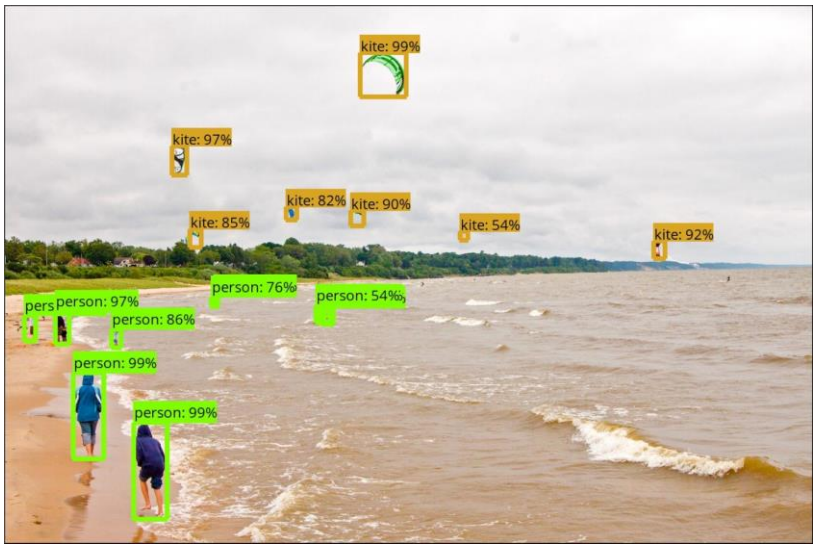
CH. 11. CNN 기초

Section 01 딥러닝 이미지 인식

- 이미지 인식이란 이미지나 영상에 비치는 문자와 얼굴 등의 물체나 특장을 감지하는 기술이다.
- 구체적으로는 이미지 분류, 물체 위치 감지, 이미지 내용 확인 등 다양한 인식 기술이 있다.



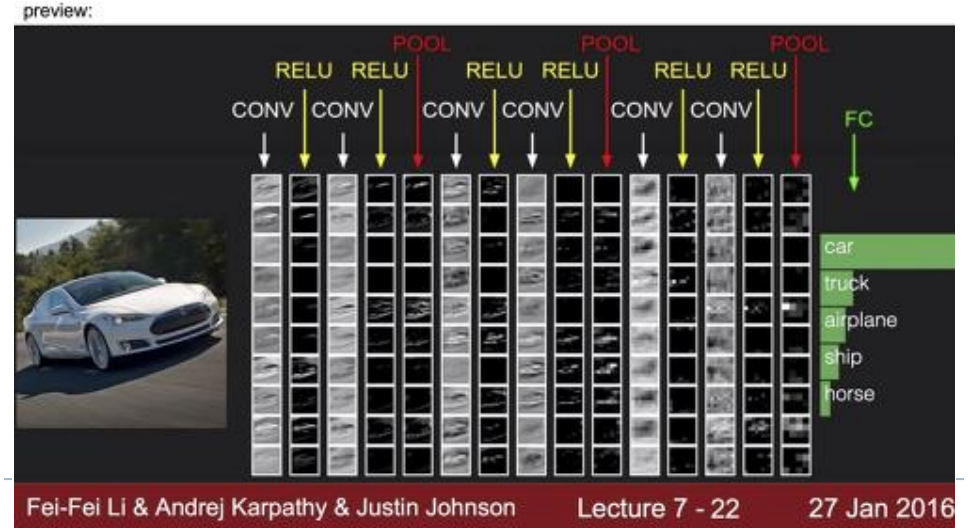
이미지 분류와 물체 위치 감지

이미지 영역 분할

Section 02 CNN

❑ CNN의 개요

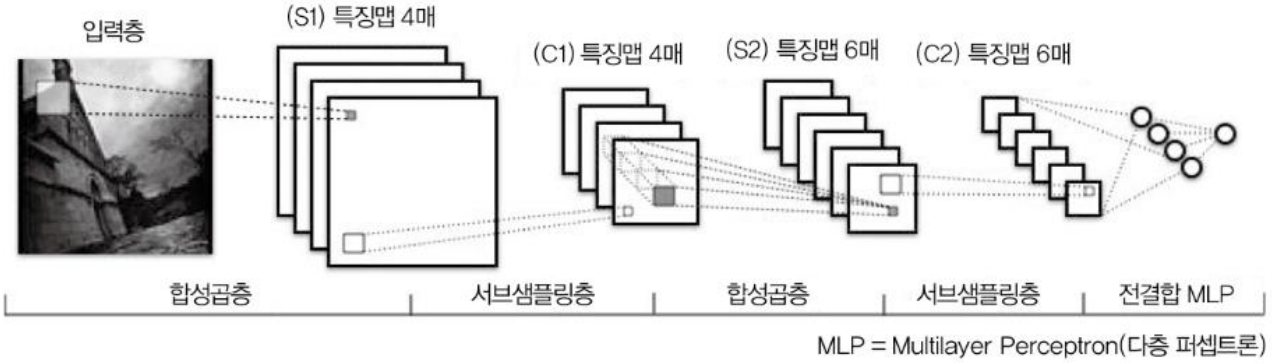
- CNN(합성곱 신경망)은 인간 뇌의 시각 피질과 유사한 구조를 가진 합성곱 층을 사용하여 특징을 추출하는 신경망이다.
- 앞 장에서 학습한 전결합층만 있는 신경망에 비해 이미지 인식 등의 분야에서 진가를 발휘한다.
- CNN은 대부분의 경우 합성곱층과 함께 풀링층이 사용된다.
- 합성곱층에서 얻은 정보를 축약하여 풀링층에서 최종적으로 이미지의 분류등을 실시하게 된다.



Section 02 CNN

❑ CNN의 개요

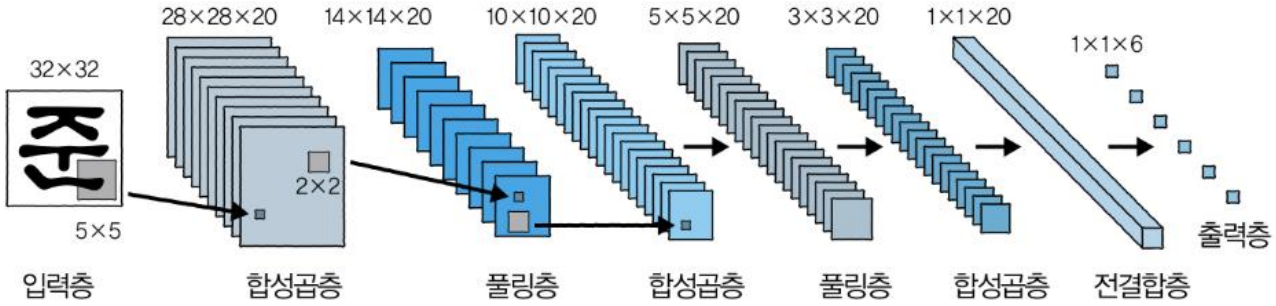
- 합성곱층은 전결합층처럼 특징을 추출하는 계층이지만 전결합층과 달리 2차원 그대로의 이미지 데이터를 처리하여 특징을 추출할 수 있기 때문에 선이나 모서리와 같은 2차원적인 특장을 추출하는데 뛰어나다.
- 다음 절에서는 각 층에 대해 배우고 그림과 같은CNN 모델을 구축하여 실제로 이미지를 분류해볼 것이다.



