

Going deeper

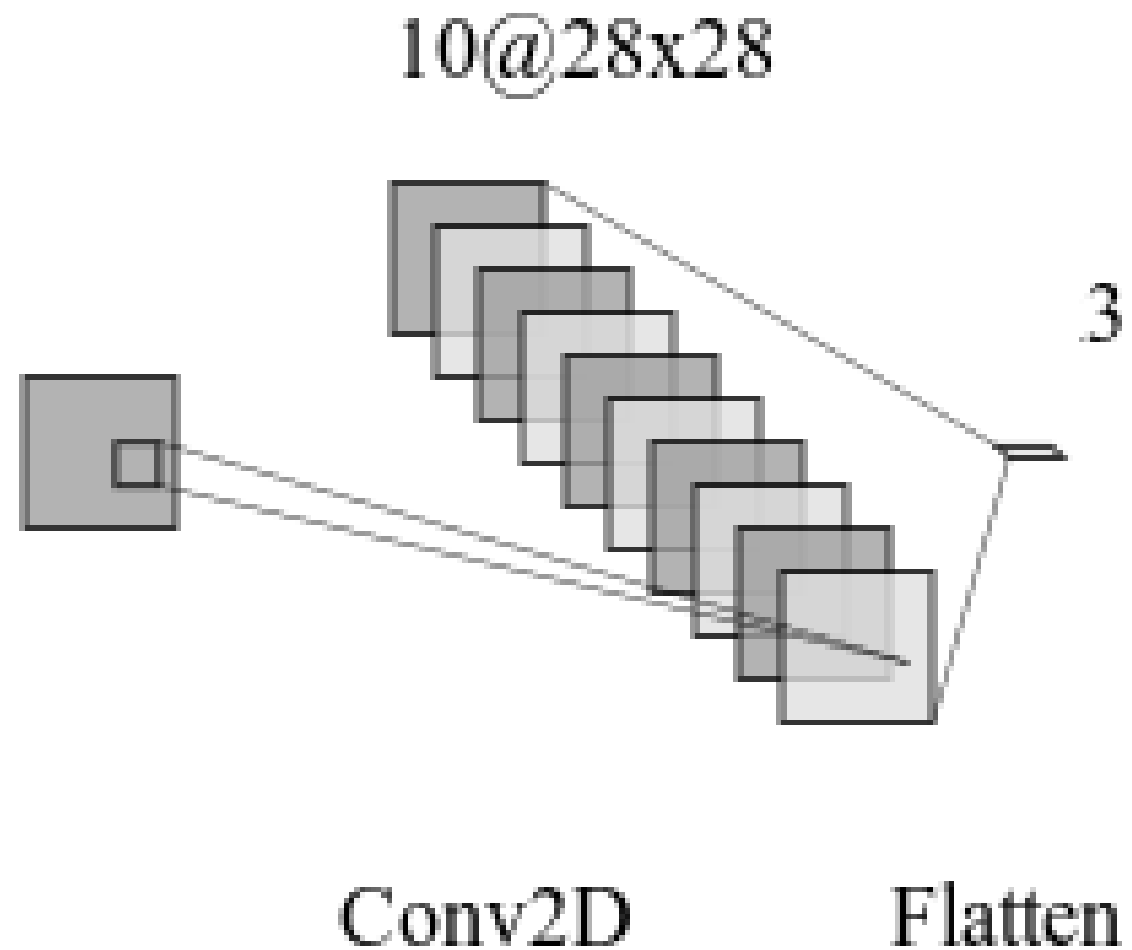
IMAGE PROCESSING WITH KERAS IN PYTHON



Ariel Rokem

Senior Data Scientist, University of
Washington

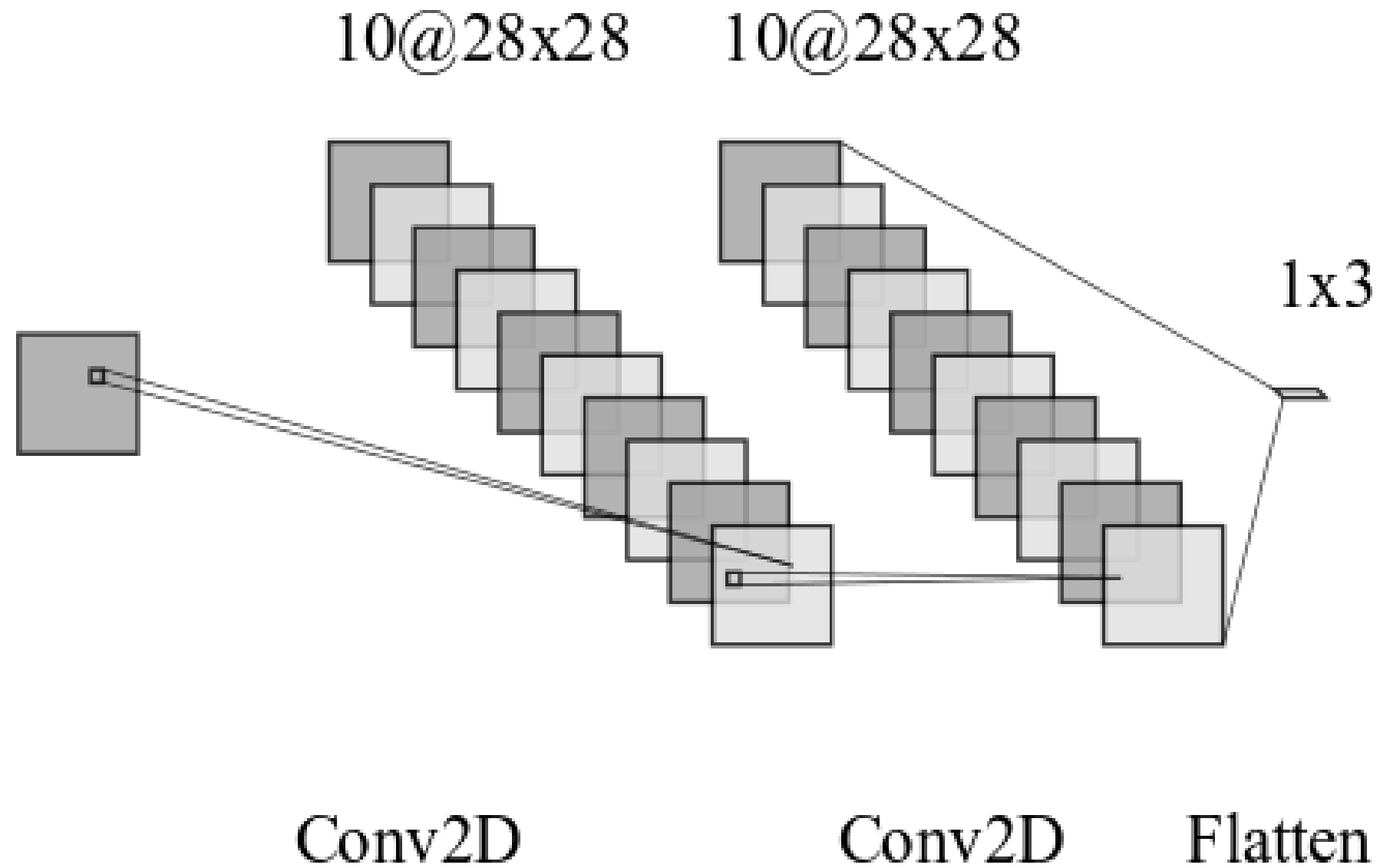
Network with one convolutional layer



