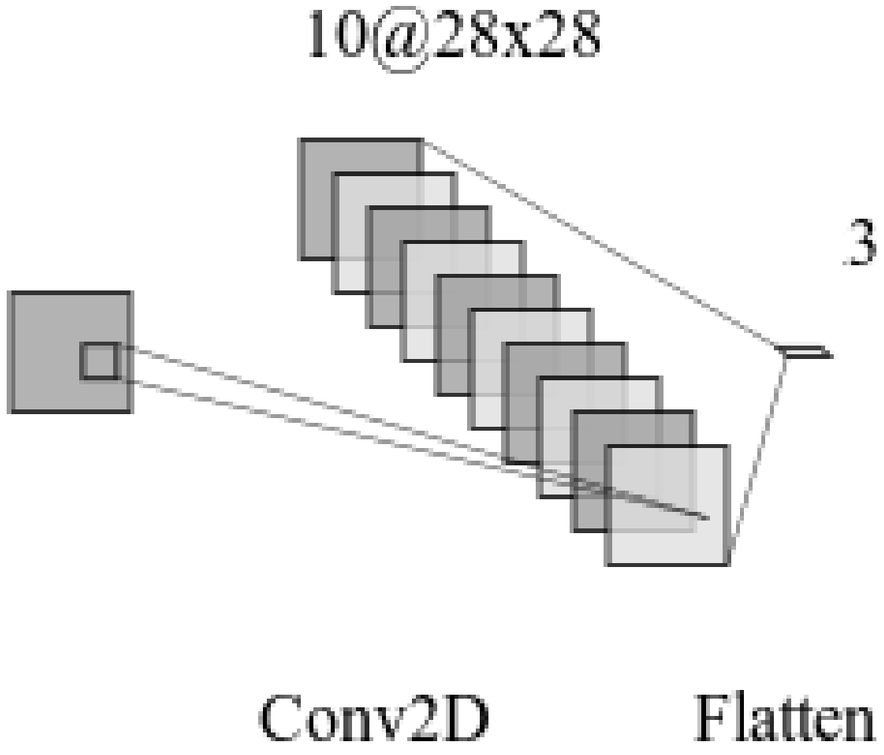
Going deeper

IMAGE PROCESSING WITH KERAS IN PYTHON

Ariel Rokem

Senior Data Scientist, University of Washington

Network with one convolutional layer



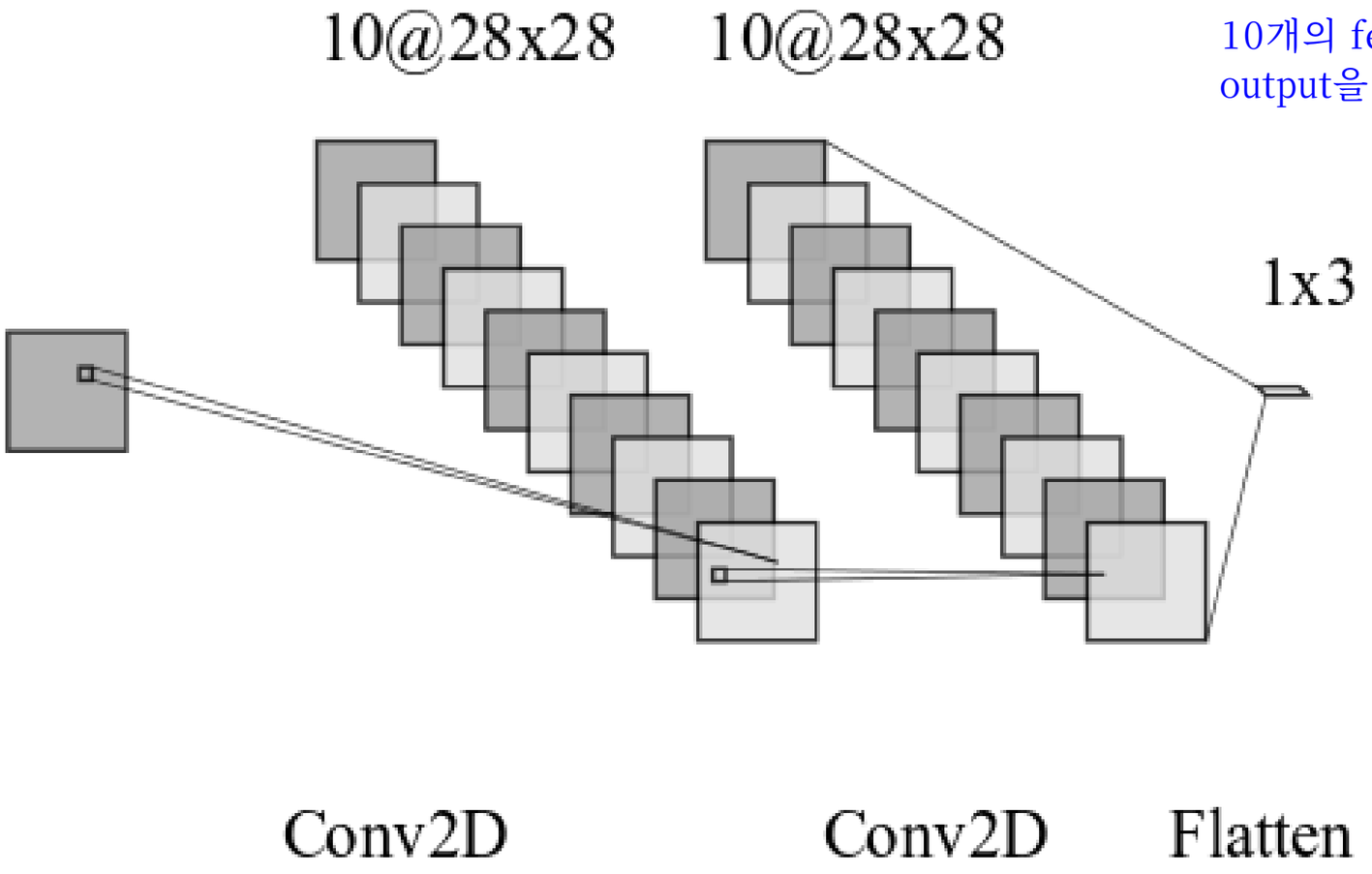
하나의 컨볼루션 레이어를 가지고 있고,
이어서 3개의 유닛으로 완전히 연결된 layer로 flatten을 시킨 것입니다.