Section 02 CNN

❑ 합성곱층

- 합성곱층은 그림과 같이 입력 데이터의 일부분에 주목하여 그 부분에 대한 이미지의 특징을 조사하는 층이다.

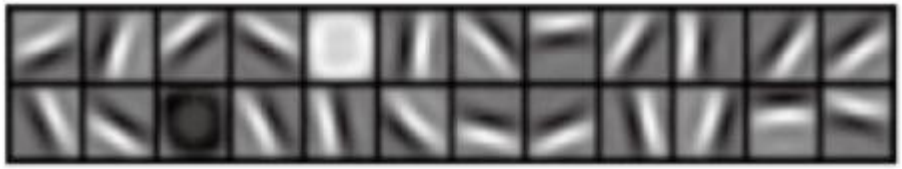


- 어떤 특징에 주목하면 좋을지에 대해서는 학습용 데이터와 손실 함수 등을 적절하게 결정하여 자동으로 학습한다.
- 예를 들어 얼굴 인식 CNN의 경우 적절한 학습이 진행되면 입력층에 가까운 합성곱 층에서는 선과 점 등 저차원적인 개념의 특징에 주목하고, 출력층에 가까운 합성곱 층에서는 눈이나 코등 고차원적인 개념의 특징에 주목하게 된다.

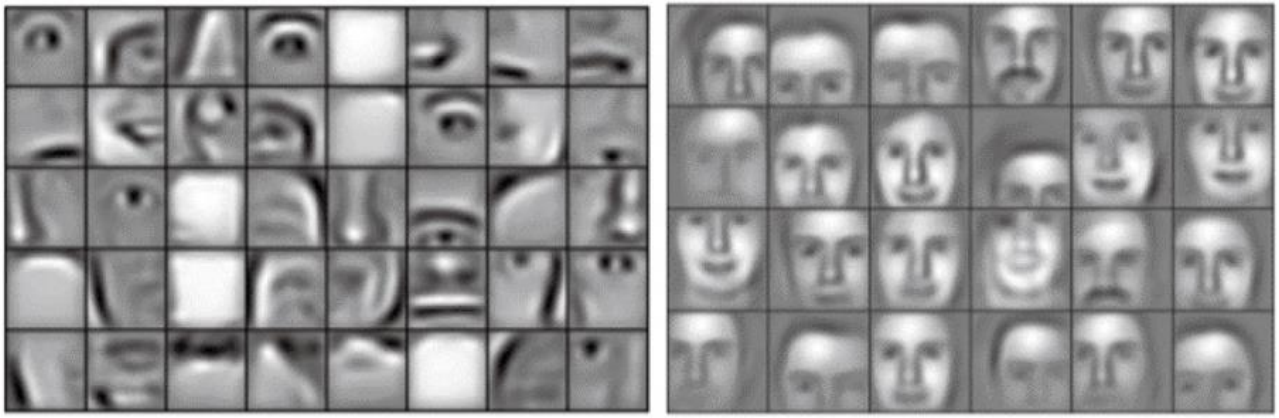
Section 02 CNN

❑ 합성곱층

- **주목할 만한 특징은 프로그램 내부에서는 필터(커널, kernel)로 불리는 가중치 행렬로 처리되며, 각 특징마다 하나의 필터를 사용한다.**



입력층과 가장 가까운 합성곱 층의 학습이 완료된 필터의 예

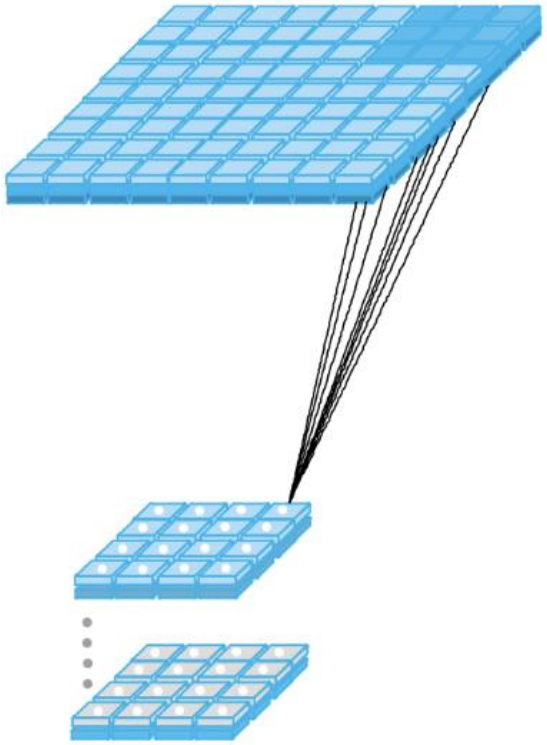


출력층에 가까운 합성곱 층의 학습이 완료된 필터의 예

Section 02 CNN

❑ 합성곱층

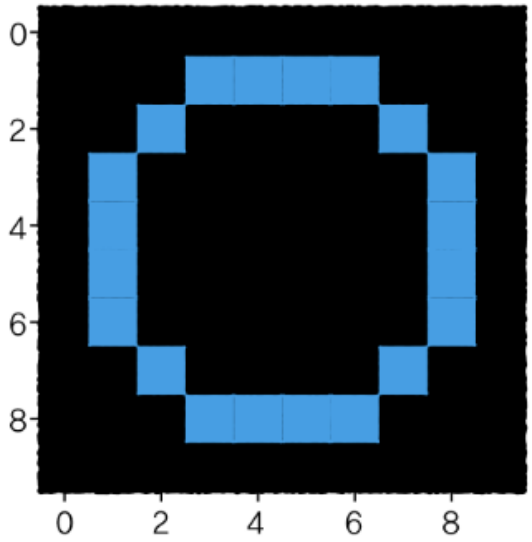
- 그림은 9 x 9 x 3(가로x세로x채널 수)의 이미지에 대해 3 x 3 x 3(가로x세로x채널 수)의 필터로 합성곱하는 모습이다.
- 하나의 3 x 3 x 3의 필터를 사용하여 새로운 4 x 4 x 4의 특징 맵(feature map)을 만들고 있다.
- 또한 각각 다른 필터를 사용하여 총 N장의 4 x 4 x 1의 맵을 만든다.
- 전체적으로 이 합성곱층에서는 9 x 9 x 3 이미지가 4 x 4 x 4의 특징 맵으로 변환된다.
- 합성곱 층을 설명할 때 2차원 필터를 예로 사용하는 경우가 많지만 실제로는 그림처럼 3차원 필터가 사용되는 경우가 더 많다.