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

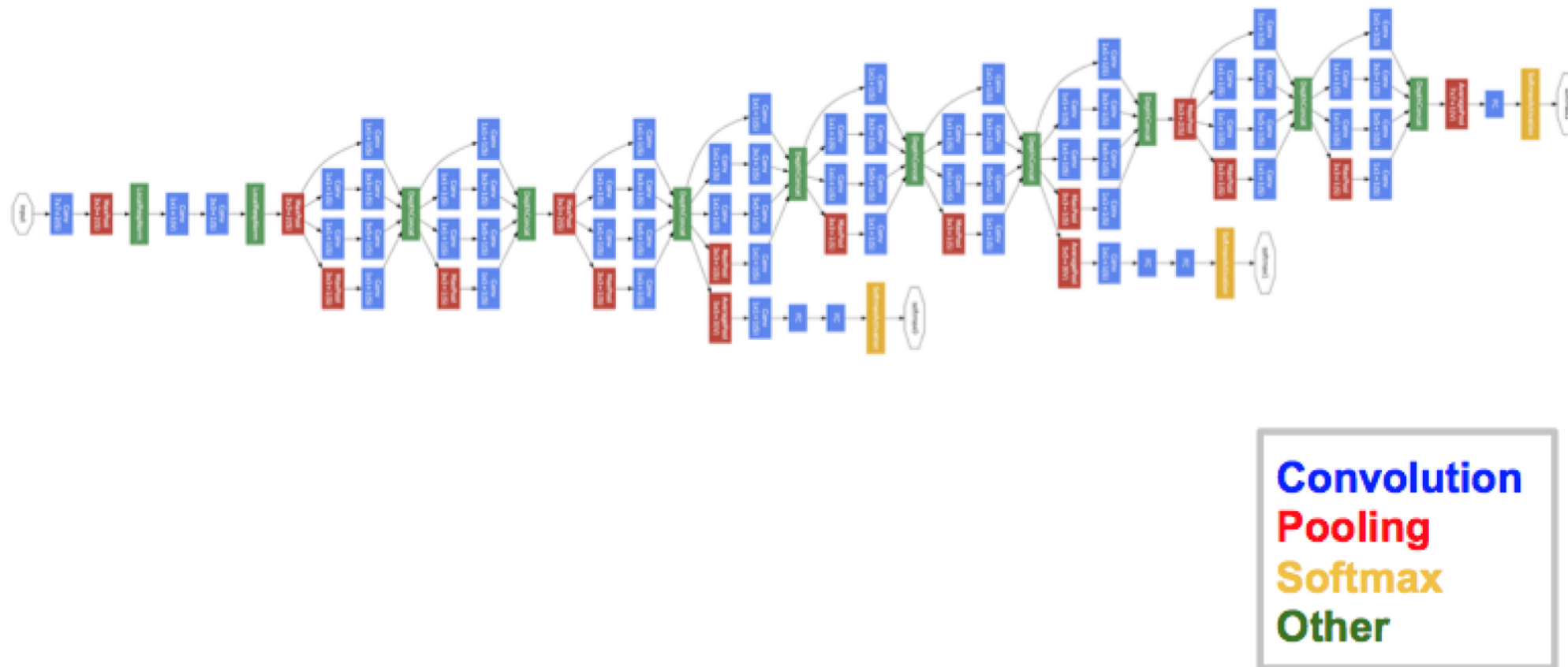


Building a deep network

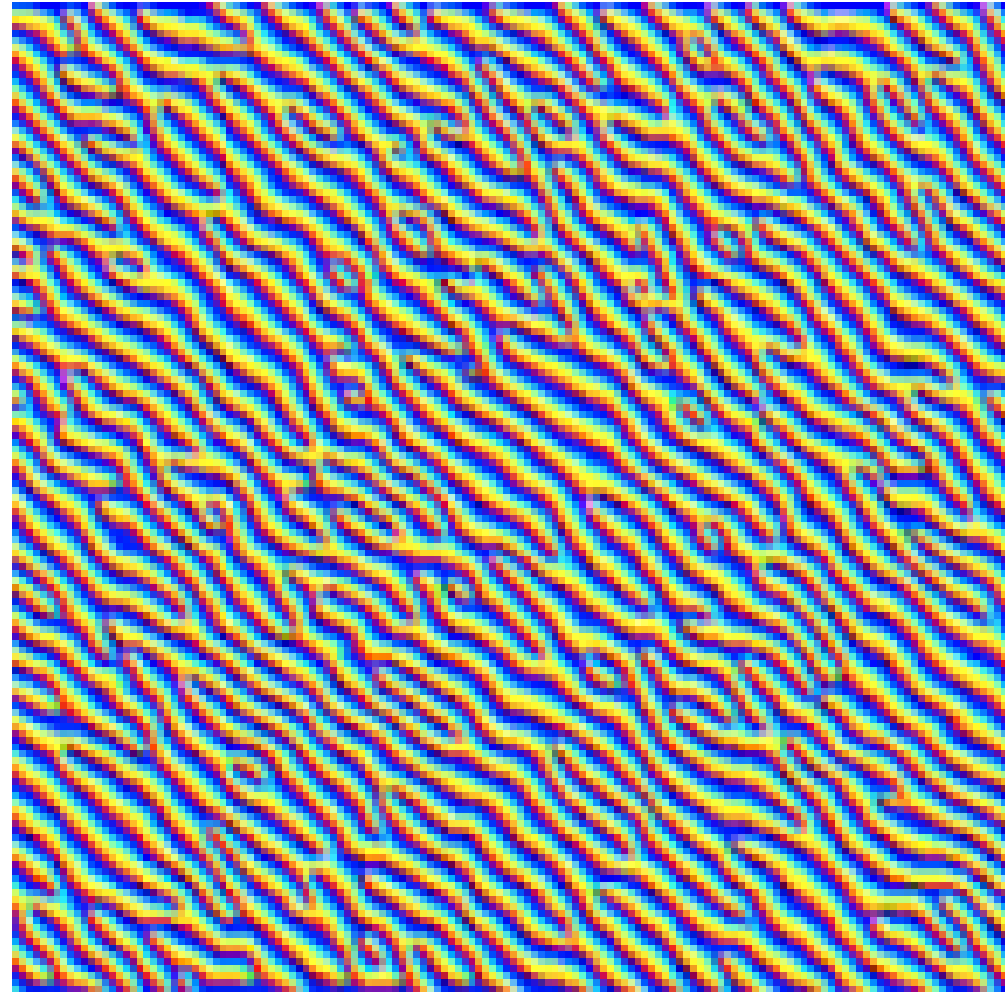
```
model = Sequential()
model.add(Conv2D(10, kernel_size=2, activation='relu',
                 input_shape=(img_rows, img_cols, 1),
                 padding='equal'))

# Second convolutional layer
model.add(Conv2D(10, kernel_size=2, activation='relu'))
model.add(Flatten())
model.add(Dense(3, activation='softmax'))
```