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'))
```

네트워크를 구현하는 코드가 있고

Building a deeper network



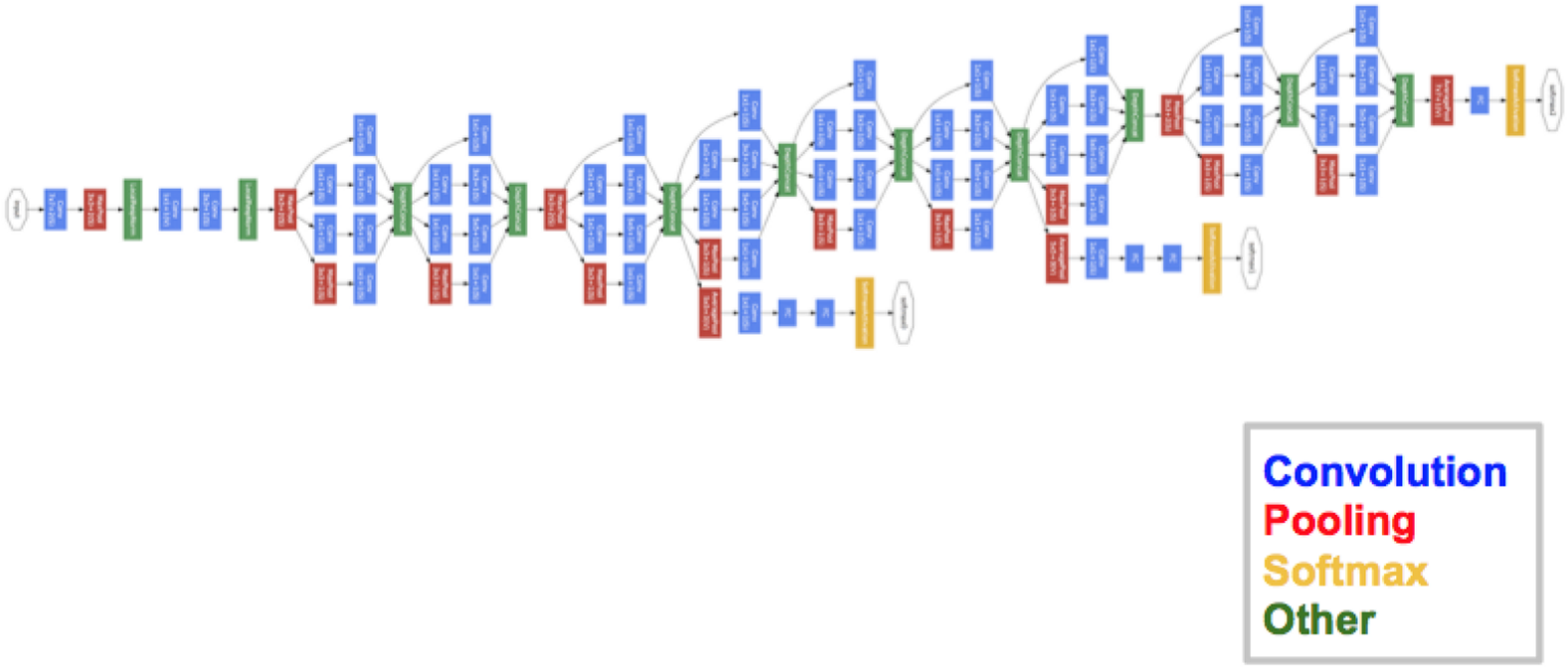
10개의 feature map을 추가하고 flatten하게 만든 다음 flatten된 output을 3unit의 연결layer로 전달합니다.

Building a deep network

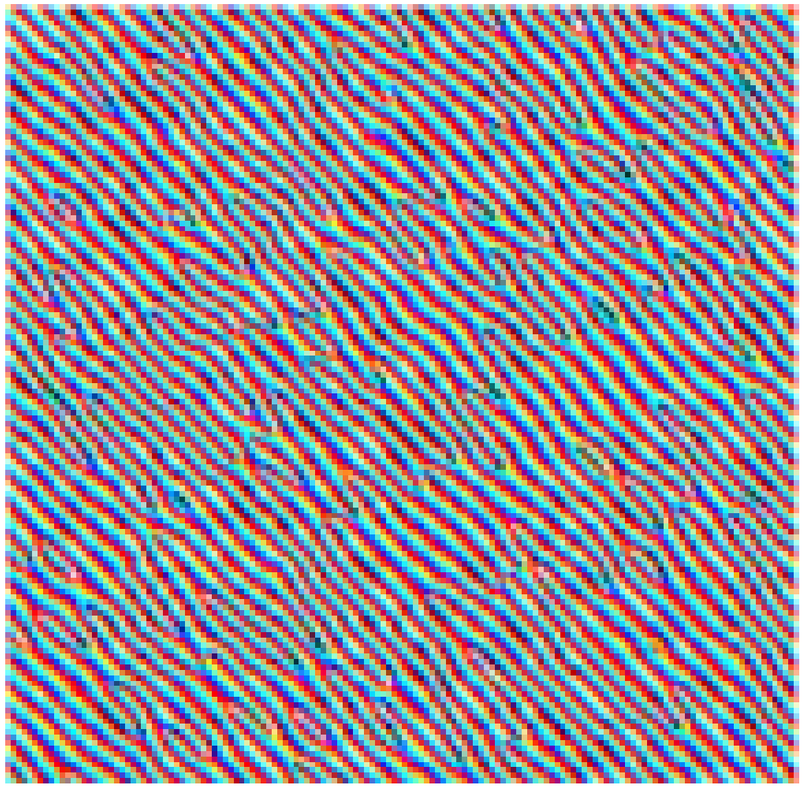
```
model = Sequential()
model.add(Conv2D(10, kernel_size=2, activation='relu',
                 input_shape=(img_rows, img_cols, 1),
                 padding='equal'))
# Second convolutional layer
model.add(Conv2D(10, kernel_size=2, activation='relu')
model.add(Flatten())
model.add(Dense(3, activation='softmax'))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