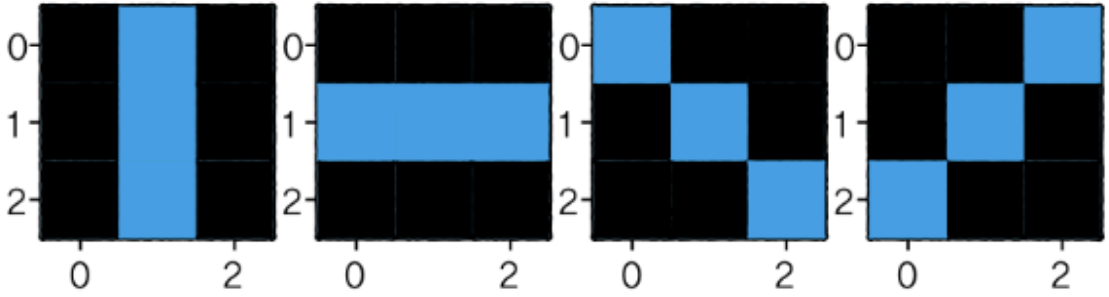


Section 02 CNN

❑ 합성곱층

- 예제 11-1은 합성곱층이나 풀링층에서 구체적으로 어떤 처리가 이루어지는지 살펴보기 위해 NumPy에서 구현된 코드이다.
- 여기서는 왼쪽 그림의 원 이미지 (10 x 10 크기의 흑백 이미지)에 대해 오른쪽 그림과 같은 필터를 사용하여 합성곱을 실시하고, 가로, 세로, 대각선을 검출한다.





Section 02 CNN

❑ 합성곱층

- **예제 11-1은 합성곱층이나 풀링층에서 구체적으로 어떤 처리가 이루어지는지 살펴보기 위해 NumPy에서 구현된 코드이다.**
- ex11_1.ipynb