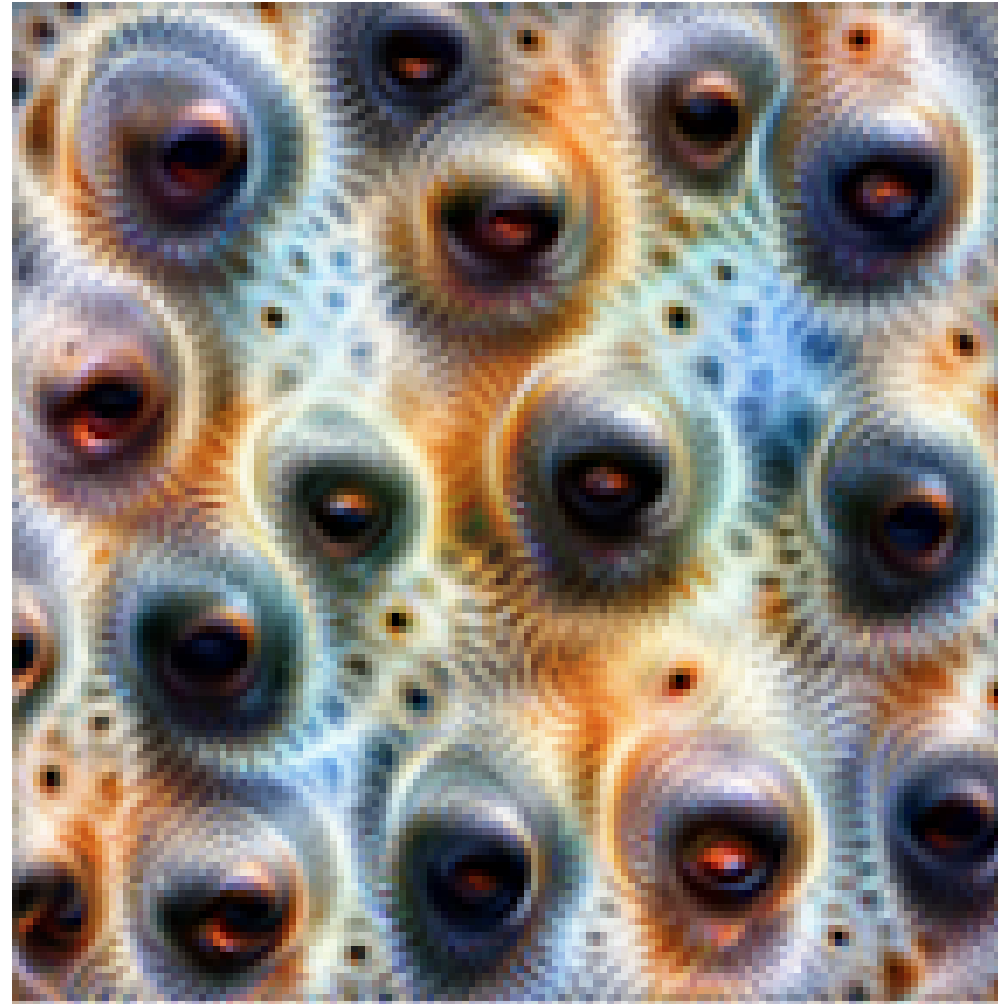
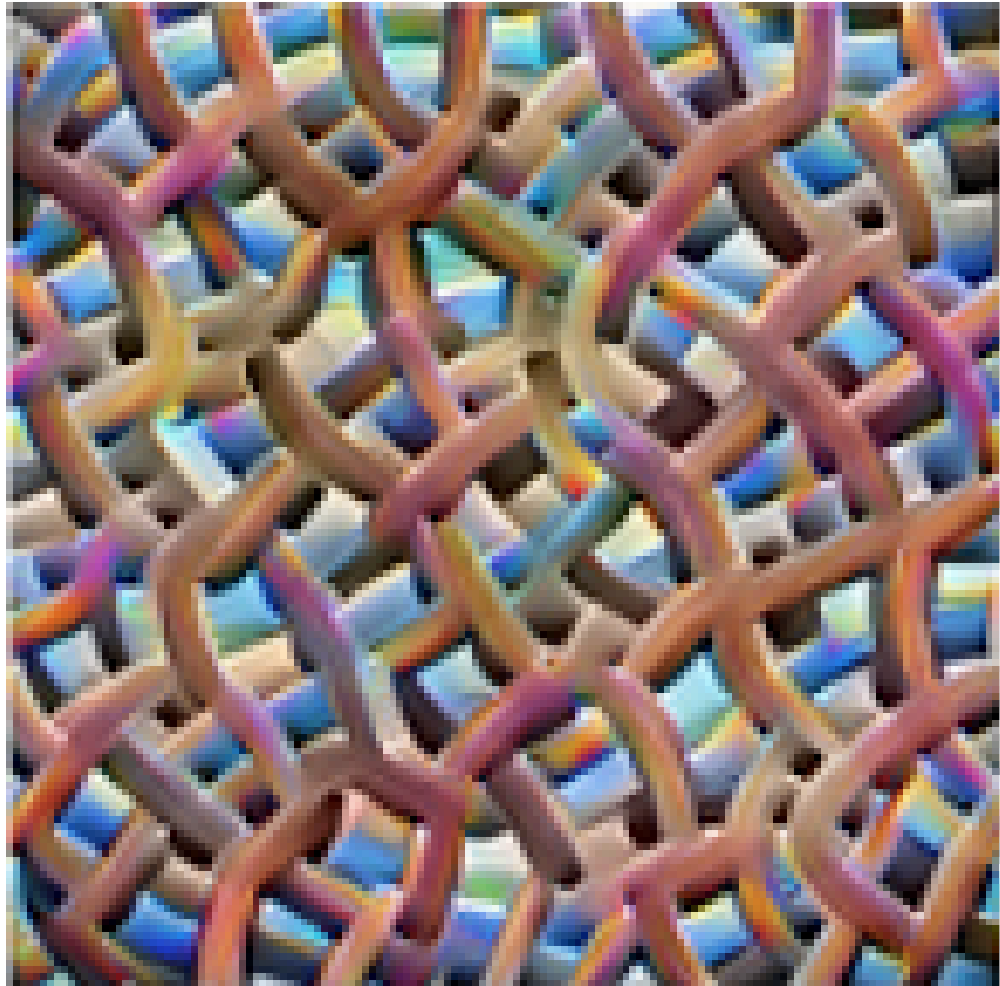
Why do we want deep networks?



