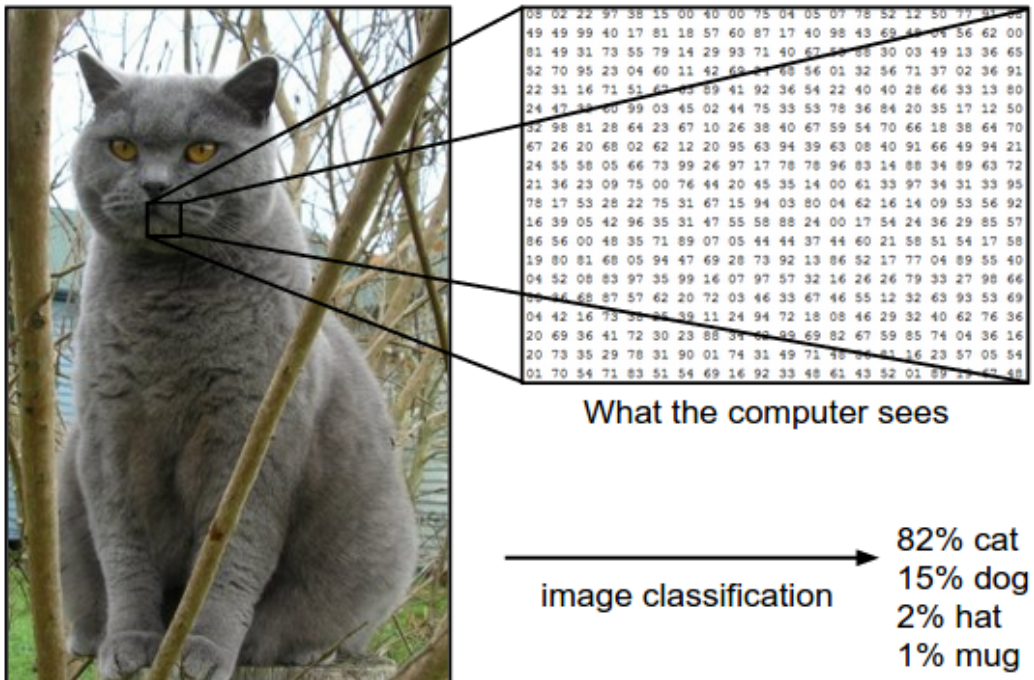
Lecture 4. Introduction to Convolutional Neural Networks