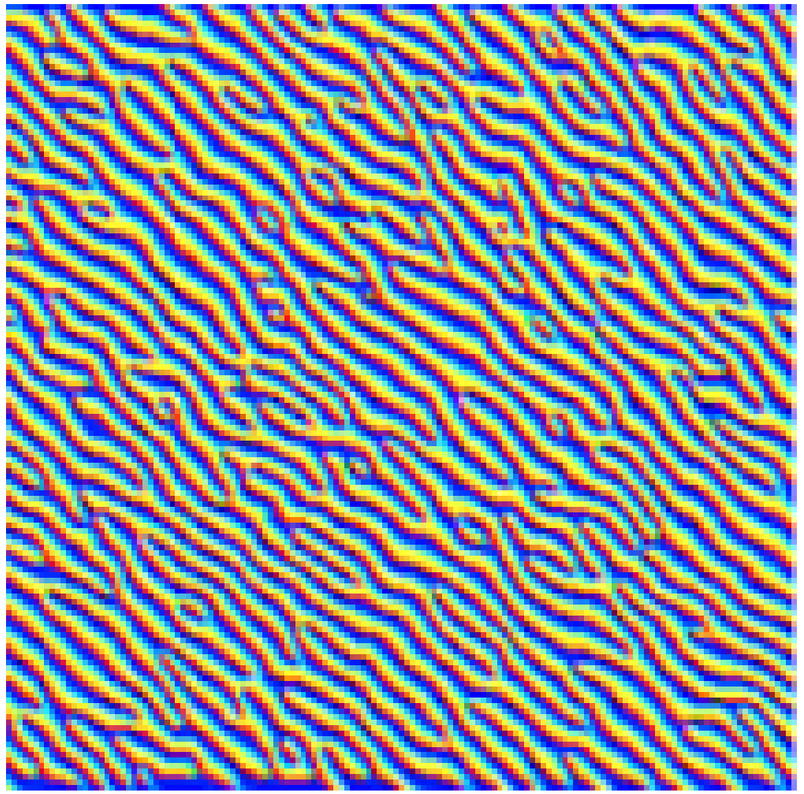
아까 앞에서 봤던 것과 같은 컨볼루션 layer인데 하나를 추가합니다. 이때 첫번째 layer로 부터 입력을 받기때문에 input shape을 인수ㅜ로 제공할 필요가 없습니다.

Why do we want deep networks?