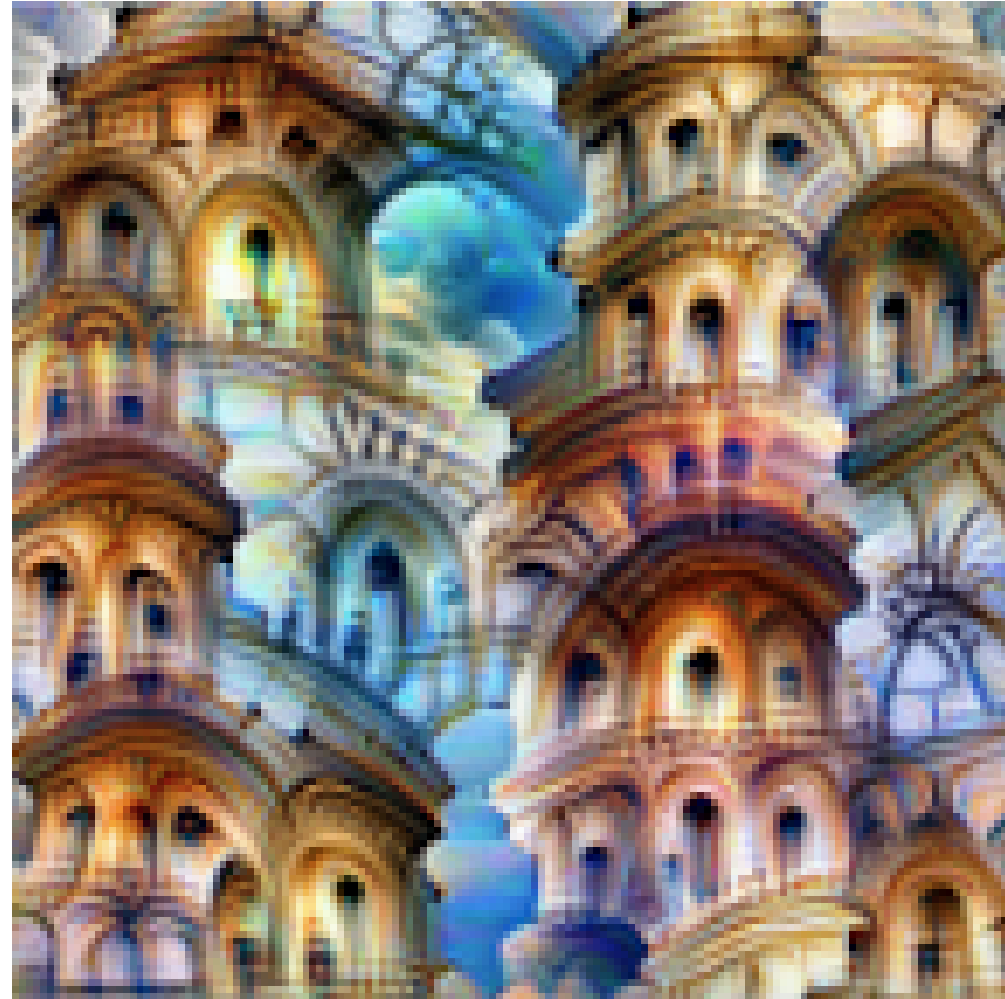
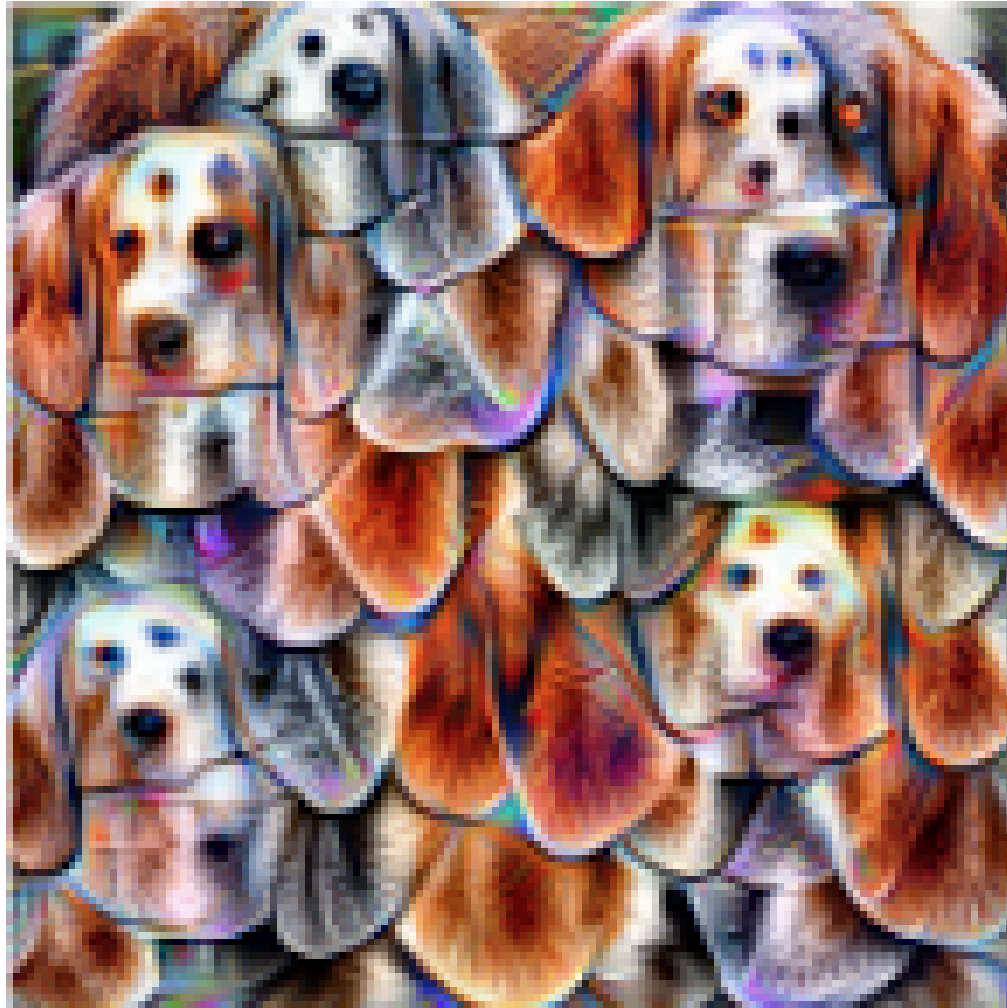
Features in early layers