4.1 Overview

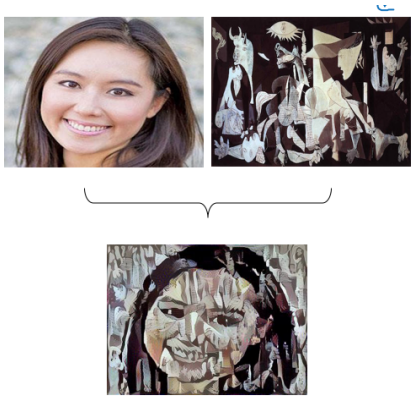
4.1.1 Computer vision problem

- Image classification
  - 64*64 이미지를 입력하였을 때 고양이인지 여부 판단



- Object detection
  - 이미지 안의 물체를 탐색하여 자동차에 네모 박스

- Neural style transfer
  - 입력 이미지와 스타일 이미지를 합쳐 새로운 이미지 생성



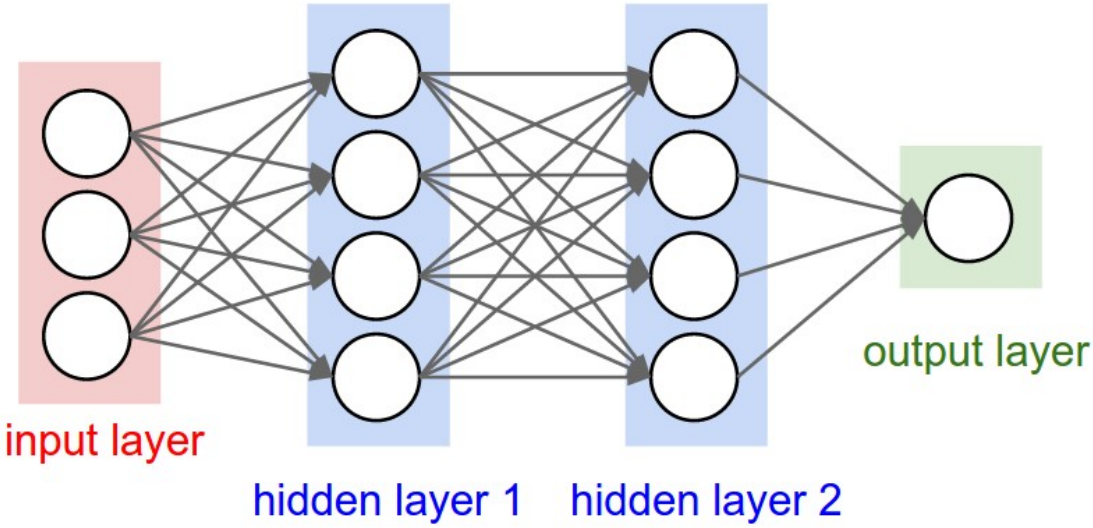
Deep learning with image

- 이미지를 input으로 사용하면 dimension이 매우 큼
  - 64*64 color image: 12288 input units
  - 1000*1000 color image: 3,000,000 input units
- Hidden unit의 수에 따라 weight의 수가 지나치게 커지기 때문에 메모리 부족

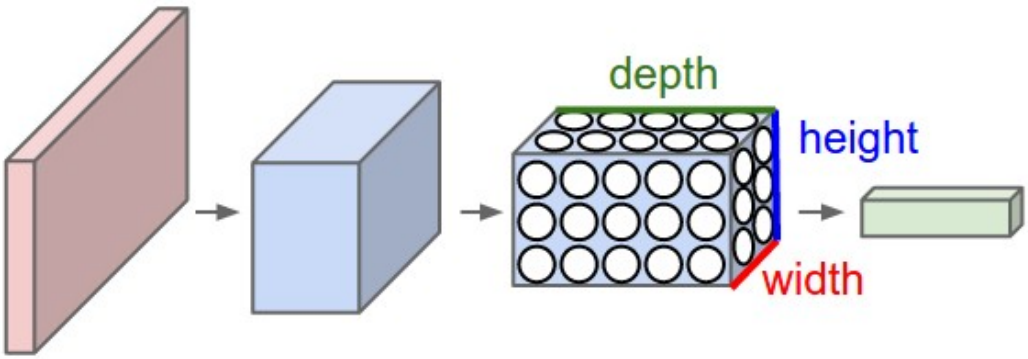
- Fully connected layer(dense layer)만을 사용한다면 이미지의 공간적 구조 학습이 어려움

4.1.2 Overview of convolution architecture

- Architecture of DNN



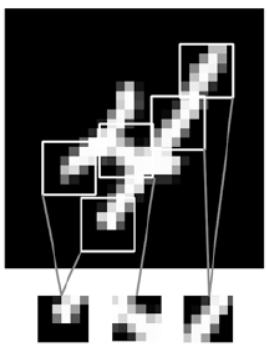
- Architecture of CNN
  - 각 layer는 3D input을 3D output으로 변환함
  - 빨간색 input layer는 input image(Red, Green, Blue channels)
  - 각 layer는 convolution operation을 통해 다음 layer로 변환됨



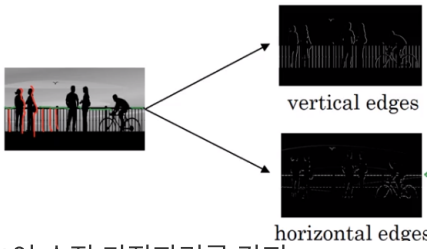
4.2 Convolution operation

4.2.1 Convolution operation 이란?

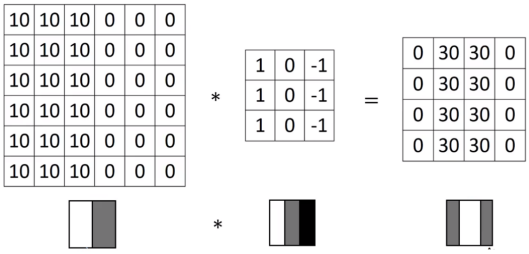
- 이미지를 작은 2D window를 통해 입력하여 local pattern을 학습하는 과정



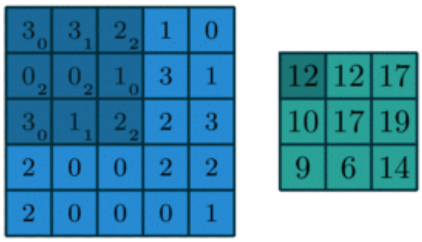
- Convolution operation의 예
  - Blue grid: 5*5 input feature map
  - 움직이는 matrix $\begin{pmatrix} 0 & 1 & 2 \\ 2 & 2 & 0 \\ 0 & 1 & 2 \end{pmatrix}$: 3*3 kernel
  - Green grid: 3*3 output feature map
- Convolution operation을 통한 edge detection