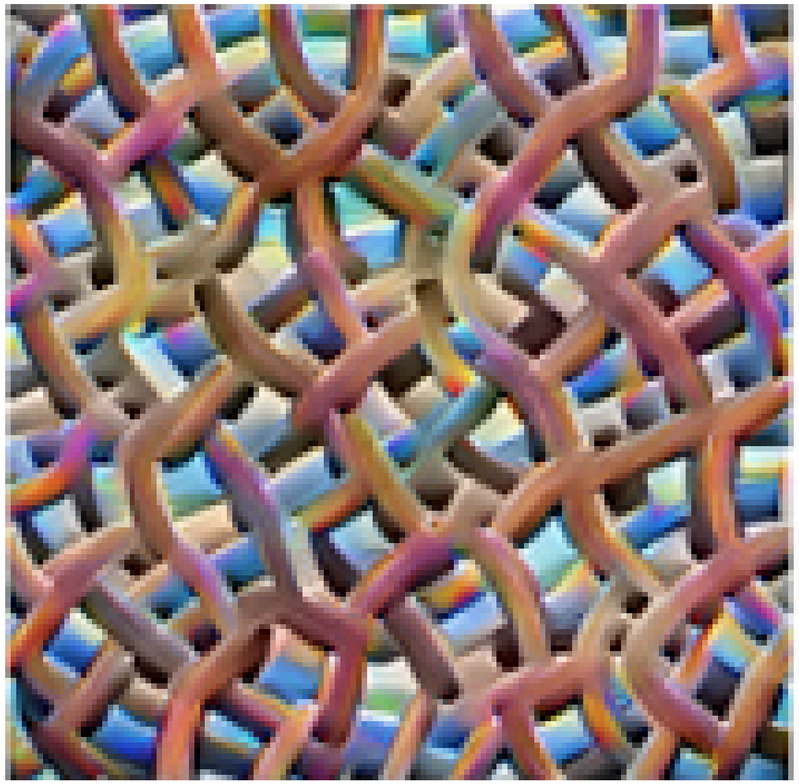
Features in early layers

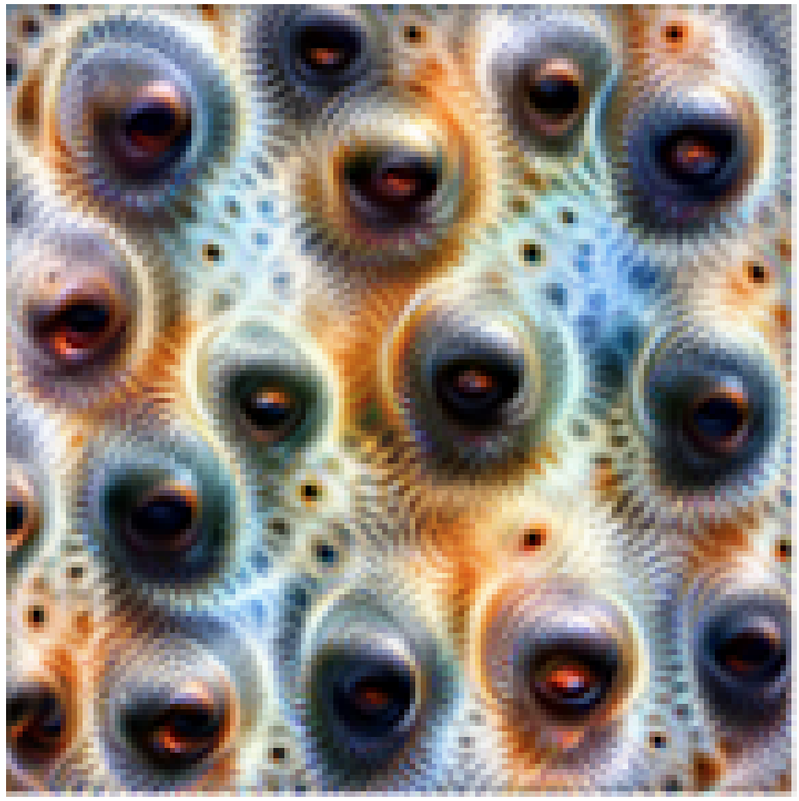




초기 layer은 방향 선 또는 단순한 텍스처 입니다.