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   import urllib.request
4
5   class Conv:'
6       def __init__(self, W):
7           self.W = W
8
9       def f_prop(self, X):
10          out = np.zeros((X.shape[0] - 2, X.shape[1] - 2))
11          for i in range(out.shape[0]):
12              for j in range(out.shape[1]):
13                  x = X[i:i+3, j:j+3]
14                  out[i, j] = np.dot(self.W.flatten(), x.flatten())
15          return out
16  local_filename, headers = urllib.request.urlretrieve('https://aidemystorageprd.blob.core.windows.net/data/5100_cnn_data/ci
    rcle.npy')
17  X = np.load(local_filename)
```

Section 02 CNN

❑ 합성곱층

- ex11_1.ipynb

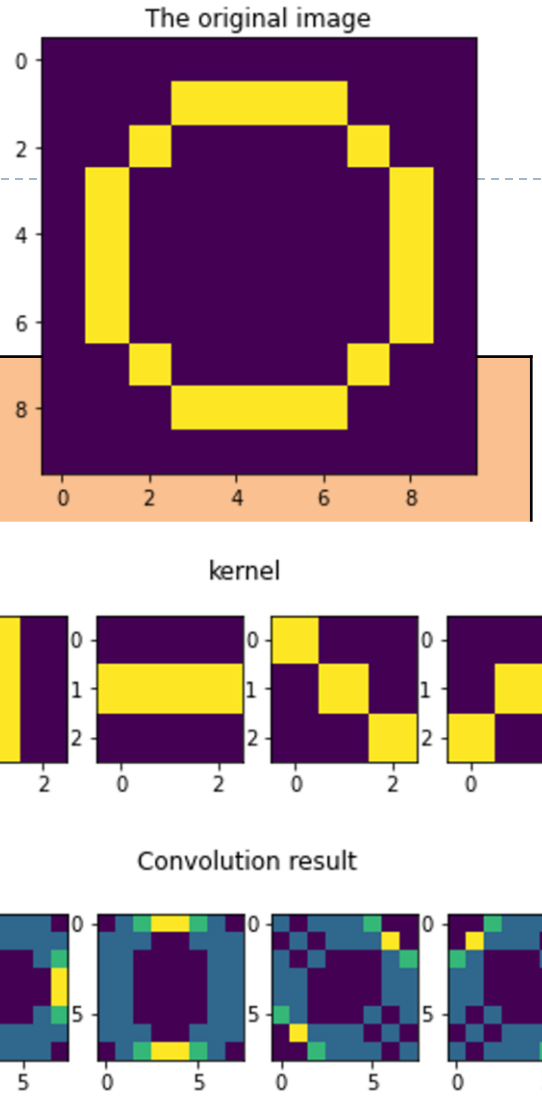
```
18
19  plt.imshow(X)
20  plt.title('The original image', fontsize=12)
21  plt.show()
22
23  W1 = np.array([[0, 1, 0],
                   [0, 1, 0],
                   [0, 1, 0]])
24  W2 = np.array([[0, 0, 0],
                   [1, 1, 1],
                   [0, 0, 0]])
25  W3 = np.array([[1, 0, 0],
                   [0, 1, 0],
                   [0, 0, 1]])
26  W4 = np.array([[0, 0, 1],
                   [0, 1, 0],
                   [1, 0, 0]])
27
```

Section 02 CNN

❑ 합성곱층

- ex11_1.ipynb

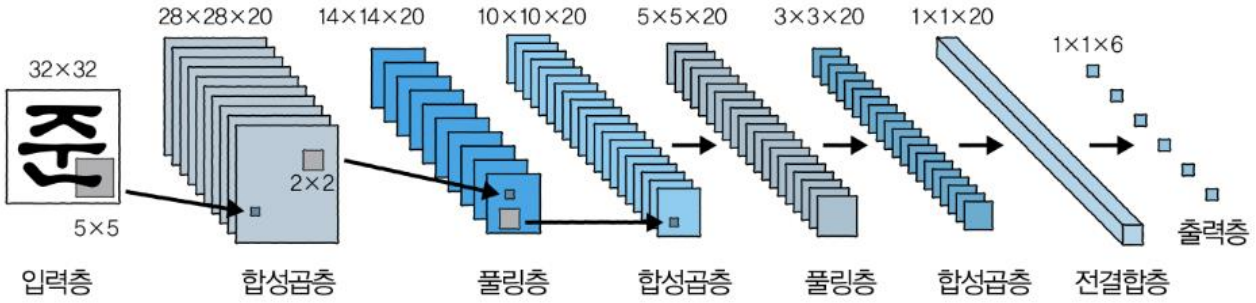
```
28  plt.subplot(1, 4, 1); plt.imshow(W1)
29  plt.subplot(1, 4, 2); plt.imshow(W2)
30  plt.subplot(1, 4, 3); plt.imshow(W3)
31  plt.subplot(1, 4, 4); plt.imshow(W4)
32  plt.suptitle("kernel", fontsize=12)
33  plt.show()
34
35  conv1 = Conv(W1); C1 = conv1.f_prop(X)
36  conv2 = Conv(W2); C2 = conv2.f_prop(X)
37  conv3 = Conv(W3); C3 = conv3.f_prop(X)
38  conv4 = Conv(W4); C4 = conv4.f_prop(X)
39
40  plt.subplot(1, 4, 1); plt.imshow(C1)
41  plt.subplot(1, 4, 2); plt.imshow(C2)
42  plt.subplot(1, 4, 3); plt.imshow(C3)
43  plt.subplot(1, 4, 4); plt.imshow(C4)
44  plt.suptitle("Convolution result", fontsize=12)
45  plt.show()
```



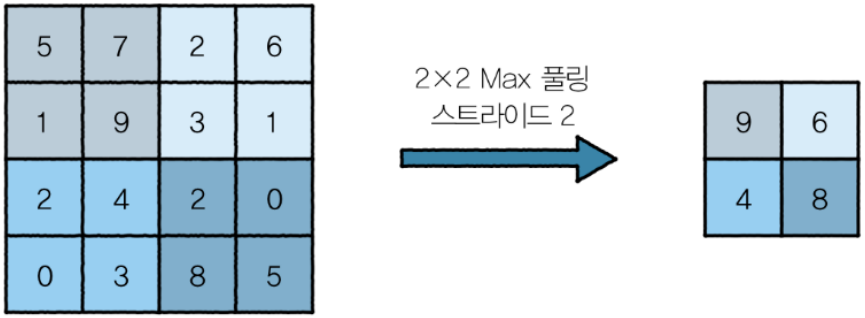
Section 02 CNN

❑ 풀링층

- 풀링층은 그림과 같이 합성곱층의 출력을 축약하고 데이터양을 줄이는 층이다.



- 다음 그림과 같이 특징 맵의 부분구간의 최댓값을 취하거나(Max 풀링) 또는 평균을 취하여(Average 풀링) 데이터의 압축을 실시한다.



Section 02 CNN

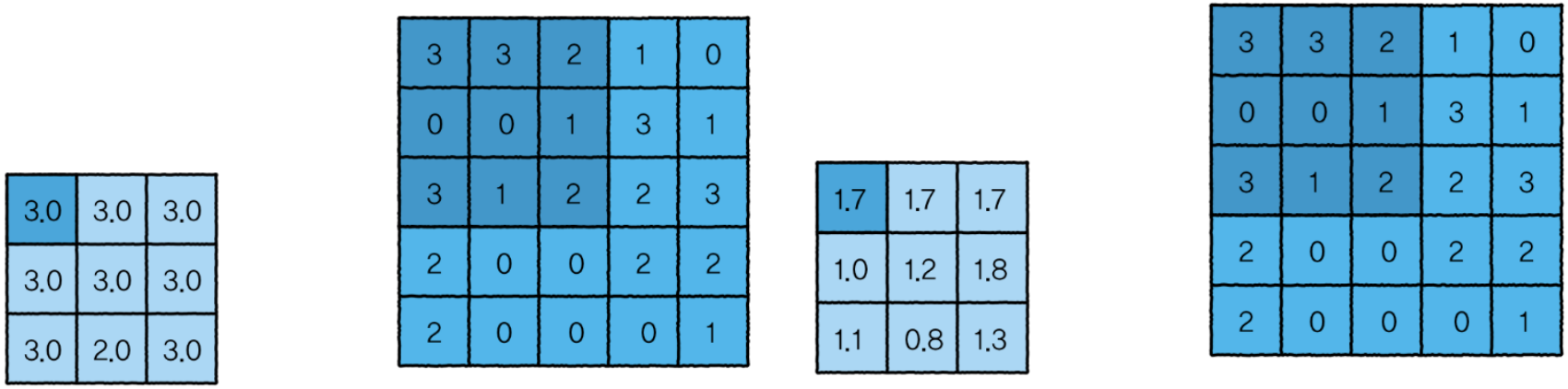
❑ 풀링층

- 합성곱은 이미지 내의 특징량의 분포를 확인할 수 있다.
- 하지만 같은 특징이 유사한 위치에 뭉쳐 분포하는 경우가 많고, 특징이 없는 장소가 넓게 분포하는 경우도 있어서 합성곱층에서 출력되는 특징 맵은 데이터 낭비가 많다.
- 풀링은 그러한 데이터 낭비를 줄이고, 정보 손실을 최소화하면서 데이터를 압축할 수 있다.
- 풀링에 의해 미세한 위치 정보는 삭제되어 버리지만 반대로 풀링층에 의해 추출된 특징이 원래 이미지의 평행이동 등에서 영향을 받지 않는 역할을 한다.
- 예를 들어 사진에 비친 필기체 숫자를 분류할 경우 숫자의 위치는 중요하지 않지만 풀링을 통해 그다지 중요하지 않은 정보를 삭제하여 입력 이미지의 위치 변화에 강한 모델을 구축할 수 있다.

Section 02 CNN

❑ 풀링층

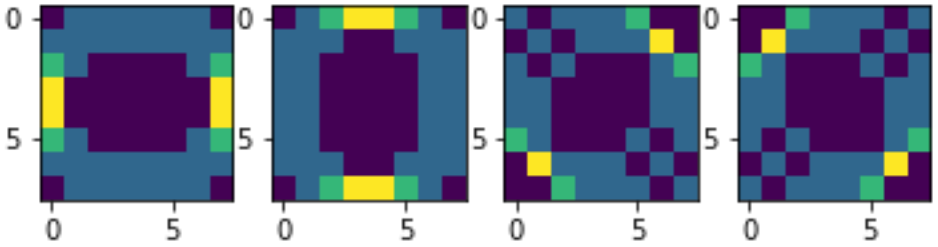
- **다음 그림은** 5 x 5(**가로** x **세로**) **의 특징 맵을** 3 x 3**마다 풀링하는 모습이다.**



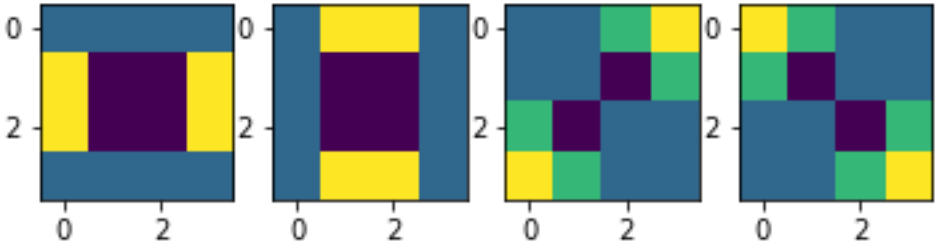
Section 02 CNN

❑ 풀링층

- **예제** 11-2**는** Keras**와** TensorFlow**를 이용하지 않고 풀링층을 정의하고 구현한 코드이다**.
- **다음 그림의 이미지는 앞 절에서 합성곱을 실시한 결과 이미지**(8X8 **크기의 특징 맵**)**이다**.
- **이 특징 맵에 대해** Max**풀링을 실시한다**.



- Max **풀링이 제대로 이루어지면 다음 그림과 같은 특징 맵으로 변환된다**.



Section 02 CNN

❑ 풀링층

- ex11_2.ipynb

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   import urllib.request
4
5   class Conv:
6       def __init__(self, W):
7           self.W = W
8
9       def f_prop(self, X):
10          out = np.zeros((X.shape[0] - 2, X.shape[1] - 2))
11          for i in range(out.shape[0]):
12              for j in range(out.shape[1]):
13                  x = X[i:i+3, j:j+3]
14                  out[i, j] = np.dot(self.W.flatten(), x.flatten())
15          return out
16
```