- 특정 kernel을 사용한 output이 수직 가장자리를 감지

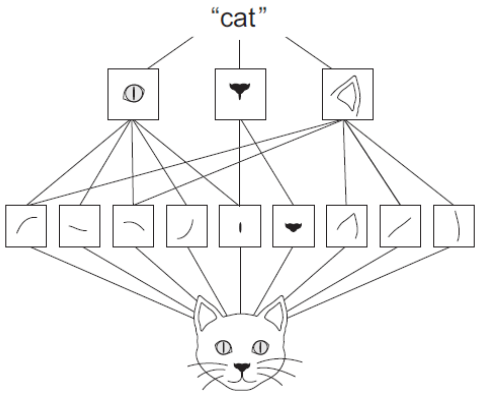


- CNN에서는 주어진 kernel의 값을 적용하지 않고 kernel의 값을 weight로 모형을 통해 학습



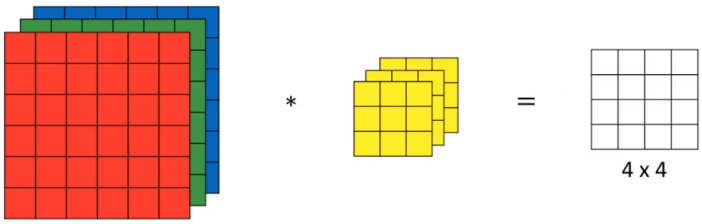
4.2.2 Convolution operation의 특징

- Translation invariant
  - 오른쪽 아래 모서리에서 학습된 패턴을 다른 곳에서도 인식할 수 있음
  - 일반화 능력을 가진 표현을 학습
- Learning spatial hierarchies of patterns
  - 첫 번째 layer는 edge와 같은 작은 지역 패턴을 학습
  - 두 번째 layer는 첫 번째 층의 특성으로 구성된 더 큰 패턴 학습 (추상적인 시각적 개념을 학습)

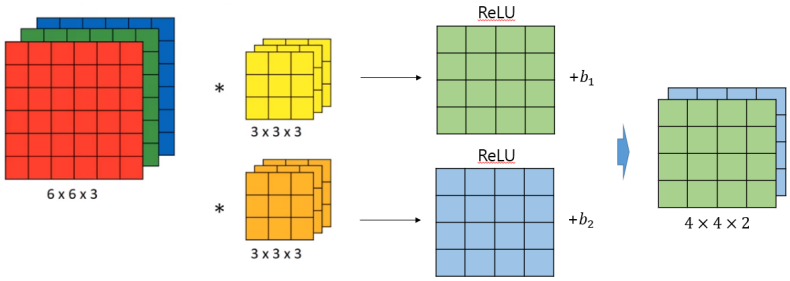


4.2.3 Convolutional operation의 작동방식

- 두 개의 parameter로 정의
  - Filter의 크기: 일반적으로 3*3*, *5*5 크기의 filter를 주로 사용
  - Filter의 수: Feature map output의 깊이
- 컬러 이미지는 RGB의 각 이미지로 구성되어 3개의 feature map으로 구성
- 예)
  - Input image는 $6 \times 6 \times 3$ 형태의 volume
  - Filter: $3 \times 3 \times 3$ 크기 1개
  - Output: 4×4 feature map 1개 생성

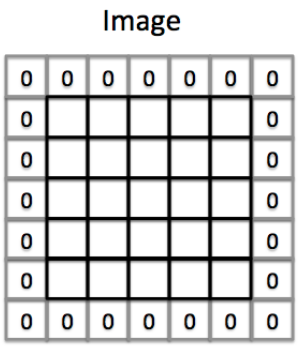


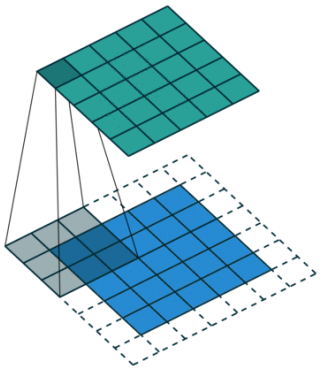
- 예)
  - Input image는 $6 \times 6 \times 3$ 형태의 volume
  - Filter: $3 \times 3 \times 3$ 크기 2개
  - Output: 4×4 feature map 2개 생성



4.2.4 Padding

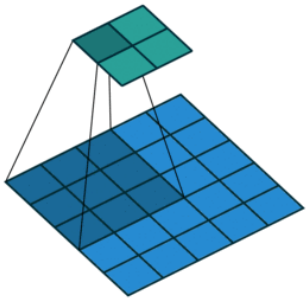
- 이미지 가장자리의 픽셀은 convolution 계산에 상대적으로 적게 반영
- 이미지 가장자리를 0으로 둘러싸서 가장자리 픽셀에 대한 반영 횟수를 늘림
- "Valid" padding
  - Padding을 적용하지 않음
- "Same" padding
  - Input과 output의 이미지 크기가 동일하게 되도록 padding 수를 결정