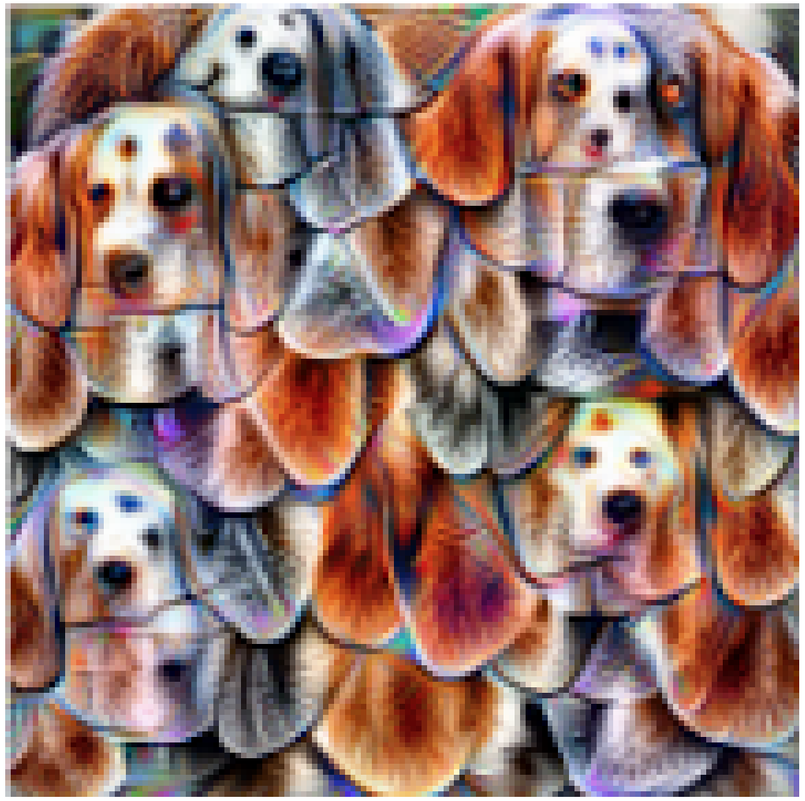
Features in intermediate layers

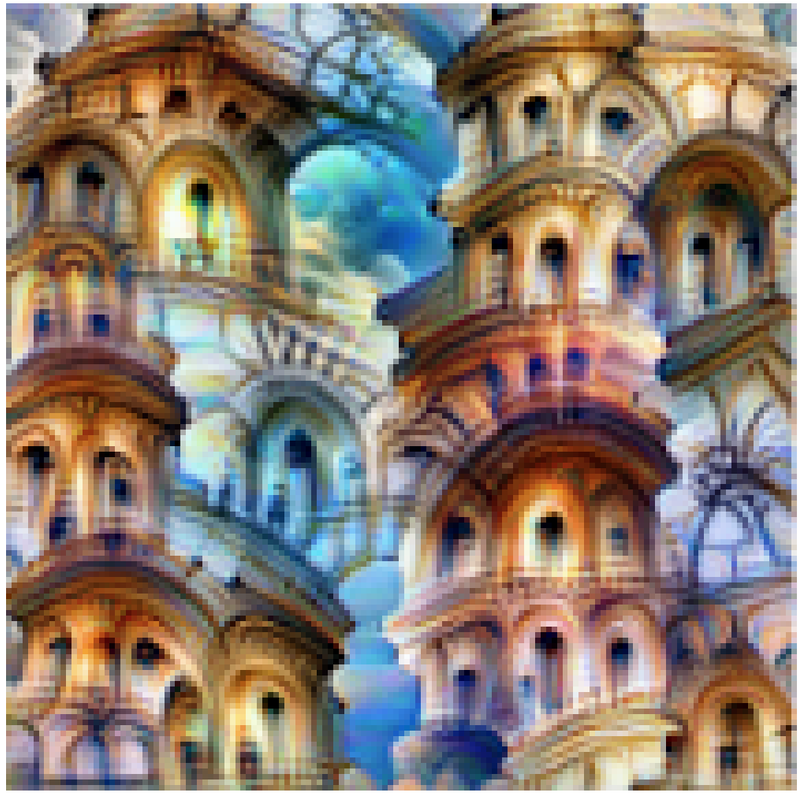




중간 layer의 특징은 눈과 같은 단순한 물체를 포함하는 더 복잡한 feature에 반응하는 경향이 있습니다.

Features in late layers





후기 layer에서의 특징은 네트워크에 여러 층의 컨볼루션을 가지면서 단순한 feautre에서 복잡한 feature까지 구축할 수 있습니다.

How deep?

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

깊이 네트워크를 쌓으려면 많은 데이터가 필요합니다.

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'))
```

28x28 픽셀의 이미지를 처리하는 이 고밀도 2계층 네트워크를 생각해 보십시오. 첫 번째 층은 10개의 단위를 가지고 있다. 그들 각각은 가중치를 통해 이미지 내의 픽셀들 각각에 연결된다. 제2 층은 10개의 유닛을 가지며, 이들 각각은 제1 층의 모든 유닛에 연결된다. 마지막으로, 마지막 레이어의 각 유닛은 레이어 2의 각 유닛에 연결됩니다.

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

summary()를 사용해 model에 대한 설명을 볼 수 있고 총 parameters는 7993개입니다.

```
_________________________________________________________________
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,)))
```

$$parameters = 784 * 10 + 10$$

$$= 7850$$

첫 번째 계층에는 7850개의 매개 변수가 있습니다. 이는 이미지의 각 픽셀에 대해 1개입니다. 784픽셀은 이 레이어의 단위 수에 10단위 + 이 모든 단위의 치우침 항에 대한 10개의 매개 변수입니다.

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

$$parameters = 10 * 10 + 10$$

$$= 110$$

두 번째 레이어에는 레이어 1의 모든 유닛에 대해 하나의 파라미터가 있으며, 이 레이어의 유닛 수에 바이어스 항에 대한 10개의 파라미터를 곱한다.

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

$$parameters = 10 * 3 + 3$$

$$= 33$$

마지막 레이어는 레이어 2의 각 유닛에 대해 하나의 매개 변수를 가지며, 이 레이어 + 3 바이어스 항에 3개의 단위를 곱한다. 그래서 모수의 총 수가 7,993개입니다.

$$7850 + 110 + 33 = 7993$$

```
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'))
```

여기 우리의 조밀하게 연결된 네트워크와 동일한 수의 계층과 단위를 가진 CNN을 구현하는 코드가 있다.

```
model.summary()
```

summary를 통해 매개변수가 24533개인것을 확인 할 수 있습니다.

```
_________________________________________________________________
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'))
```

$$parameters = 9 * 10 + 10$$

$$= 100$$

계산을 해보자: 첫 번째 레이어에는 각각 9개의 매개변수를 가진 10개의 커널이 있고 10개의 바이어스 항이 있다. 100달러입니다.

.

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

$$= 910$$

제2 레이어에서, 각 유닛은 컨볼루션 커널을 통해 제1 레이어의 각 피처 맵에 연결된다. 그것은 10 곱하기 9 곱하기 10 파라미터, 즉 900이고, 각 단위에 대한 편향 항, 총 910입니다. 평면 도면층에는 매개변수가 전혀 없습니다. 레이어 2의 기능 맵에서 출력을 가져와 하나의 큰 어레이로 평평하게 만듭니다. 여기에는 패딩이 없기 때문에 컨볼루션은 각 후속 레이어에 동일한 수의 픽셀을 남기므로 각 피쳐 맵에서 28x28 픽셀로 끝나며, 피쳐 맵은 10개 또는 총 7840 픽셀이 된다.

$$parameters = 0$$

$$parameters = 7840 * 3 + 3$$

$$= 23523$$

마지막 레이어의 3 곱하기 단위는 23520 + 3 바이어스 항은 23523입니다. 합치면 24,533명이 됩니다. 따라서 컨볼루션 레이어가 매개 변수의 수를 반드시 줄이는 것은 아닙니다. 다른 예를 봅시다.

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

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()
```

