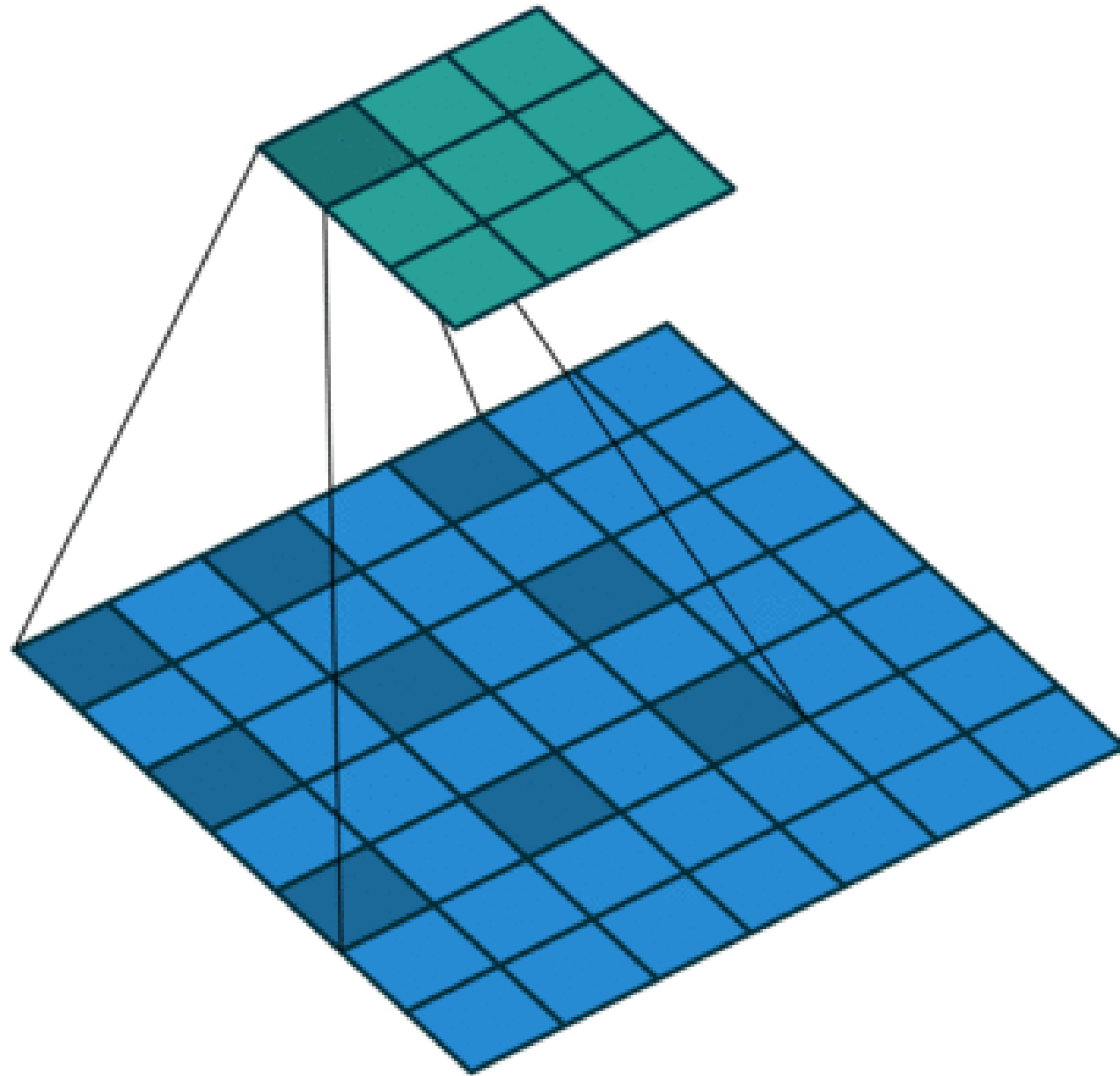
- $I$  = size of the input
- $K$  = size of the kernel
- $P$  = size of the zero padding
- $S$  = strides

# Calculating the size of the output

$$28 = ((28 - 3 + 2)/1) + 1$$

$$10 = ((28 - 3 + 2)/3) + 1$$

# Dilated convolutions



# Dilation in Keras

```
model.add(Conv2D(10, kernel_size=3, activation='relu',  
                 input_shape=(img_rows, img_cols, 1)),  
            dilation_rate=2)
```



# Let's practice!

IMAGE PROCESSING WITH KERAS IN PYTHON