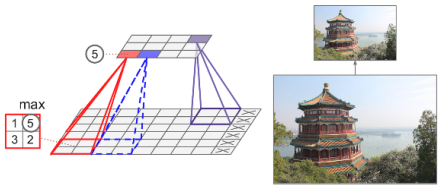
4.2.5 Strides

- Kernel의 이동을 한 번에 몇 칸씩 이동하는가
- Stride=2: 한 번에 두 칸씩 이동 (feature map의 너비와 높이가 2배수로 다운샘플링 되었음을 의미)



4.2.6 Max-pooling layer

- Filter를 통하지 않고 해당 영역의 input 중 가장 큰 값을 출력
- 일반적으로 2*2 크기의 window, stride=2 사용
- 강제적인 subsampling 효과
  - weight 수를 줄여 계산속도를 높임
  - 특징을 견고하게 감지
- 학습할 weight가 없음: 일반적으로 convolutional layer+pooling layer를 하나의 레이어로 취급



Max pooling layer(2 × 2 pooling kernel, stride 2, no padding)

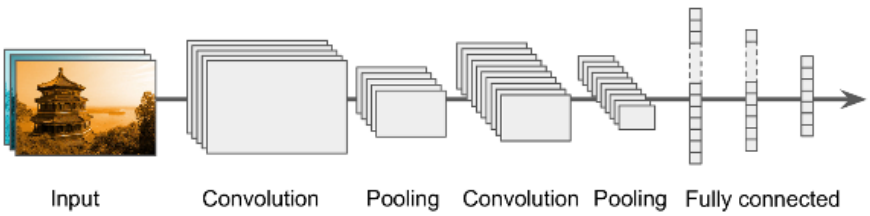
참고

- W: input volume size
- F: filter size
- S: strides
- P: the number of zero-paddings

$$\text{output neuron의 수:} \frac{W - F + 2P}{S} + 1$$

- Ex) 7x7 input, 3x3 filter with stride 1 and pad 0 : 5x5 output
- Ex) 10x10 input, 3x3 filter with stride 2 and pad 0?

4.3 Example of CNN architecture



 - 일반적으로 convolutional layer (+ ReLU or other activations) + pooling layer를 여러 개 쌓음 - Network가 진행될 수록 feature map의 크기는 줄어들고 깊이는 증가 - 마지막에 Fully connected layer(+ ReLU or other activations) 추가 - Output 형태에 맞는 output layer

4.4 CNN structure with Keras

- Input shape: `(image_height, image_width, image_channels)`
- `keras.layers.Conv2D(filters, kernel_size, strides=(1, 1), padding='valid', activation=None)` https://keras.io/layers/convolutional/ (https://keras.io/layers/convolutional/)
  - `filters` : the dimensionality of the output space (i.e. the number of output filters)
  - `kernel_size` : height and width of the 2D convolution window
  - `strides` : the strides of the convolution along the height and width
  - `padding` : "valid" or "same"
  - `activation` : activation function https://keras.io/activations/ (https://keras.io/activations/)
- `keras.layers.MaxPooling2D(pool_size=(2, 2), strides=None, padding='valid')`
  - `pool_size` : Pooling window size

    - strides : default = pool_size
- keras.layers.Flatten()
    - Flattens the input
    - Fully connected layer를 적용하기 위함

In [1]:
```
from keras import layers
from keras import models

model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(32, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
model.summary()
```

```
Using TensorFlow backend.
/usr/lib/python3.6/importlib/_bootstrap.py:219: RuntimeWarning: compiletime version
3.5 of module 'tensorflow.python.framework.fast_tensor_util' does not match runtime
version 3.6
  return f(*args, **kwds)


_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 32)        320
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1600)              0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                102464
_________________________________________________________________
dense_2 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_3 (Dense)              (None, 10)                330
=================================================================
Total params: 123,690
Trainable params: 123,690
Non-trainable params: 0
_________________________________________________________________
```

- conv2d_1: 3*3 크기의 filter 32개 (288개 weight) + bias 32개 = 320 parameters
- max_pooling2d_1: 학습할 parameter 없음

TO DO: 나머지 layer의 parameter의 수를 계산하고 확인하시오.

References

- https://www.coursera.org/specializations/deep-learning (https://www.coursera.org/specializations/deep-learning)
- Hands on Machine Learning with Scikit-Learn and Tensorflow, Aurélien Géron (http://www.hanbit.co.kr/store/books/look.php?p_code=B9267655530)
- Deep Learning with Python, François Chollet, (https://www.manning.com/books/deep-learning-with-python)
- http://cs231n.github.io/ (http://cs231n.github.io/)

In [ ]: