Convolutions

IMAGE PROCESSING WITH KERAS IN PYTHON

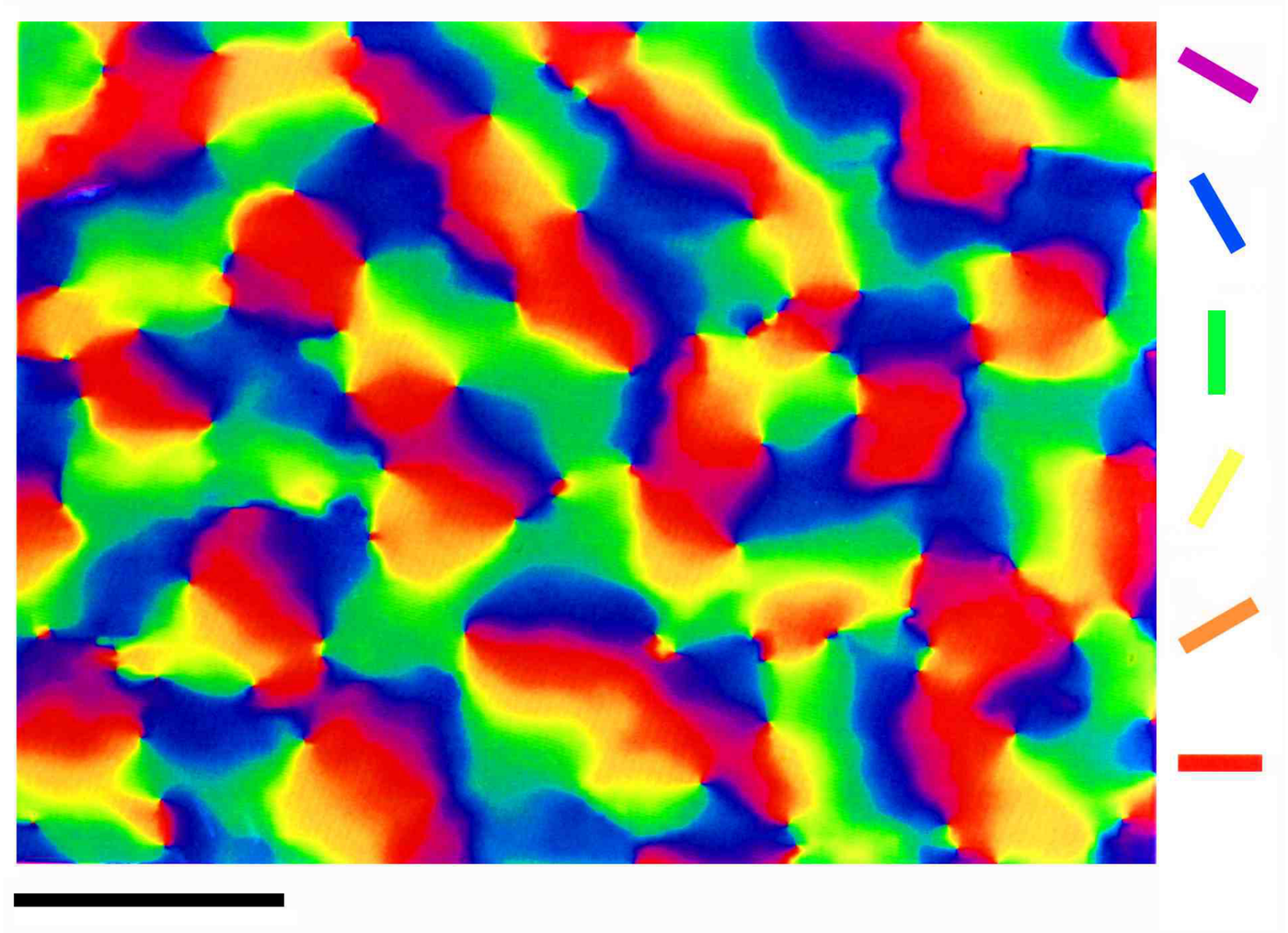
Ariel Rokem

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
```

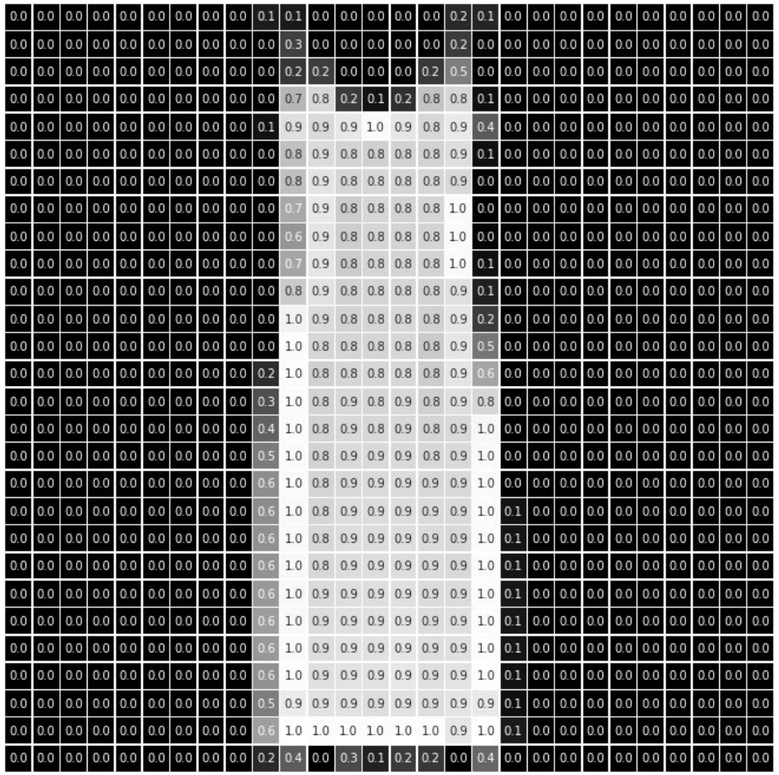
```
array([0, 0, 0, 0, 1, 0, 0, 0, 0])
```

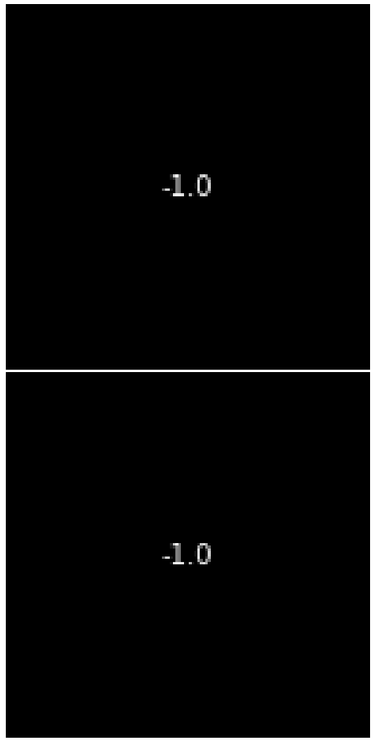
Convolution in one dimension

```
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

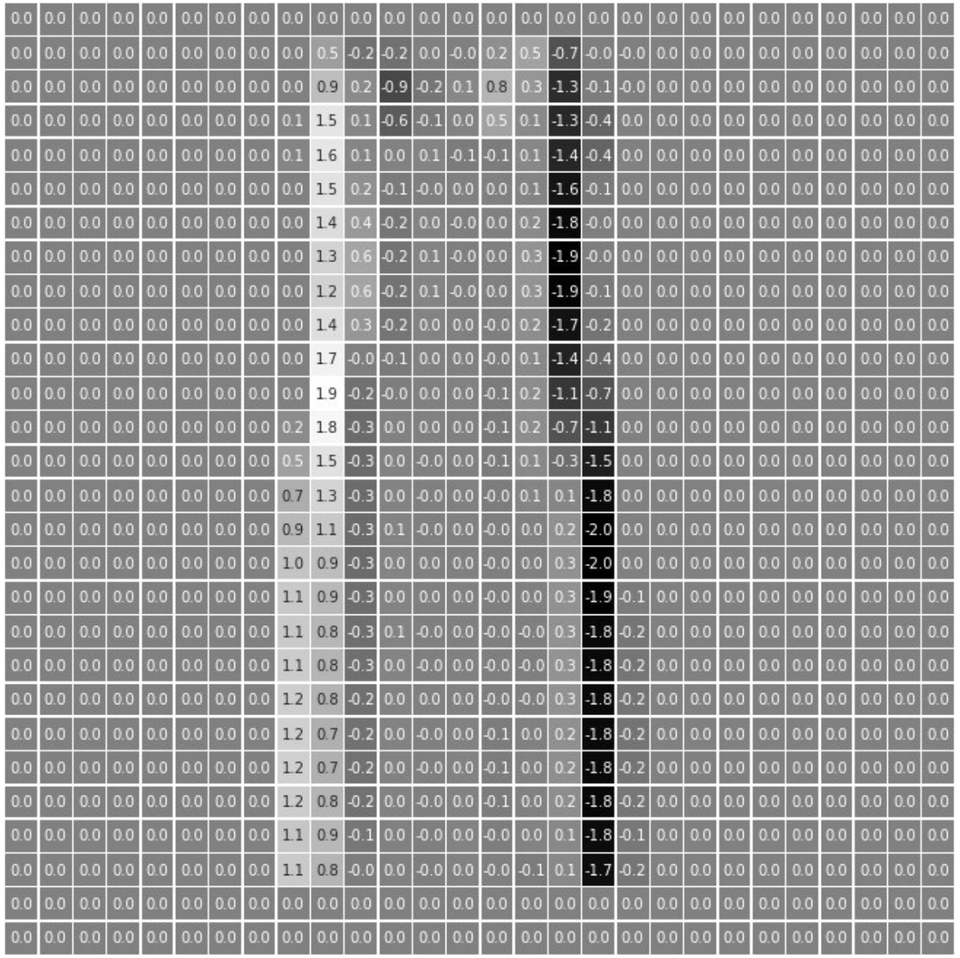




1.0

1.0

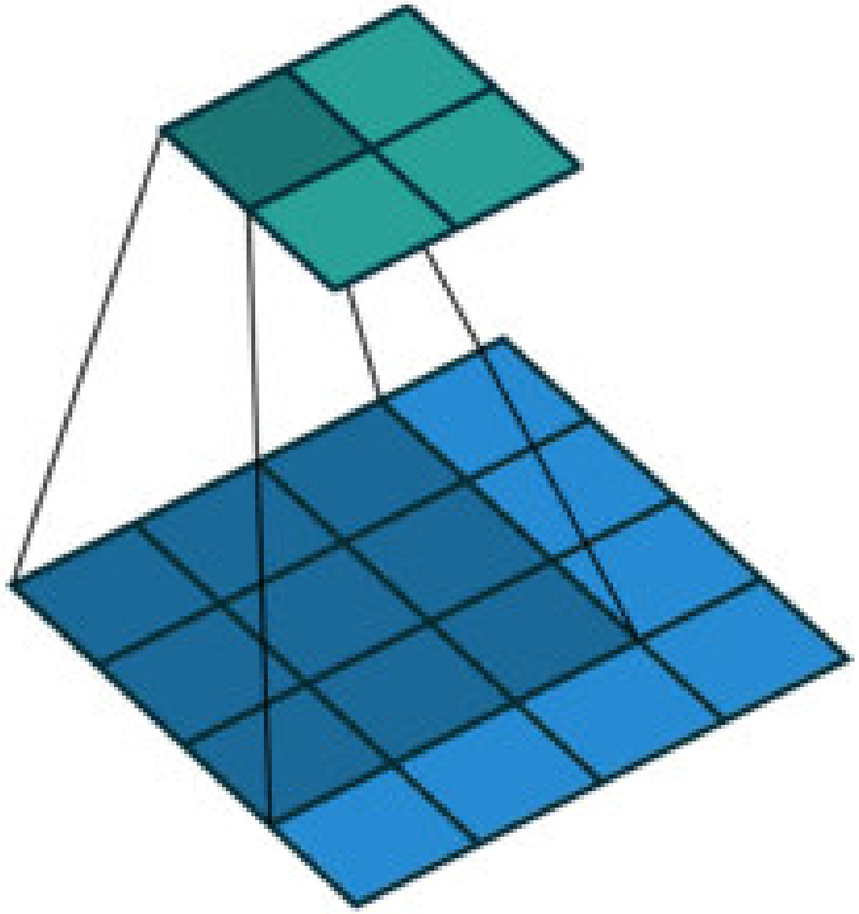
Image convolution



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

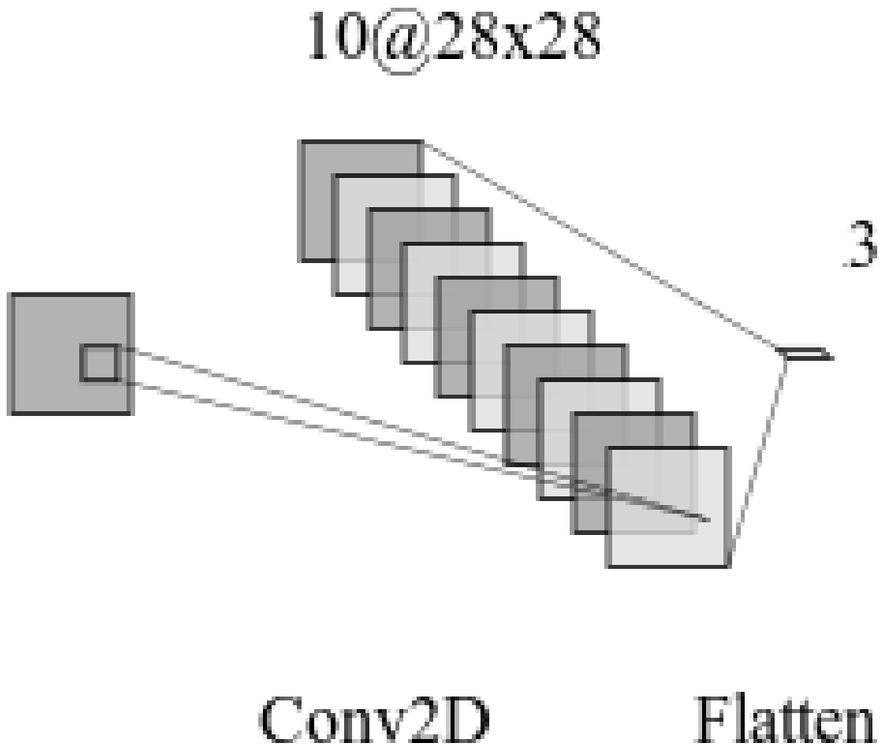
Keras Convolution layer

```
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