Section 02 CNN

❑ 풀링층

- ex11_2.ipynb

```
17  class Pool:
18      def __init__(self, l):
19          self.l = l
20
21      def f_prop(self, X):
22          l = self.l
23          out = np.zeros((X.shape[0]//self.l, X.shape[1]//self.l))
24          for i in range(out.shape[0]):
25              for j in range(out.shape[1]):
26                  out[i, j] = np.max(X[i*l : (i+1)*l, j*l : (j+1)*l])
27          return out
28
29  local_filename, headers = urllib.request.urlretrieve('https://aidemystorageprd.blob.core.windows.net/data/5100_cnn_data/circle.npy')
30  X = np.load(local_filename)
31
32  plt.imshow(X)
33  plt.title('The original image', fontsize=12)
34  plt.show()
```

❑ 풀링층

- ex11_2.ipynb

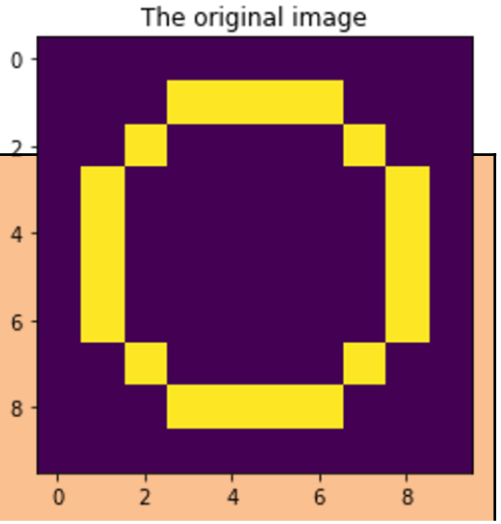
```
35
36  W1 = np.array([[0, 1, 0],
                   [0, 1, 0],
                   [0, 1, 0]])
37  W2 = np.array([[0, 0, 0],
                   [1, 1, 1],
                   [0, 0, 0]])
38  W3 = np.array([[1, 0, 0],
                   [0, 1, 0],
                   [0, 0, 1]])
39  W4 = np.array([[0, 0, 1],
                   [0, 1, 0],
                   [1, 0, 0]])
40
41  conv1 = Conv(W1); C1 = conv1.f_prop(X)
42  conv2 = Conv(W2); C2 = conv2.f_prop(X)
43  conv3 = Conv(W3); C3 = conv3.f_prop(X)
44  conv4 = Conv(W4); C4 = conv4.f_prop(X)
45
```

Section 02 CNN

❑ 풀링층

- ex11_2.ipynb

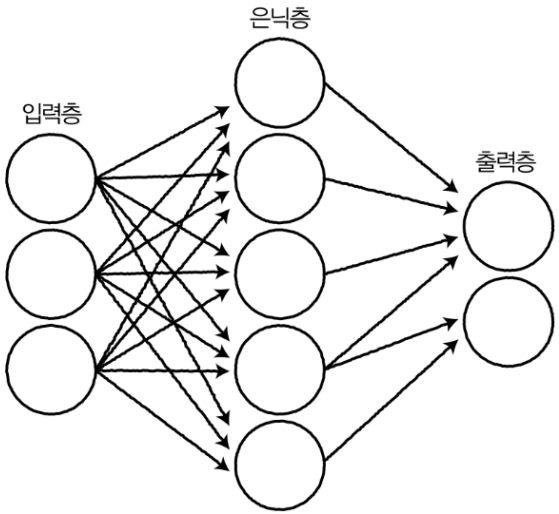
```
46  plt.subplot(1, 4, 1); plt.imshow(C1)
47  plt.subplot(1, 4, 2); plt.imshow(C2)
48  plt.subplot(1, 4, 3); plt.imshow(C3)
49  plt.subplot(1, 4, 4); plt.imshow(C4)
50  plt.suptitle("Convolution result", fontsize=12)
51  plt.show()
52
53  pool = Pool(2)
54  P1 = pool.f_prop(C1)
55  P2 = pool.f_prop(C2)
56  P3 = pool.f_prop(C3)
57  P4 = pool.f_prop(C4)
58
59  plt.subplot(1, 4, 1); plt.imshow(P1)
60  plt.subplot(1, 4, 2); plt.imshow(P2)
61  plt.subplot(1, 4, 3); plt.imshow(P3)
62  plt.subplot(1, 4, 4); plt.imshow(P4)
63  plt.suptitle("Pooling result", fontsize=12)
64  plt.show()
```



Section 02 CNN

❑ CNN 구현

- **예제** 11-3**은** Keras**와**TensorFlow**를 사용하여** CNN**을 구현한다**.
- **실무에서는 이러한 라이브러리를 사용하여 모델을 구현하는 경우가 대부분이다**.



Section 02 CNN

❑ CNN 구현

- Keras에서는 먼저 모델을 관리하는 인스턴스를 만들고, add () 메서드로 모델의 층을 하나씩 추가한다.
- 먼저 인스턴스를 만든다.

```
model=Sequential()
```

- 다음처럼 add () 메서드를 사용하여 모델의 층을 하나씩 추가한다.
- 전결합층은 다음과 같이 정의한다.

```
model.add(Dense(128, activation='relu'))
```

- 합성곱층은 다음과 같이 추가한다(64종의 3x3 필터를 입력 이미지에 적용하여 128가지를 출력한다는 뜻이다).

```
model.add(Conv2D(filters=64, kernel_size=(3, 3)))
```

Section 02 CNN

❑ CNN 구현

- 풀링층은 다음과 같이 추가한다.

```
model.add(MaxPooling2D(pool_size=(2, 2)))
```

- 마지막으로 컴파일을 하면 신경망 모델의 생성이 종료된다.

```
model.compile(optimizer=sgd, loss="categorical_crossentropy", metrics=["accuracy"])
```

- 다음과 같이 기술하면 앞 장의 그림처럼 모델의 구조를 나타낸 표가 출력된다.

```
model.summary()
```

Section 02 CNN

❑ CNN 구현

- ex11_3.ipynb

```
1  from tensorflow.keras.layers import Activation, Conv2D, Dense, Flatten, MaxPooling2D
2  from tensorflow.keras.models import Sequential, load_model
3  from tensorflow.keras.utils import to_categorical
4
5  model = Sequential()
6  model.add(Conv2D(input_shape=(28, 28, 1), filters=32, kernel_size=(2, 2), strides=(1, 1), padding="same"
   , activation='relu'))
7  model.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1)))
8  model.add(Conv2D(filters=32, kernel_size=(2, 2), strides=(1, 1), padding="same", activation='relu'))
9  model.add(MaxPooling2D(pool_size=(2, 2), strides=(1, 1)))
10 model.add(Flatten())
```

Section 02 CNN

❑ CNN 구현

- ex11_3.ipynb

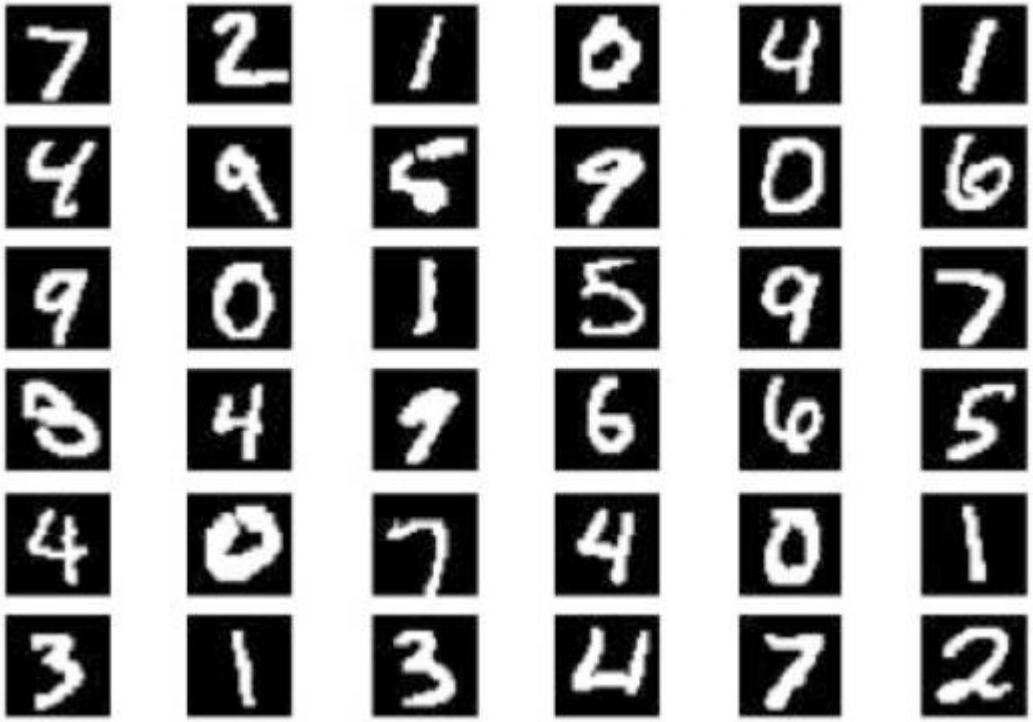
```
11  model.add(Dense(256, activation='relu'))
12  model.add(Dense(128, activation='relu'))
13  model.add(Dense(10, activation='softmax'))
14
15  model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_3 (Conv2D)            (None, 28, 28, 32)        160
max_pooling2d_2 (MaxPooling  (None, 27, 27, 32)        0
2D)
conv2d_4 (Conv2D)            (None, 27, 27, 32)        4128
max_pooling2d_3 (MaxPooling  (None, 26, 26, 32)        0
2D)
flatten_1 (Flatten)          (None, 21632)             0
dense_3 (Dense)              (None, 256)               5538048
dense_4 (Dense)              (None, 128)               32896
dense_5 (Dense)              (None, 10)                1290
=================================================================
Total params: 5,576,522
Trainable params: 5,576,522
Non-trainable params: 0
_________________________________________________________________
```

Section 02 CNN

❑ CNN을 이용한 분류

- MNIST는 그림과 같은 필기체 숫자의 데이터셋이다.
- 각 이미지는 크기가 28 x 28 픽셀로 1채널 (흑백)의 데이터이며, 각각 0~9의 클래스 라벨이 있다.
- CNN으로 MNIST 데이터 셋을 분류해보자.



Section 02 CNN

❑ CNN을 이용한 분류

- 예제 11-4는 텐서플로우로 그림과 같은 구조의 모델을 구축하고 실행하는 프로그램이다.

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_7 (Conv2D)            (None, 26, 26, 32)        320

 conv2d_8 (Conv2D)            (None, 24, 24, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 12, 12, 64)        0
 2D)

 dropout (Dropout)            (None, 12, 12, 64)        0

 flatten_2 (Flatten)          (None, 9216)              0

 dense_6 (Dense)              (None, 128)               1179776

 dropout_1 (Dropout)          (None, 128)               0

 dense_7 (Dense)              (None, 10)                1290

=================================================================
Total params: 1,199,882
Trainable params: 1,199,882
Non-trainable params: 0
_________________________________________________________________
```

Section 02 CNN

❑ CNN을 이용한 분류

- **예제 11-4의 각층의 파라미터는 다음처럼 지정한다.**
  - Conv2D(32, kernel_size=(3, 3), input_shape=(28,28,1))
  - Activation ('relu')
  - Conv2D(filters=64, kernel_size=(3, 3))
  - Activation('relu')
  - MaxPooling2D(pool_size=(2, 2))
  - Dropout(0.25)
  - Flatten ()
  - Dense (128)
  - Activation('relu')
  - Dropout (0.5)
  - Dense(10)

Section 02 CNN

❑ CNN을 이용한 분류

- ex11_4.ipynb

```
1  from tensorflow.keras.datasets import mnist
2  from tensorflow.keras.layers import Dropout, Conv2D, Dense, Flatten, MaxPooling2D
3  from tensorflow.keras.models import Sequential
4  from tensorflow.keras.utils import to_categorical, plot_model
5
6  import numpy as np
7  import matplotlib.pyplot as plt
8
9  (X_train, y_train), (X_test, y_test) = mnist.load_data()
10
11 X_train = X_train.reshape(-1, 28, 28, 1)
12 X_test = X_test.reshape(-1, 28, 28, 1)
13 y_train = to_categorical(y_train)
14 y_test = to_categorical(y_test)
15
```

Section 02 CNN

❑ CNN을 이용한 분류

- ex11_4.ipynb

```
16  model = Sequential()
17  model.add(Conv2D(input_shape=(28, 28, 1), filters=32, kernel_size=(3, 3), activation='relu'))
18  model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
19  model.add(MaxPooling2D(pool_size=(2, 2)))
20  model.add(Dropout(0.25))
21  model.add(Flatten())
22  model.add(Dense(128, activation='relu'))
23  model.add(Dropout(0.5))
24  model.add(Dense(10, activation='softmax'))
25
26  model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
27  model.fit(X_train, y_train, batch_size=128, epochs=1, verbose=1)
28
29  scores = model.evaluate(X_test, y_test, verbose=1)
30  print('Test loss:', scores[0])
31  print('Test accuracy', scores[1])
```

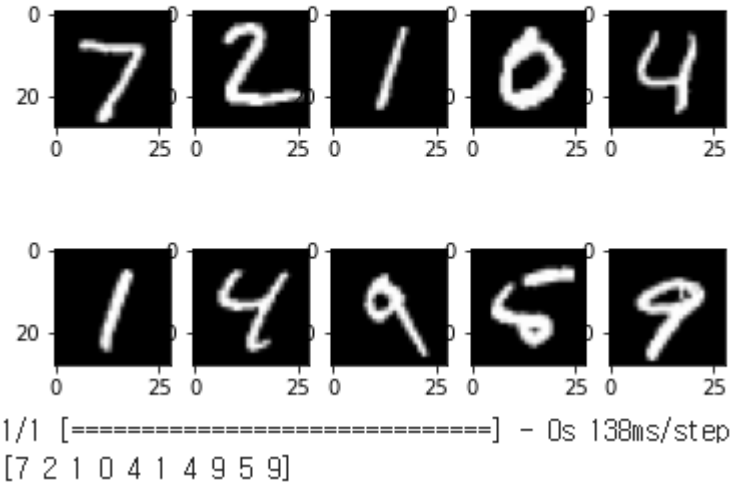
Section 02 CNN

❑ CNN을 이용한 분류

- ex11_4.ipynb