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,)))
```

$$\begin{aligned} \text{parameters} &= 784 * 10 + 10 \\ &= 7850 \end{aligned}$$

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

$$\begin{aligned} \text{parameters} &= 10 * 10 + 10 \\ &= 110 \end{aligned}$$

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

$$\begin{aligned} \text{parameters} &= 10 * 3 + 3 \\ &= 33 \end{aligned}$$

$$7850 + 110 + 33 = 7993$$

```
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'))
```

$$parameters = 9 * 10 + 10$$

$$= 100$$

.

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

$$= 910$$

$$parameters = 0$$

$$parameters = 7840 * 3 + 3$$

$$= 23523$$

$$100 + 910 + 0 + 23523 = 24533$$

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



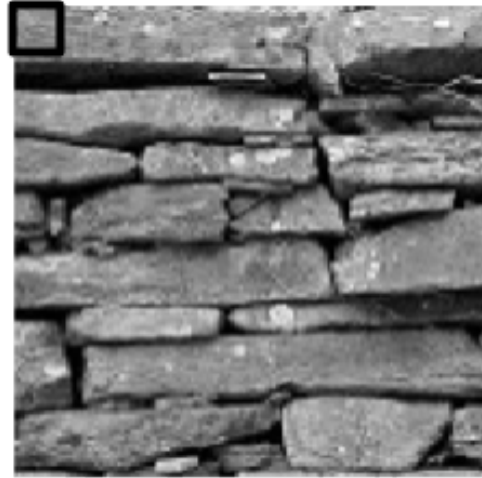
Ariel Rokem

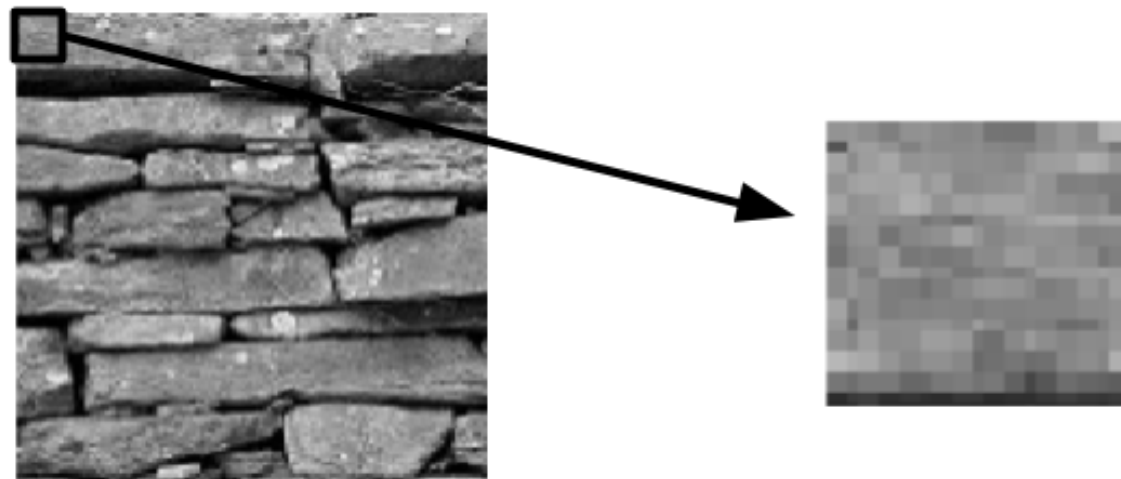
Senior Data Scientist, University of
Washington

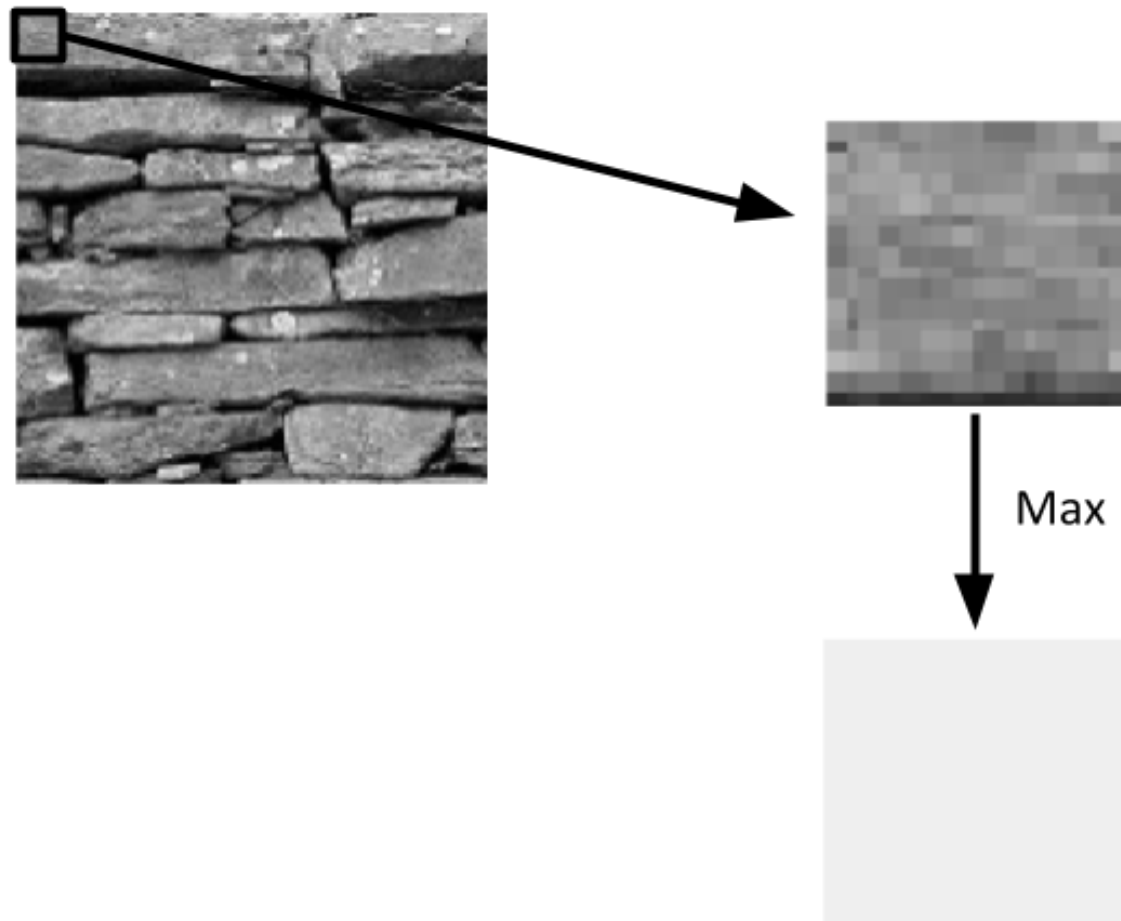
```
model.summary()
```

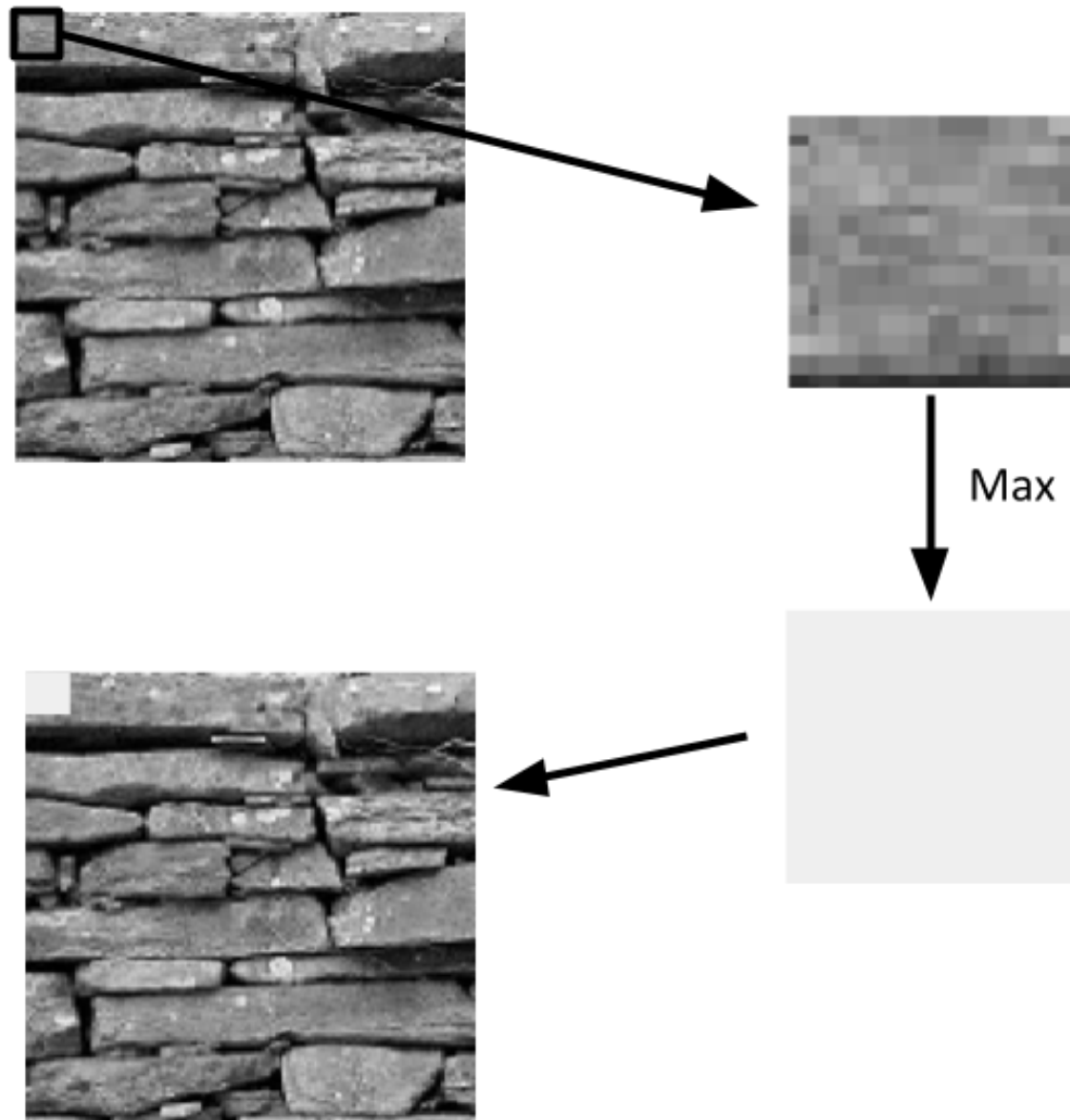
```
-----  
Layer (type)                 Output Shape                 Param #  
-----  
conv2d_12 (Conv2D)           (None, 28, 28, 5)           50  
-----  
conv2d_13 (Conv2D)           (None, 28, 28, 15)          690  
-----  
flatten_6 (Flatten)          (None, 11760)                0  
-----  
dense_9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