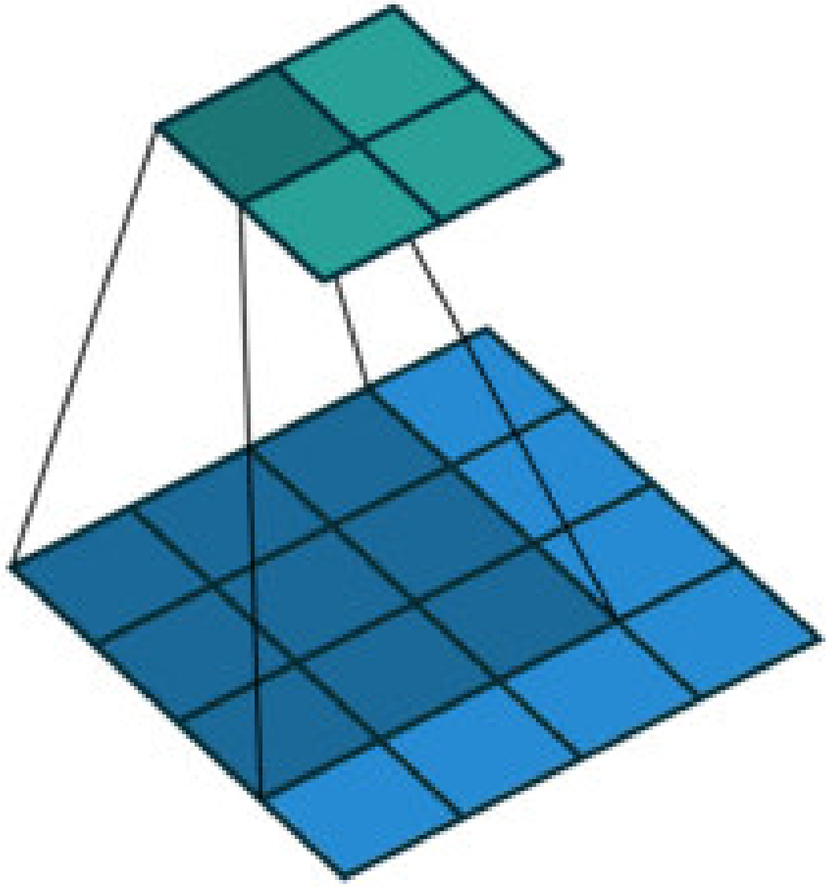
Tweaking your convolutions

IMAGE PROCESSING WITH KERAS IN PYTHON

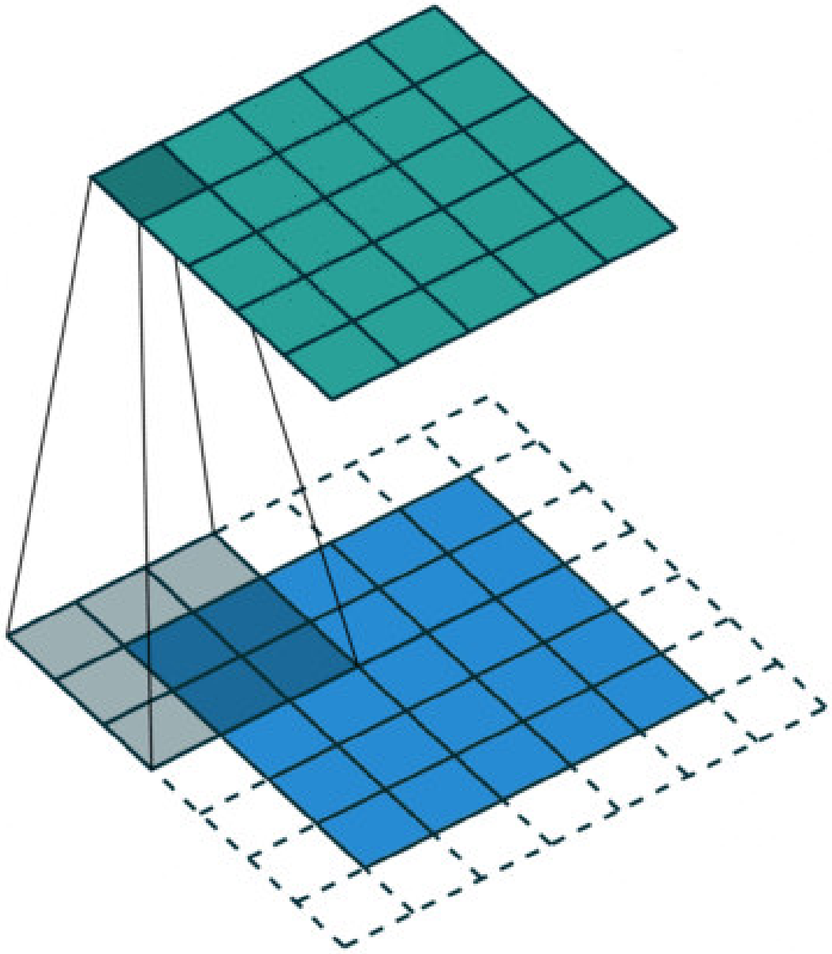
Ariel Rokem

Senior Data Scientist, University of Washington

Convolution



Convolution with zero padding



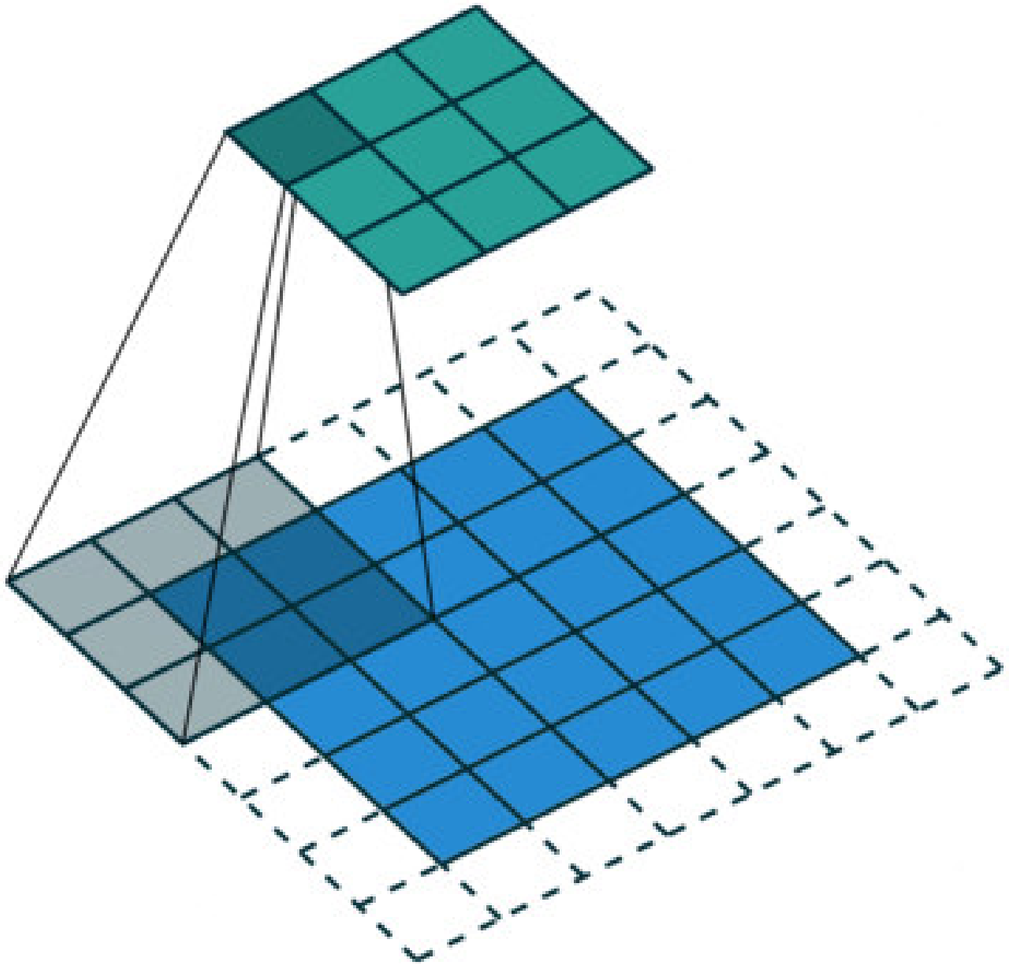
Zero padding in Keras