첫번째 layer에 집중된 후 다른 layer에는 훨씬 적은 매개 변수를 가집니ㅏㄷ.

```
_________________________________________________________________
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'))
```

이것은 각 레이어에 동일한 수의 유닛을 가진 컨볼루션 네트워크이다. 촘촘하게 연결된 신경망과는 대조적으로,

```
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
_________________________________________________________________
```

이 네트워크에서 요약 방법을 실행하면 처음 몇 개의 계층은 매우 적은 매개 변수를 사용하는 반면 마지막 계층은 많은 매개 변수를 사용하는 것을 볼 수 있습니다. 이것에 대해 생각하는 한 가지 방법은 컨볼루션이 더 많은 표현력을 가지고 있기 때문에 더 적은 매개 변수를 필요로 하지만, 이러한 더 많은 표현 표현을 읽어내는 것은 출력 측면에서 더 많은 매개 변수를 필요로 한다는 것이다.

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

datacamp

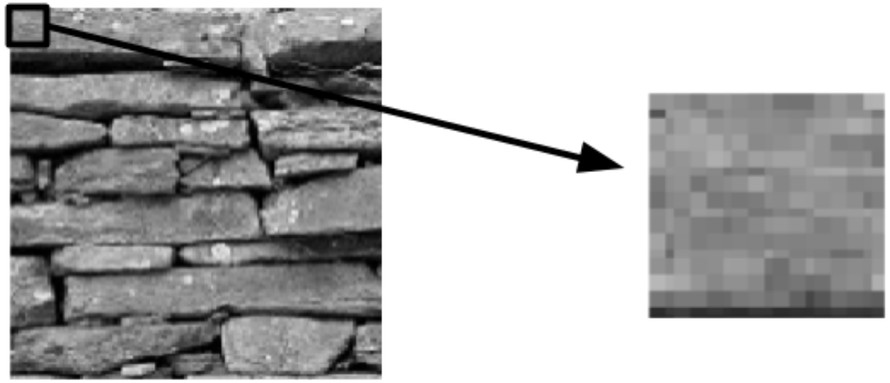
```
model.summary()
```

```
_________________________________________________________________
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
_________________________________________________________________
```

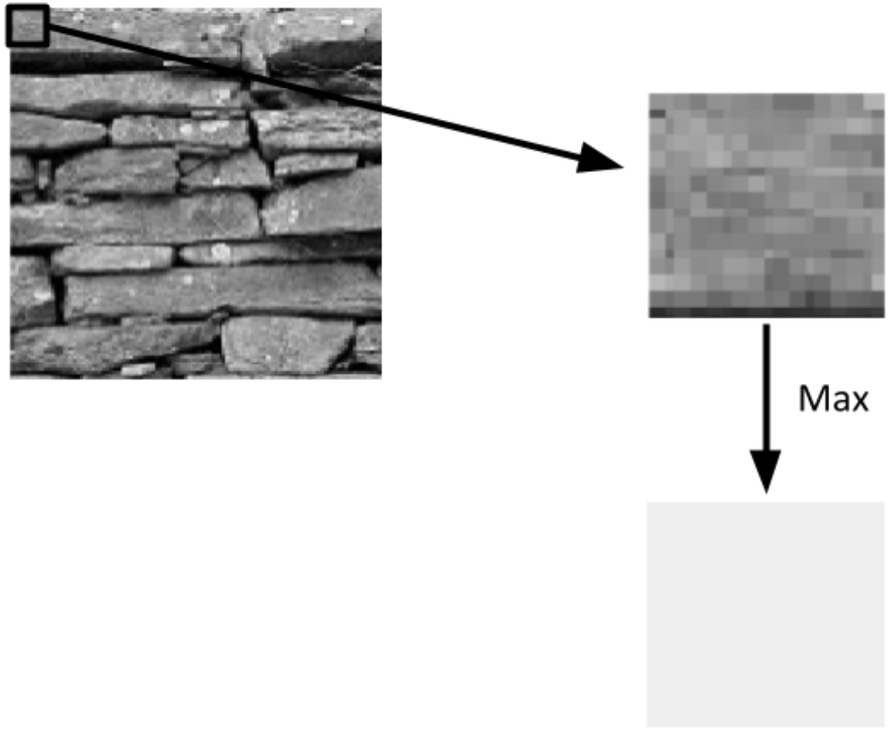
신경망을 fitting할 때의 과제 중 하나는 많은 수의 매개변수인데
이를 완화하는 방법 중 하나가 컨볼루션 layer의 출력을 간결한 방식으로 요약하는 것이라고 하는데 이를 위해 풀링 작업을 사용할 수 있다고 합니다.

예들을어 최대풀링은 최대값을 기준으로 픽셀 그룹을 요약하는 것을 말합니다.