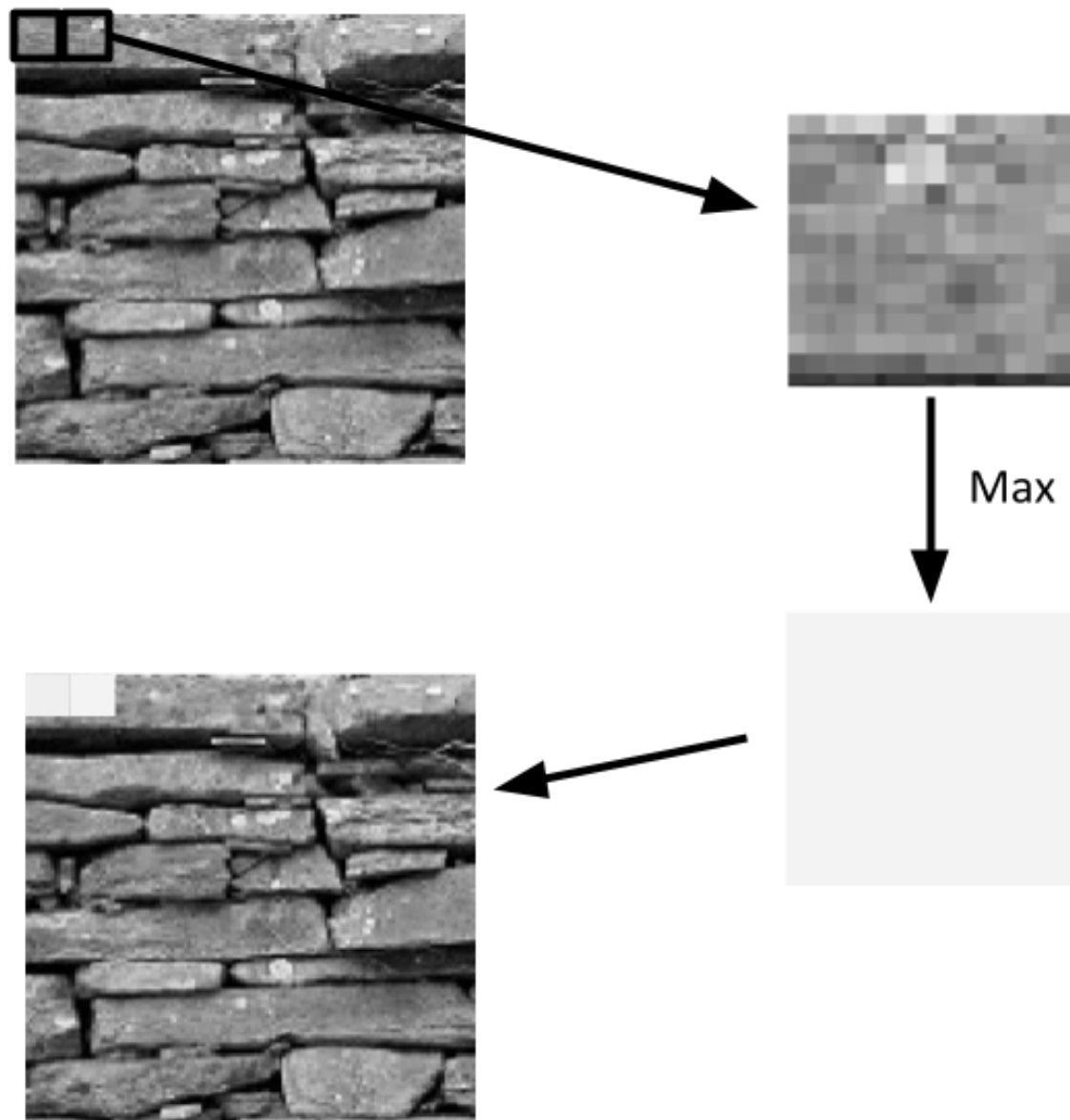














Implementing max pooling

```
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
=====
```

Let's practice!

IMAGE PROCESSING WITH KERAS IN PYTHON