```
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



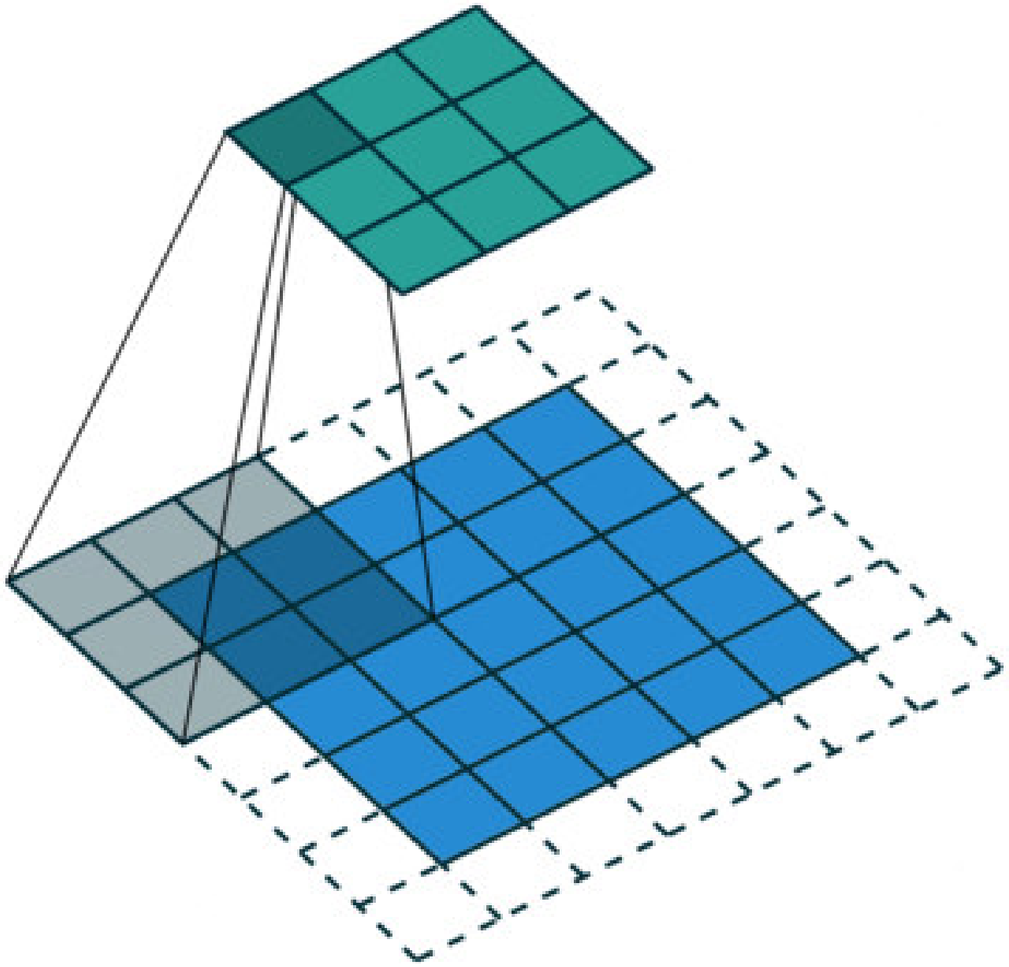
Strides in Keras

```
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

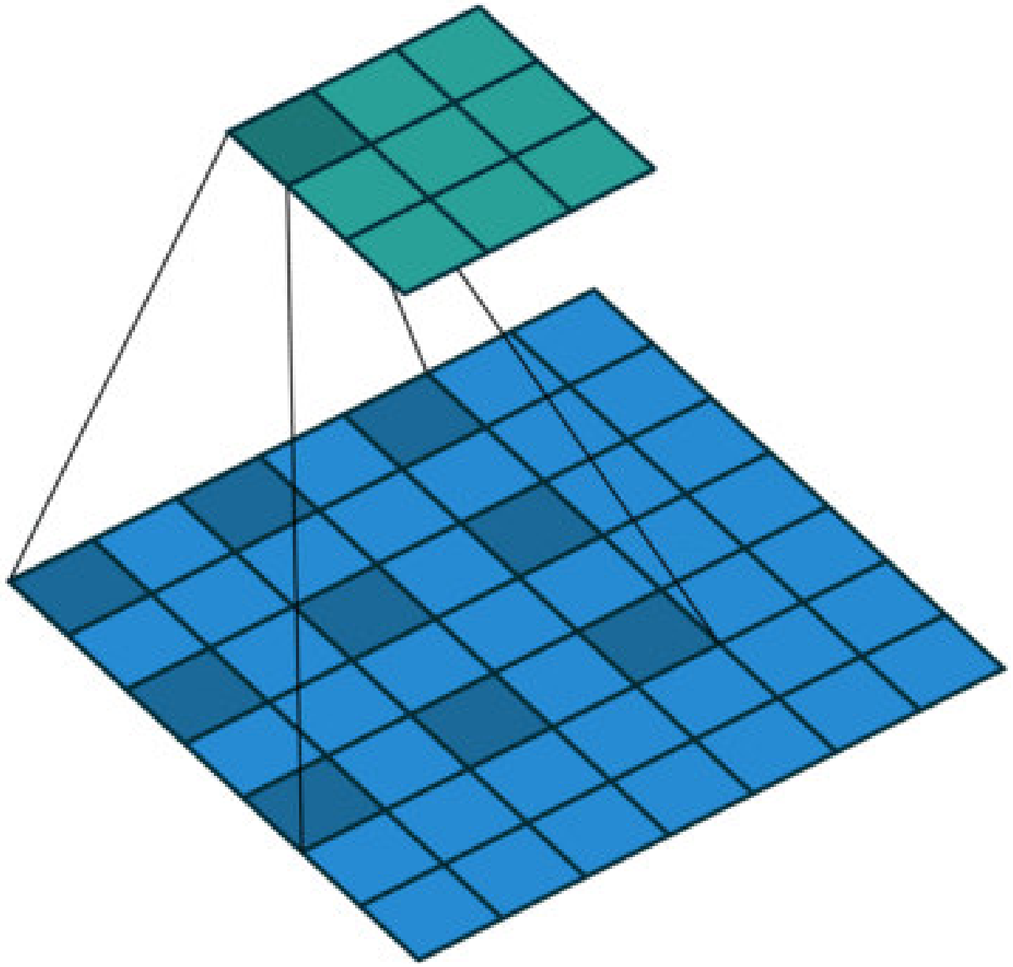
- I = size of the input
- K = size of the kernel
- P = size of the zero padding
- S = strides

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)
```

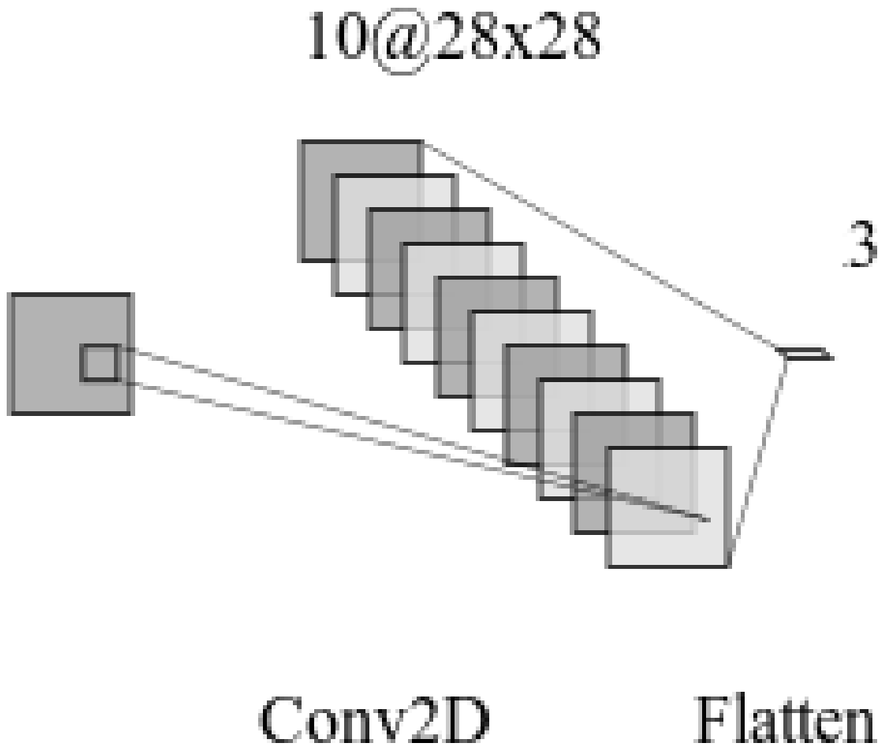
Going deeper

IMAGE PROCESSING WITH KERAS IN PYTHON

Ariel Rokem

Senior Data Scientist, University of Washington

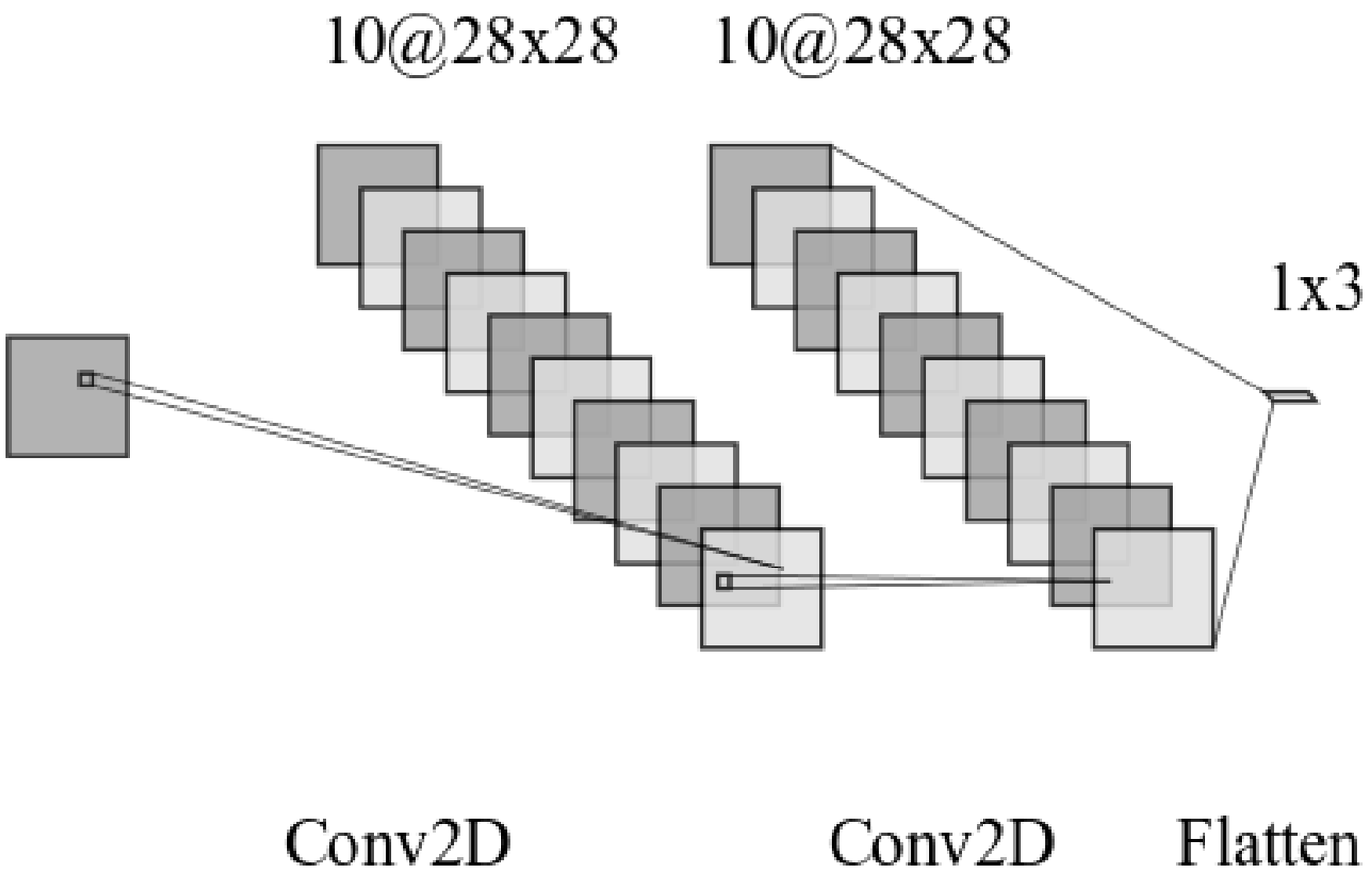
Network with one convolutional layer



Network with one convolutional layer: implementation

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

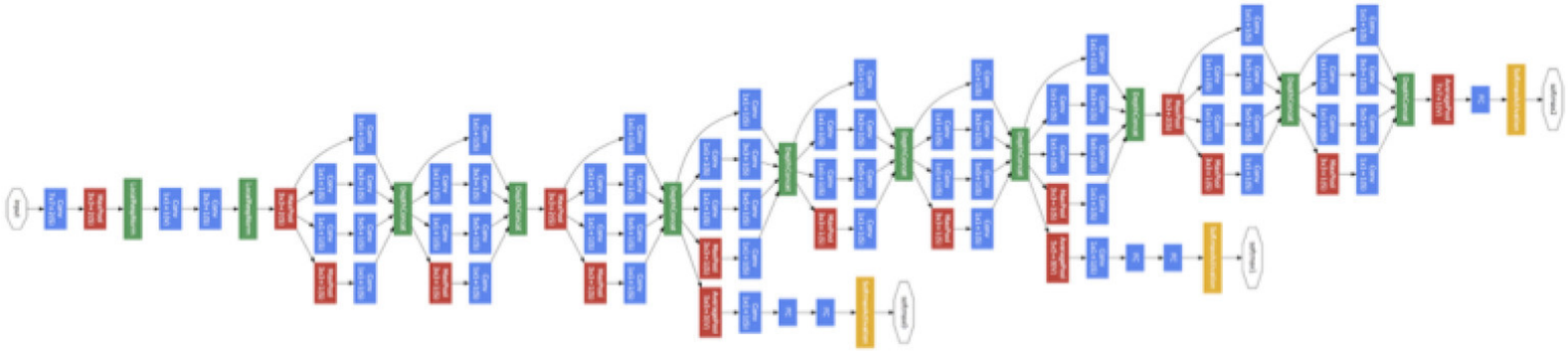
Building a deeper network



Building a deep network

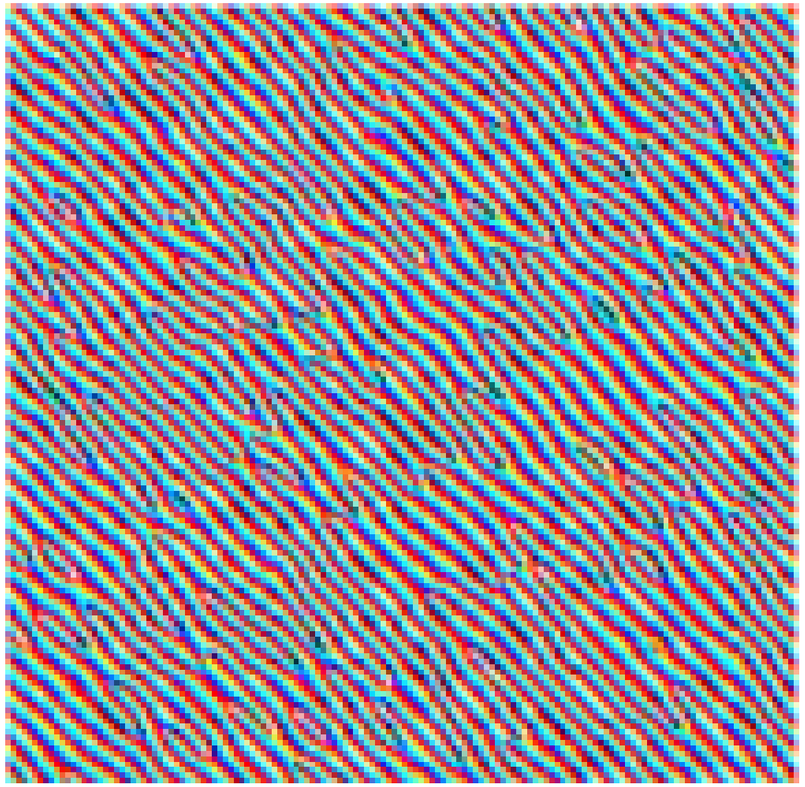
```
model = Sequential()
model.add(Conv2D(10, kernel_size=2, activation='relu',
                 input_shape=(img_rows, img_cols, 1),
                 padding='equal'))

# Second convolutional layer
model.add(Conv2D(10, kernel_size=2, activation='relu')