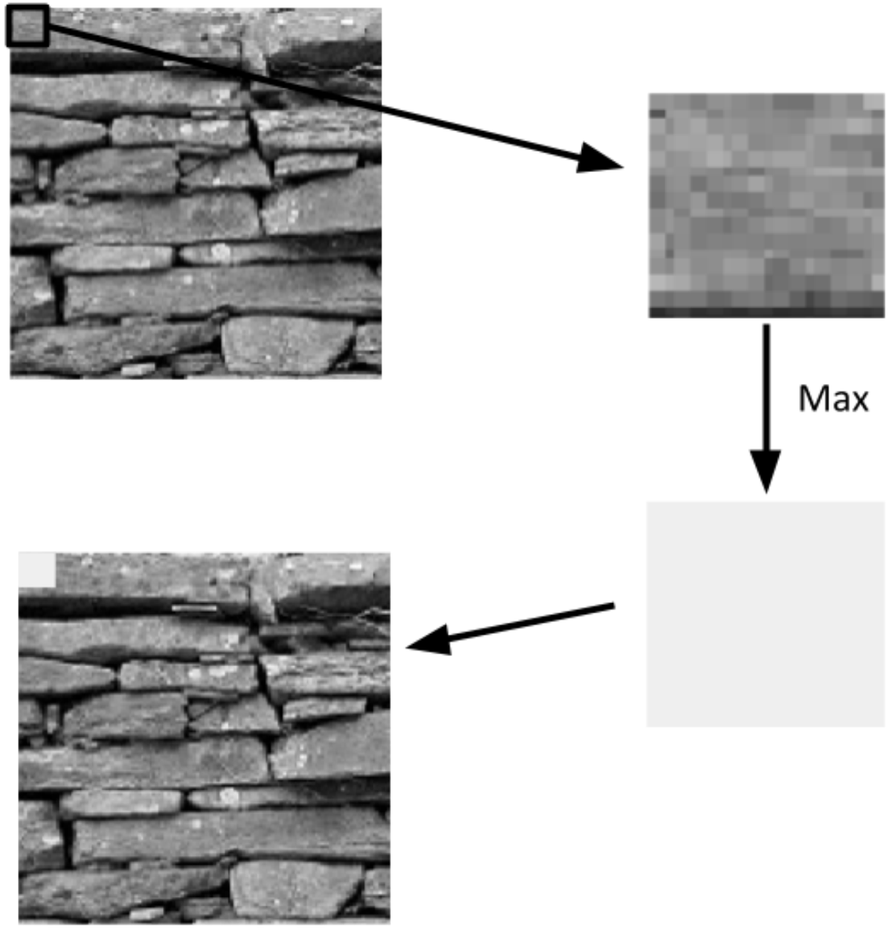
왼쪽부터



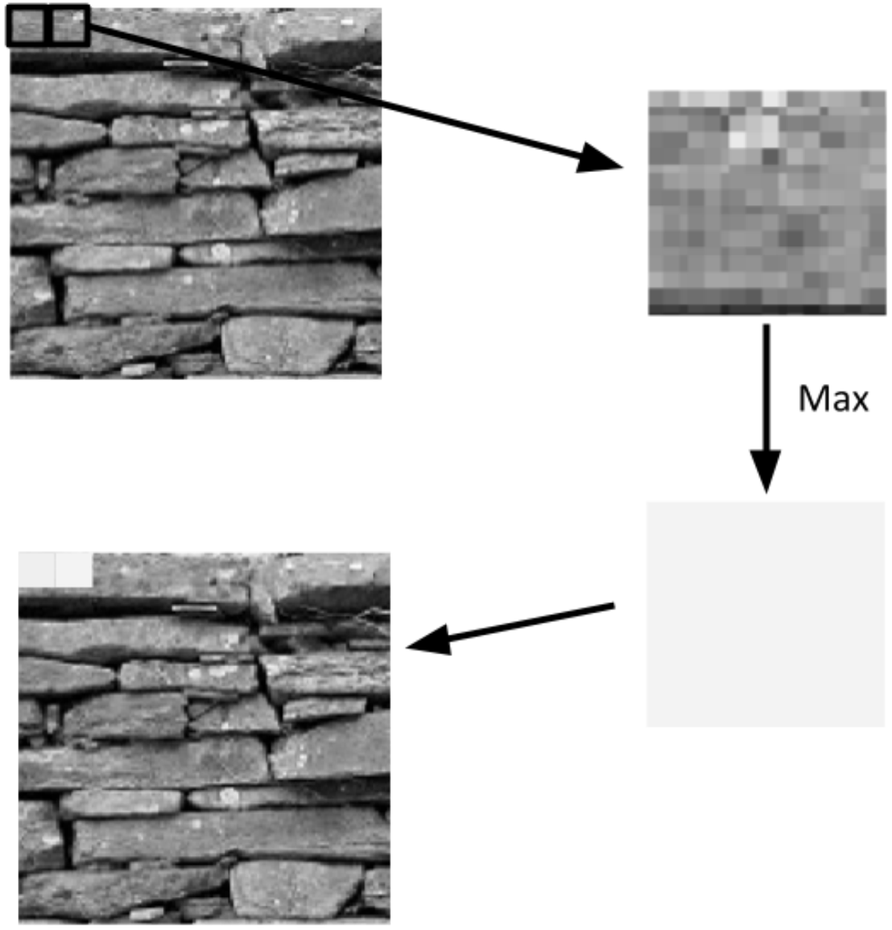
부분 픽셀을 추출하고



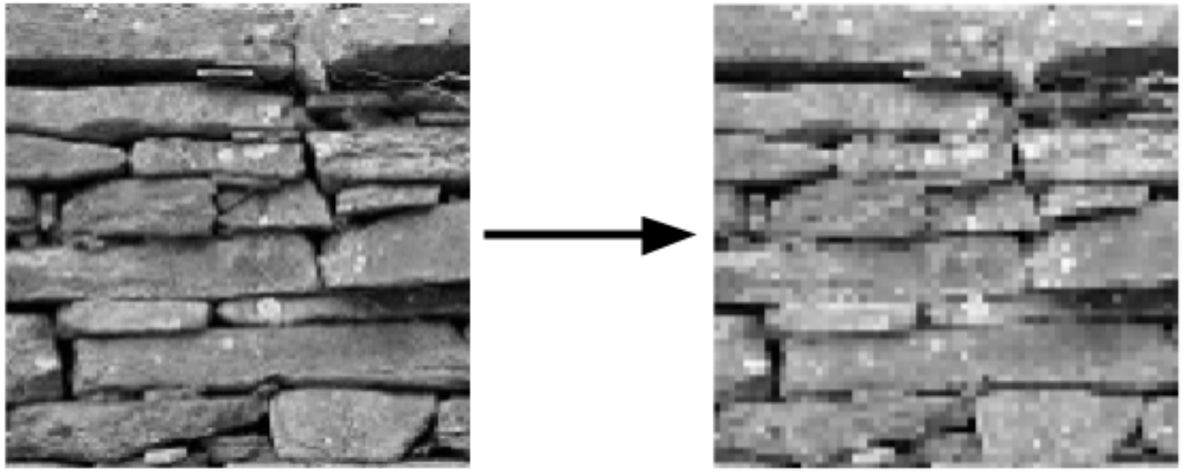
거기에 있는 픽셀의 최대값을 계산합니다.



출력에서 이 픽셀을 이 최대값을 저장하는 하나의 큰 픽셀로 바꿉니다.



우리는 창을 이동하고 다음 창에서 픽셀의 최대값을 계산하여 이미지의 해당 부분을 이미지의 해당 부분의 최대값으로 하나의 큰 픽셀로 바꿉니다.



2 x 2 크기의 여러 창에서 이 작업을 반복하면 원본 픽셀 수의 4분의 1인 이미지가 생성되고 이미지의 각 부분에서 가장 밝은 부분만 유지됩니다.

Implementing max pooling

풀링 구현방식은 다음과 같은데

```
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])
```

…

출력을 할당하는 것으로 시작합니다. 이것은 입력만큼 각 차원에 절반의 픽셀을 갖습니다. 출력의 첫 번째 좌표에서 시작하여 입력의 각 차원에서 처음 두 좌표에서 이미지의 최대값을 계산합니다. 다음으로 창을 따라 첫 번째 차원을 따라 2픽셀만큼 슬라이드하여 이 창의 최대값을 계산합니다. 입력의 첫 번째 행이 완료될 때까지 계속 그렇게 합니다. 그런 다음 창을 입력의 두 번째 행 시작 부분으로 이동하여 이 위치에 대한 입력의 세 번째 및 네 번째 행 좌표의 최대값을 계산합니다. 우리는 계속해서 창을 밀어냅니다. 궁극적으로 출력의 각 위치에서 입력의 해당 위치에서 2 x 2픽셀의 창에 대한 최대값을 계산합니다.

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])
```