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

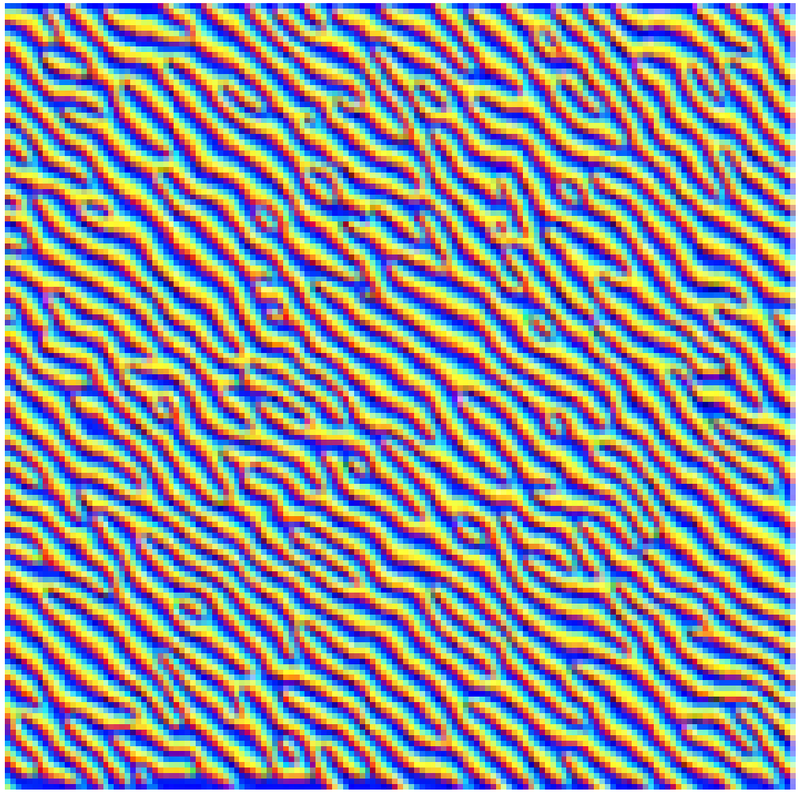
Why do we want deep networks?



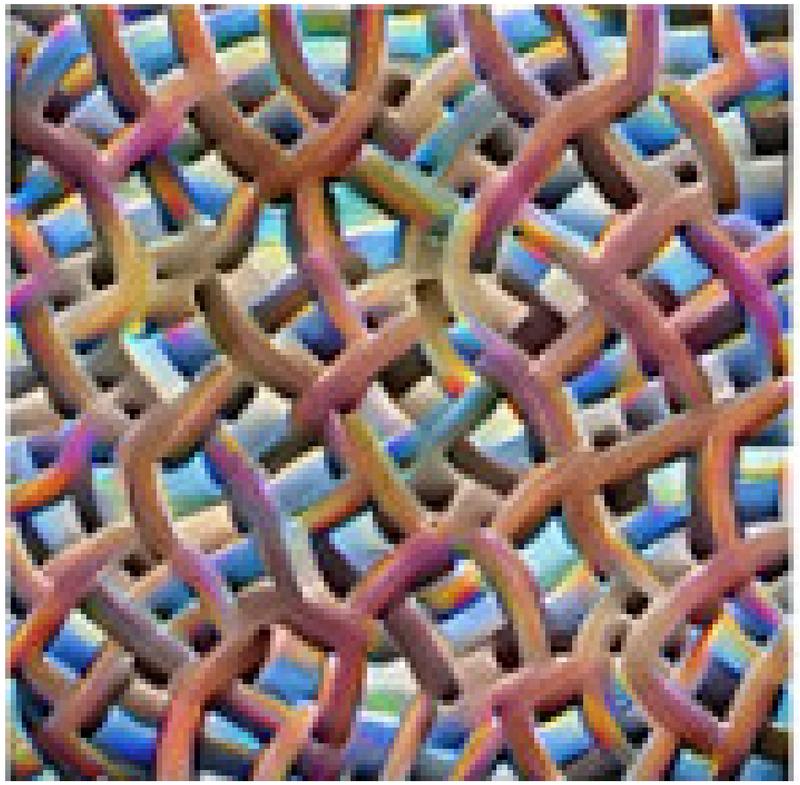
Convolution
Pooling
Softmax
Other

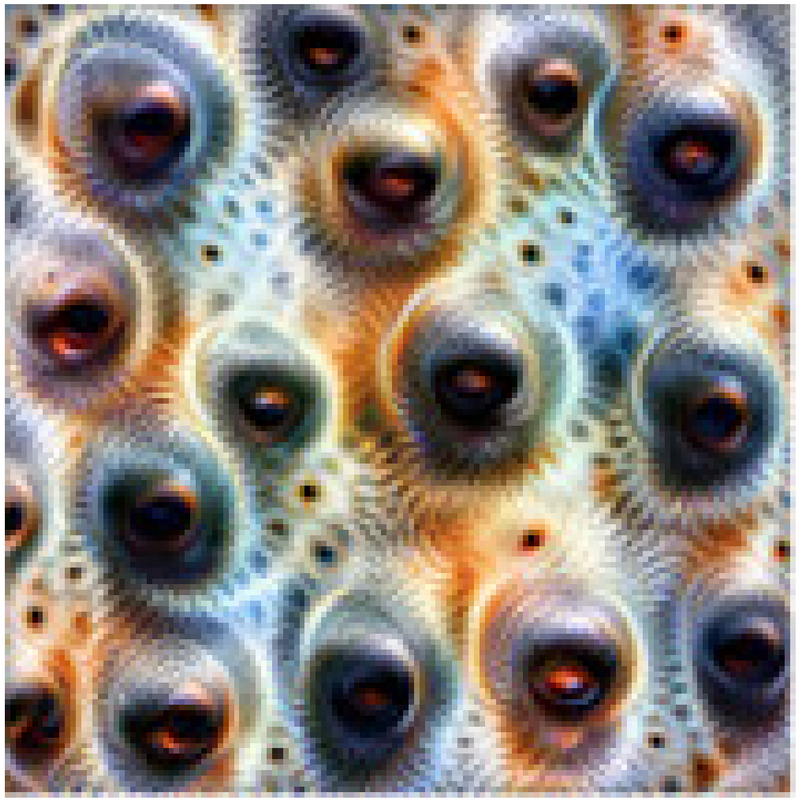
Features in early layers



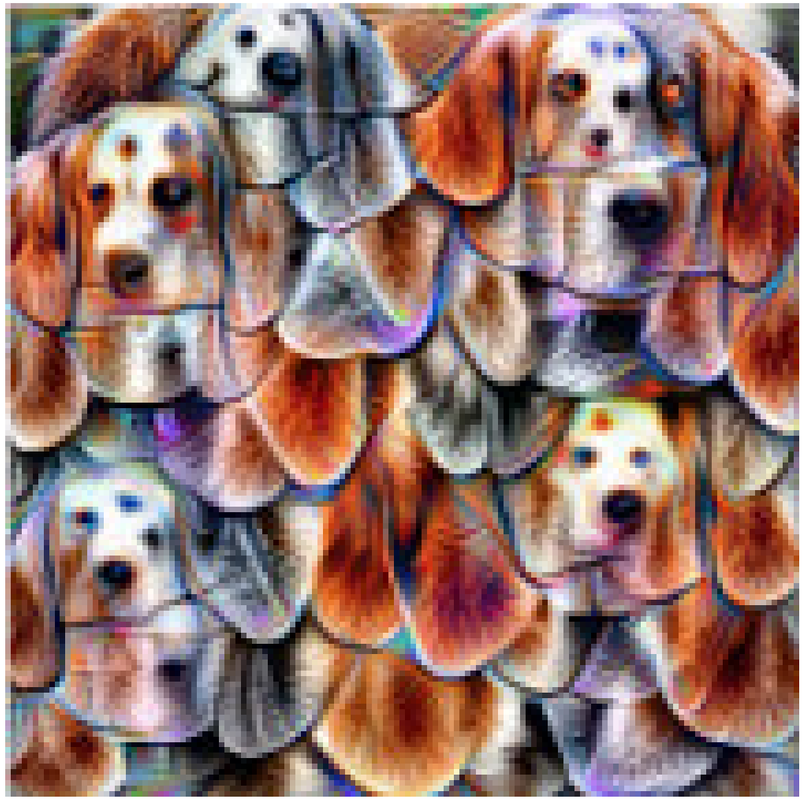


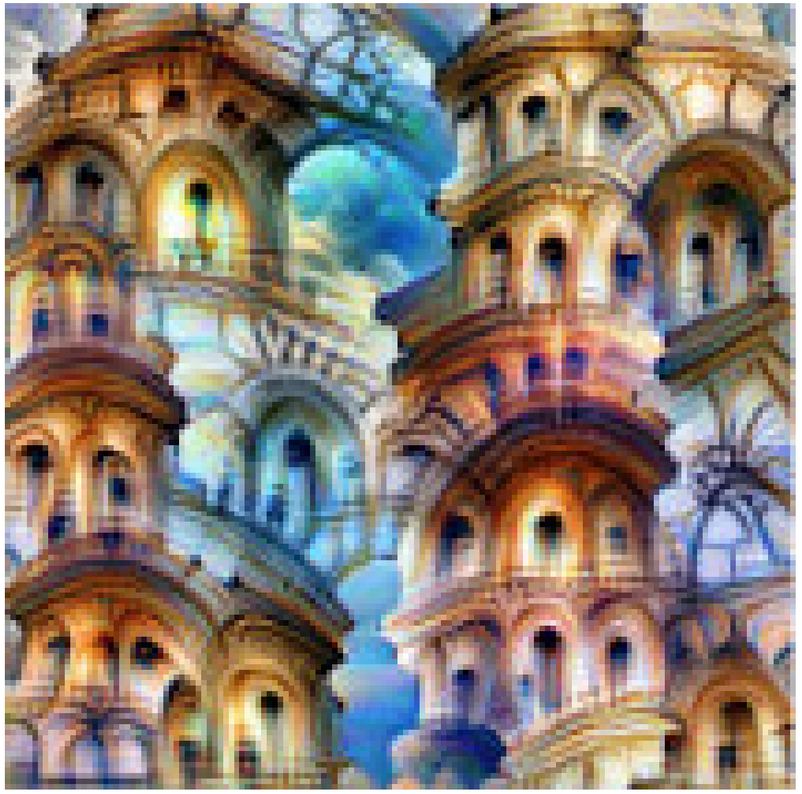
Features in intermediate layers





Features in late layers





How deep?

- Depth comes at a computational cost
- May require more data

Let's practice!

IMAGE PROCESSING WITH KERAS IN PYTHON

How many parameters?

IMAGE PROCESSING WITH KERAS IN PYTHON

Ariel Rokem

Senior Data Scientist, University of Washington

Counting parameters

```
model = Sequential()

model.add(Dense(10, activation='relu',
                input_shape=(784,)))

model.add(Dense(10, activation='relu'))

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

```
# Call the summary method
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                7850
_________________________________________________________________
dense_2 (Dense)              (None, 10)                110
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 33
=================================================================
Total params: 7,993
Trainable params: 7,993
Non-trainable params: 0
_________________________________________________________________
```

Counting parameters

```
model.add(Dense(
        10, activation='relu',
        input_shape=(784,)))
```

$$parameters = 784 * 10 + 10$$

$$= 7850$$

```
model.add(Dense(
        10, activation='relu'))
```

$$parameters = 10 * 10 + 10$$

$$= 110$$

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

$$parameters = 10 * 3 + 3$$

$$= 33$$

$$7850 + 110 + 33 = 7993$$

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 10)                7850
_________________________________________________________________
dense_2 (Dense)              (None, 10)                110
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 33
=================================================================
Total params: 7,993
Trainable params: 7,993
Non-trainable params: 0
_________________________________________________________________
```

The number of parameters in a CNN

```
model = Sequential()

model.add(Conv2D(10, kernel_size=3, activation='relu',
                 input_shape=(28, 28, 1), padding='same'))

model.add(Conv2D(10, kernel_size=3, activation='relu',
                 padding='same'))

model.add(Flatten())

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

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 10)        100
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 10)        910
_________________________________________________________________
flatten_3 (Flatten)          (None, 7840)              0
_________________________________________________________________
dense_4 (Dense)              (None, 3)                 23523
=================================================================
Total params: 24,533
Trainable params: 24,533
Non-trainable params: 0
_________________________________________________________________
```

The number of parameters in a CNN

```
model.add(
 Conv2D(10, kernel_size=3,
        activation='relu',
        input_shape=(28, 28, 1),
        padding='same'))
model.add(
 Conv2D(10, kernel_size=3,
        activation='relu',
        padding='same'))



model.add(Flatten())
```

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

$$parameters = 9 * 10 + 10$$

$$= 100$$

.

$$parameters = 10 * 9 * 10 + 10$$

$$= 910$$

$$parameters = 0$$

$$parameters = 7840 * 3 + 3$$

$$= 23523$$

Increasing the number of units in each layer

```
model = Sequential()

model.add(Dense(5, activation='relu',
                input_shape=(784,), padding='same'))

model.add(Dense(15, activation='relu', padding='same'))

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

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 5)                 3925
_________________________________________________________________
dense_2 (Dense)              (None, 15)                90
_________________________________________________________________
dense_3 (Dense)              (None, 3)                 48
=================================================================
Total params: 4,063
Trainable params: 4,063
Non-trainable params: 0
_________________________________________________________________
```

Increasing the number of units in each layer

```
model = Sequential()

model.add(Conv2D(5, kernel_size=3, activation='relu',
                 input_shape=(28, 28, 1),
                 padding="same"))

model.add(Conv2D(15, kernel_size=3, activation='relu',
                 padding="same"))

model.add(Flatten())

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

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 28, 28, 5)         50
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 28, 28, 15)        690
_________________________________________________________________
flatten_6 (Flatten)          (None, 11760)             0
_________________________________________________________________
dense_9 (Dense)              (None, 3)                 35283
=================================================================
Total params: 36,023
Trainable params: 36,023
Non-trainable params: 0
_________________________________________________________________
```

Let's practice!

IMAGE PROCESSING WITH KERAS IN PYTHON

Reducing parameters with pooling

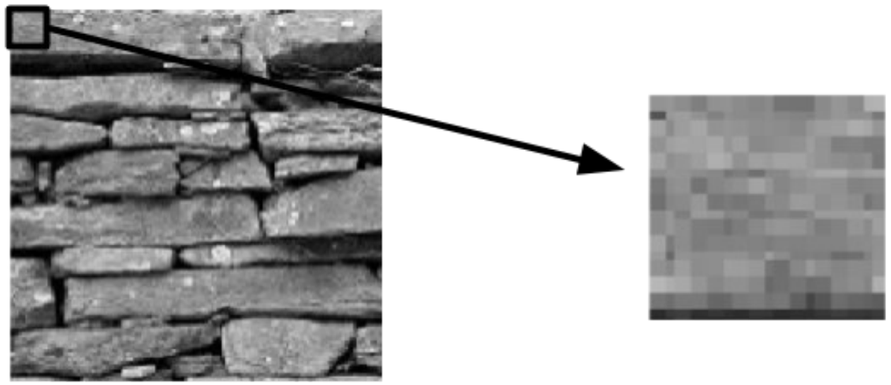
IMAGE PROCESSING WITH KERAS IN PYTHON

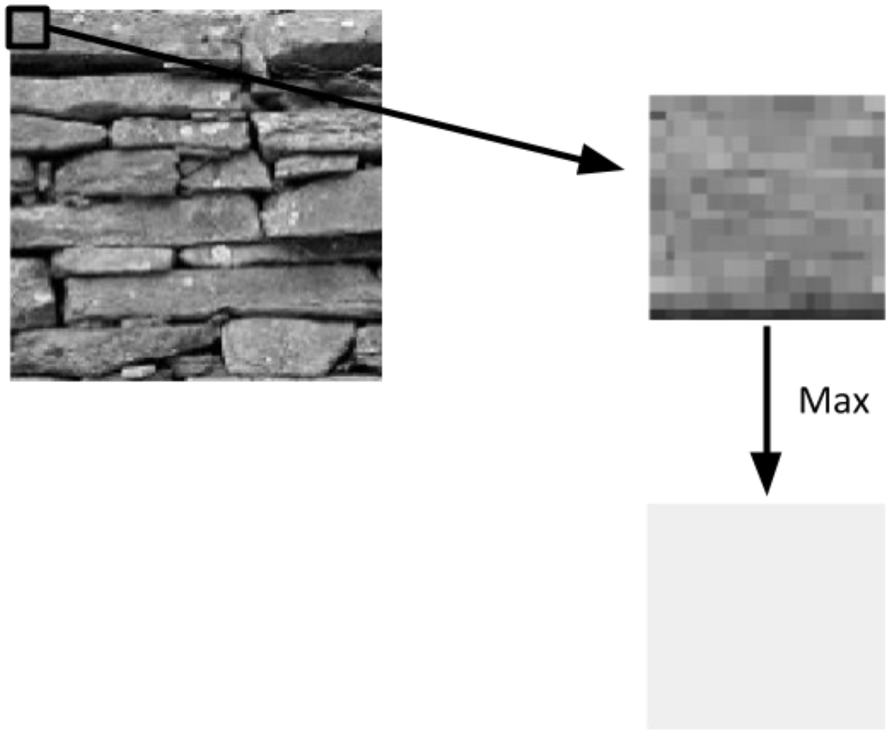
Ariel Rokem

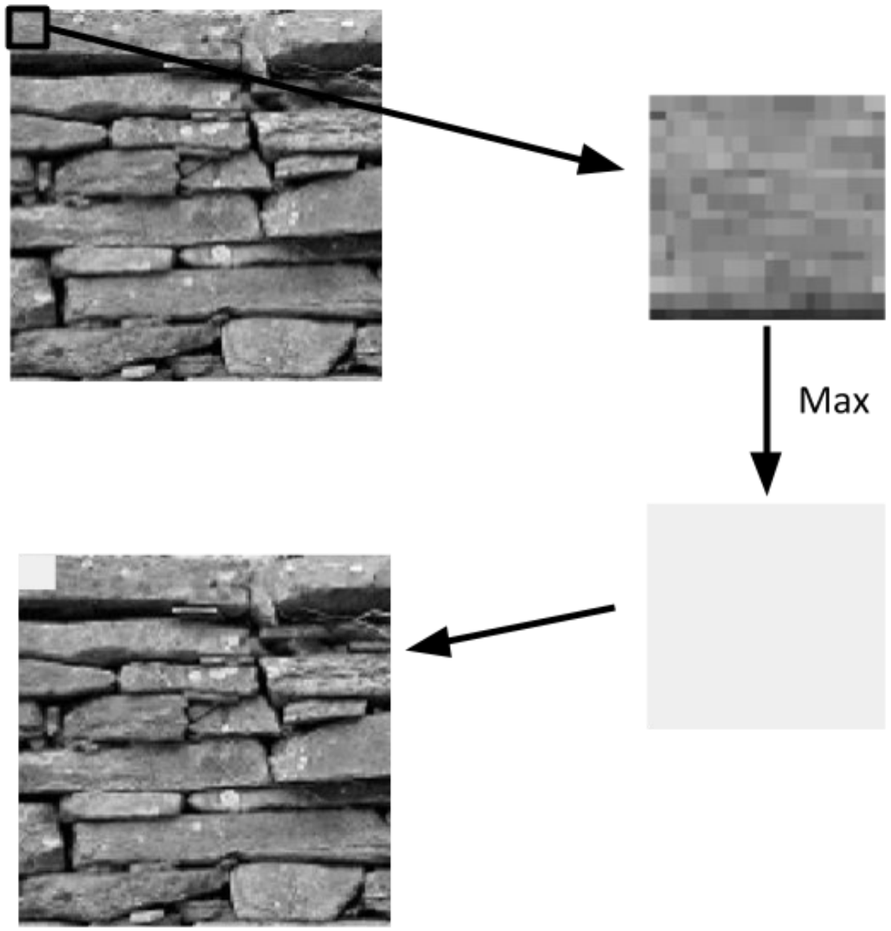
Senior Data Scientist, University of Washington

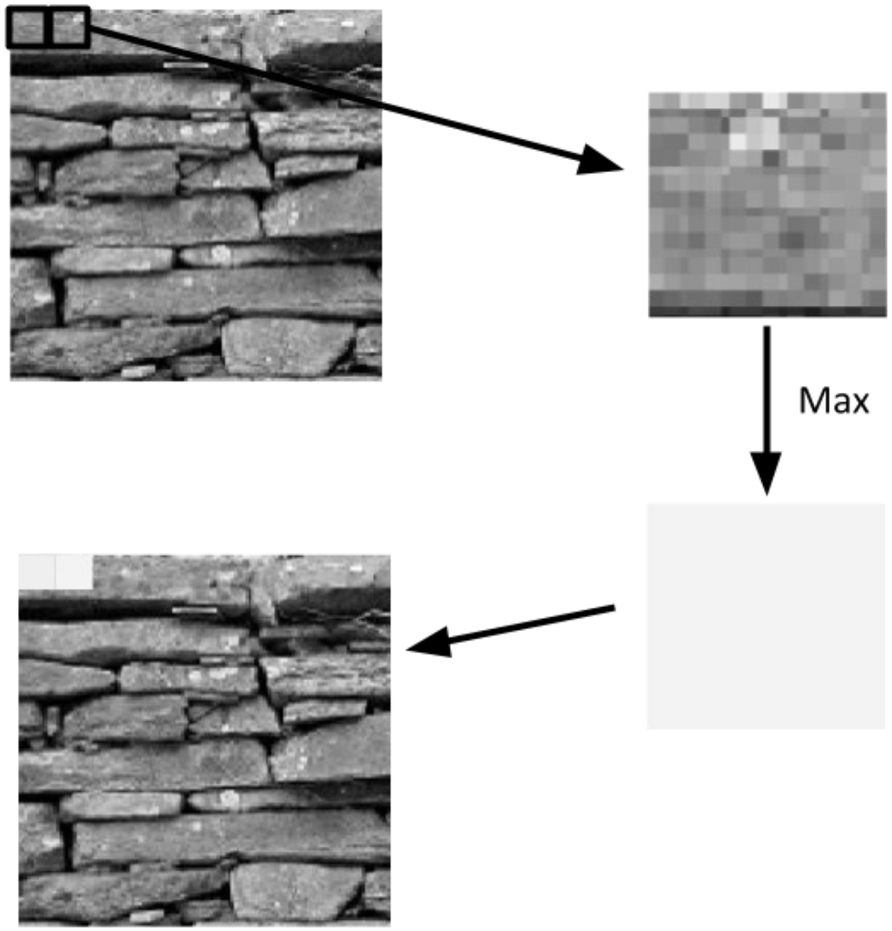
```
model.summary()
```

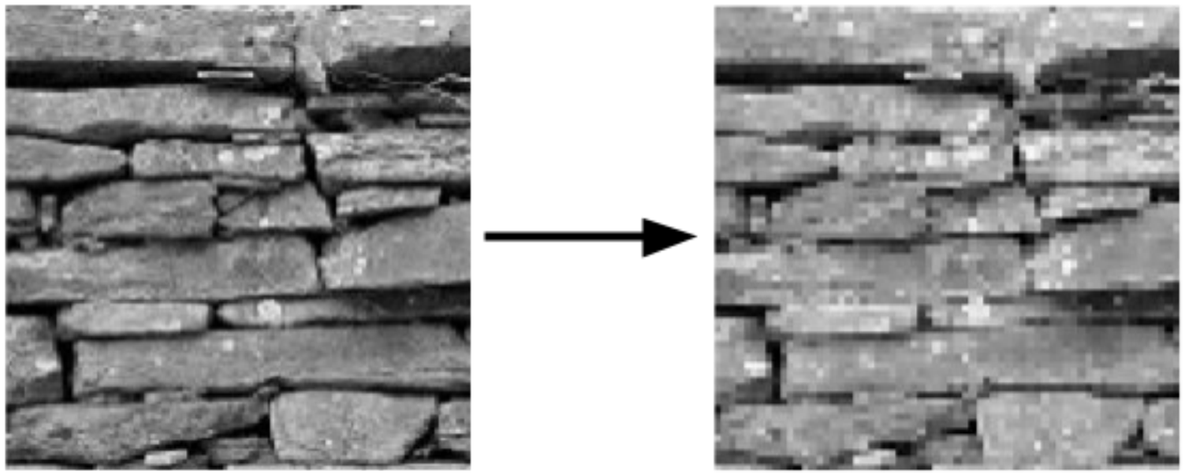
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_12 (Conv2D)           (None, 28, 28, 5)         50
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 28, 28, 15)        690
_________________________________________________________________
flatten_6 (Flatten)          (None, 11760)             0
_________________________________________________________________
dense_9 (Dense)              (None, 3)                 35283
=================================================================
Total params: 36,023
Trainable params: 36,023
Non-trainable params: 0
_________________________________________________________________
```




Max


Max


Max



Implementing max pooling

```
result = np.zeros((im.shape[0]//2, im.shape[1]//2))
result[0, 0] = np.max(im[0:2, 0:2])
result[0, 1] = np.max(im[0:2, 2:4])
result[0, 2] = np.max(im[0:2, 4:6])
```

...

```
result[1, 0] = np.max(im[2:4, 0:2])
result[1, 1] = np.max(im[2:4, 2:4])
```

...

Implementing max pooling

```
for ii in range(result.shape[0]):
    for jj in range(result.shape[1]):
        result[ii, jj] = np.max(im[ii*2:ii*2+2, jj*2:jj*2+2
```

Max pooling in Keras

```
from keras.models import Sequential
from keras.layers import Dense, Conv2D, Flatten, MaxPool2D

model = Sequential()
model.add(Conv2D(5, kernel_size=3, activation='relu',
                 input_shape=(img_rows, img_cols, 1)))

model.add(MaxPool2D(2))

model.add(Conv2D(15, kernel_size=3, activation='relu',
                 input_shape=(img_rows, img_cols, 1)))

model.add(MaxPool2D(2))

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

```
model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 5)         50
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 5)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 15)        690
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 15)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 375)               0
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 1128
=================================================================
```