이 작업을 구현하는 또 다른 방법은 루프를 사용하는 것입니다. 각 반복에서 먼저 해당 행을 선택합니다. 출력의 현재 행에서 인덱스 ii에 2를 곱한 다음 2픽셀을 초과할 때까지입니다. 열 인덱스 jj의 내부 루프에 대해서도 동일합니다. 이것은 이전에 행별로 분류한 것과 동일한 작업을 수행합니다.

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'))
```

MaxPool2D 개체를 사용하여 최대 풀링 작업을 Keras 합성곱 신경망에 통합할 수 있습니다. 이 객체를 가져온 후 우리가 필요로 하는 다른 객체와 함께 이전과 마찬가지로 컨볼루션 신경망을 구축하기 시작합니다. 그러나: 각 컨볼루션 레이어 다음에 풀링 레이어를 추가합니다. MaxPool2D 개체에 대한 입력(이 경우 2)은 풀링 창의 크기입니다. 즉, 여기서 풀링은 출력의 각 위치에 대한 입력에서 2x2픽셀의 창에 대해 최대값을 취합니다. 우리는 두 번째 convolutional layer를 추가하고 또 다른 maxpooling layer를 추가합니다. 네트워크의 요약 부분은 이전과 동일합니다. 평탄화 계층 다음에 softmax 활성화가 있는 조밀 계층이 뒤따릅니다. 이것이 매개변수 수에 어떤 영향을 줍니까?

```
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
=================================================================
```

이 모델에 대한 요약 방법을 실행하면 풀링 작업을 사용하면 모델의 매개변수 수가 크게 감소함을 알 수 있습니다. 풀링 작업이 없는 이 모델의 30,000개 이상의 매개변수 대신 이제 2,000개 미만의 매개변수가 있습니다.

Let's practice!

IMAGE PROCESSING WITH KERAS IN PYTHON