```
32
33  for i in range(10):
34      plt.subplot(2, 5, i+1)
35      plt.imshow(X_test[i].reshape((28, 28)), 'gray')
36  plt.suptitle("The first ten of the test data", fontsize=20)
37  plt.show()
38
39  pred = np.argmax(model.predict(X_test[0:10]), axis=1)
40  print(pred)
```

```
Test loss: 0.07592635601758957
Test accuracy 0.9756000041961670
```




```
1/1 [==============================] - 0s 138ms/step
[7 2 1 0 4 1 4 9 5 9]
```

Section 02 CNN

❑ CNN을 이용한 분류(cifar10)

- cifar10은 그림의 사진처럼 10종류의 개체가 있는 이미지 데이터셋이다.
- 각 이미지는 32 X 32 픽셀로 3채널 (R, G, B)의 데이터이며, 각각 0~9의 클래스 라벨이 붙어 있다.

Section 02 CNN

❑ CNN을 이용한 분류(cifar10)

- 각 클래스 라벨에 해당하는 개체는 다음과 같다.
  - 0 : 비행기
  - 1 : 자동차
  - 2 : 새
  - 3 : 고양이
  - 4 : 사슴
  - 5 : 개
  - 6 : 개구리
  - 7 : 말
  - 8 : 선박
  - 9 : 트럭

Section 02 CNN

❑ CNN을 이용한 분류(cifar10)

- **예제 11-5는 텐서플로우로 그림과 같은 구조의 모델을 구축한다.**

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_4 (Conv2D)            (None, 32, 32, 32)        896

conv2d_5 (Conv2D)            (None, 30, 30, 32)        9248

max_pooling2d_2 (MaxPooling  (None, 15, 15, 32)        0
2D)

dropout_3 (Dropout)          (None, 15, 15, 32)        0

conv2d_6 (Conv2D)            (None, 15, 15, 64)        18496

conv2d_7 (Conv2D)            (None, 13, 13, 64)        36928

max_pooling2d_3 (MaxPooling  (None, 6, 6, 64)          0
2D)

dropout_4 (Dropout)          (None, 6, 6, 64)          0

flatten_1 (Flatten)          (None, 2304)              0

dense_2 (Dense)              (None, 512)               1180160

dropout_5 (Dropout)          (None, 512)               0

dense_3 (Dense)              (None, 10)                5130

=================================================================
Total params: 1,250,858
Trainable params: 1,250,858
Non-trainable params: 0
```

Section 02 CNN

❑ CNN을 이용한 분류(cifar10)

- **예제 11-5 모델의 각층의 파라미터는 다음과 같다.**
  - Conv2D(32, kernel_size=(3, 3), input_shape=(32, 32, 3), activation='relu', padding='same')
  - Conv2D(filters=32, kernel_size=(3, 3), activation='relu')
  - MaxPooling2D(pool_size=(2, 2))
  - Dropout(0.25)
  - Conv2D(64, (3, 3), padding='same', activation='relu')
  - Conv2D(64, (3, 3), activation='relu')
  - MaxPooling2D(pool_size=(2, 2))
  - Dropout(0.25)
  - Flatten ()
  - Dense (512, activation='relu')
  - Dropout (0.5)
  - Dense(10, activation-'softmax')

Section 02 CNN

❑ CNN을 이용한 분류(cifar10)

- ex11_5.ipynb

```
1  from tensorflow.keras.datasets import cifar10
2  from tensorflow.keras.layers import Dropout, Conv2D, Dense, Flatten, MaxPooling2D
3  from tensorflow.keras.models import Sequential
4  from tensorflow.keras.utils import to_categorical
5
6  import numpy as np
7  import matplotlib.pyplot as plt
8
9  (X_train, y_train), (X_test, y_test) = cifar10.load_data()
10
11 y_train = to_categorical(y_train)
12 y_test = to_categorical(y_test)
13
14 model = Sequential()
15 model.add(Conv2D(input_shape=(32, 32, 3), filters=32, kernel_size=(3, 3), activation='relu', padding='same'))
16 model.add(Conv2D(filters=32, kernel_size=(3, 3), activation='relu'))
17 model.add(MaxPooling2D(pool_size=(2, 2)))
18 model.add(Dropout(0.25))
```

Section 02 CNN

❑ CNN을 이용한 분류(cifar10)

- ex11_5.ipynb

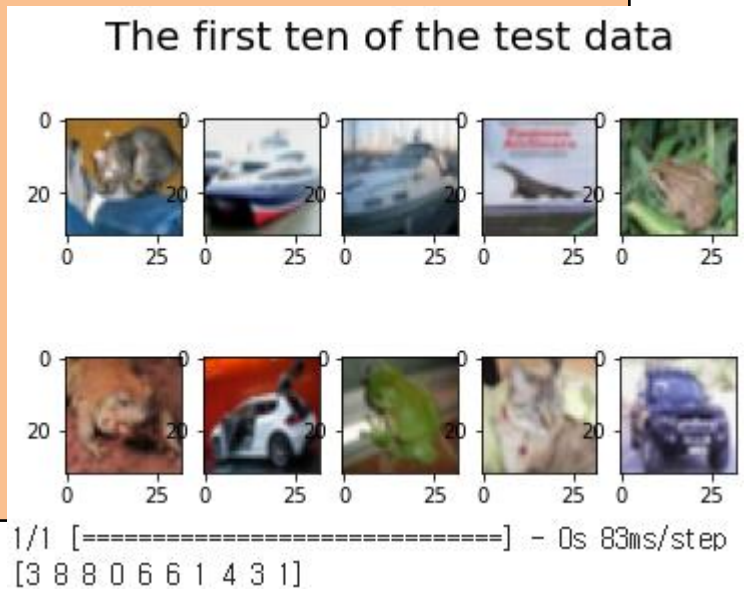
```
19  model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu', padding='same'))
20  model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'))
21  model.add(MaxPooling2D(pool_size=(2, 2)))
22  model.add(Dropout(0.25))
23  model.add(Flatten())
24  model.add(Dense(512, activation='relu'))
25  model.add(Dropout(0.5))
26  model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
27  model.add(Dense(10, activation='softmax'))
28
29  model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
30  model.fit(X_train, y_train, batch_size=32, epochs=5)
31
32  model.save_weights('param_cifar10.hdf5')
33
```

Section 02 CNN

❑ CNN을 이용한 분류(cifar10)

- ex11_5.ipynb

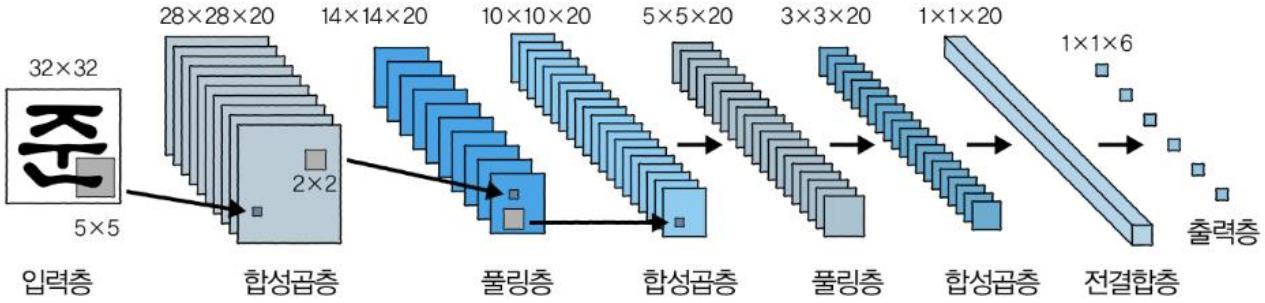
```
34  scores = model.evaluate(X_test, y_test, verbose=1)
35  print('Test loss: ', scores[0])
36  print('Test accuracy: ', scores[1])
37
38  for i in range(10):
39      plt.subplot(2, 5, i+1)
40      plt.imshow(X_test[i])
41  plt.suptitle("The first ten of the test data", fontsize=20)
42  plt.show()
43
44  pred = np.argmax(model.predict(X_test[0:10]), axis=1)
45  print(pred)
```



Section 03 하이퍼파라미터

❑ filters(합성곱층)

- **합성곱층의** filters **파라미터는 특징 맵의 수(추출할 특징의 종류)를 지정한다.**
- **다음 그림처럼 첫 번째 합성곱층에서** filters**는** 20**이고, 두 번째 합성곱층에서도** filters**는** 20**이다.**
- Filters**가 너무 작아서 필요한 특징을 추출하지 못하면 학습을 잘 진행되지 않지만 반대로 너무 크면 과적합하기 쉬우므로 주의해야 한다.**



Section 03 하이퍼파라미터

❑ filters(합성곱층)

- ex11_6.ipynb

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   import urllib.request
4
5   class Conv:
6       def __init__(self, filters):
7           self.filters = filters
8           self.W = np.random.rand(filters, 3, 3)
9
10      def f_prop(self, X):
11          out = np.zeros((filters, X.shape[0] - 2, X.shape[1] - 2))
12          for k in range(self.filters):
13              for i in range(out[0].shape[0]):
14                  for j in range(out[0].shape[1]):
15                      x = X[i:i+3, j:j+3]
16                      out[k, i, j] = np.dot(self.W[k].flatten(), x.flatten())
17          return out
18
```

Section 03 하이퍼파라미터

❑ filters(합성곱층)

- ex11_6.ipynb

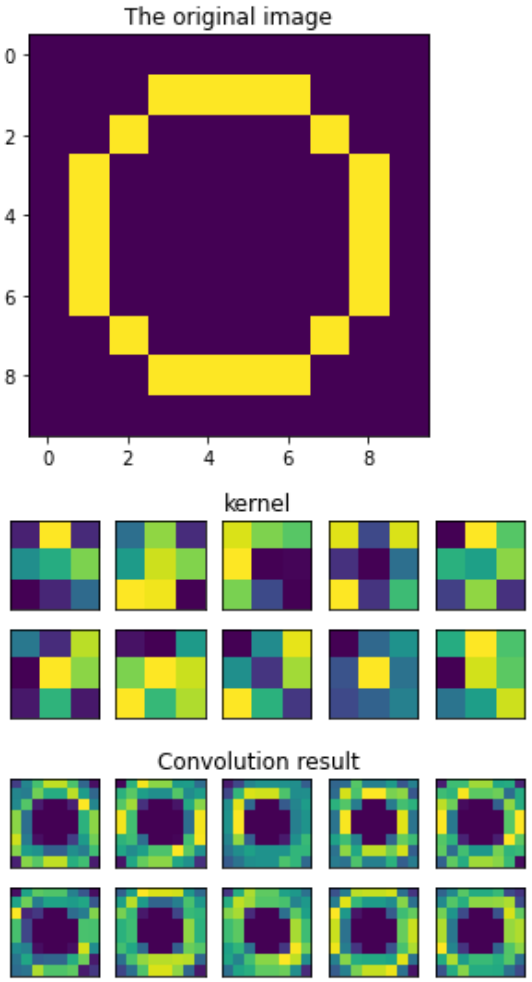
```
19  local_filename, headers = urllib.request.urlretrieve('https://aidemystorageprd.blob.core.windows.net/data/
    5100_cnn_data/circle.npy')
20  X = np.load(local_filename)
21
22  filters = 10
23  conv = Conv(filters=filters)
24  C = conv.f_prop(X)
25
26  plt.imshow(X)
27  plt.title("The original image", fontsize=12)
28  plt.show()
29
30  plt.figure(figsize=(5, 2))
31  for i in range(filters):
32      plt.subplot(2, filters//2, i+1)
33      ax = plt.gca()
34      ax.get_xaxis().set_visible(False)
35      ax.get_yaxis().set_visible(False)
36      plt.imshow(conv.W[i])
37  plt.suptitle('kernel', fontsize=12)
```

Section 03 하이퍼파라미터

❑ filters(합성곱층)

- ex11_6.ipynb

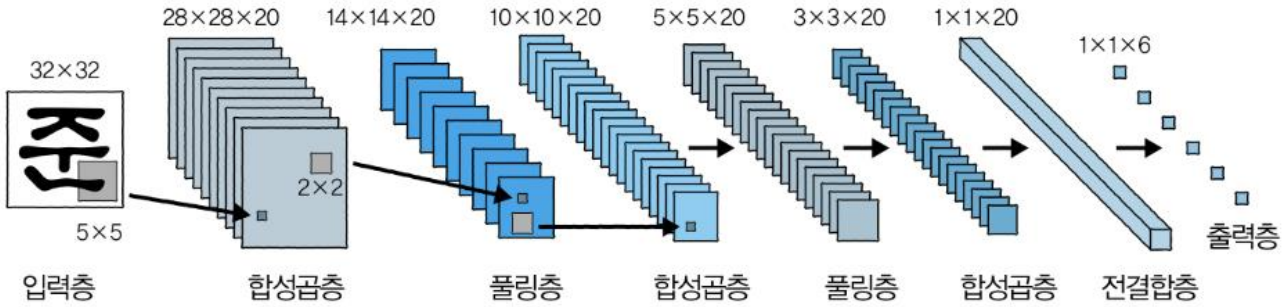
```
38  plt.show()
39
40  plt.figure(figsize=(5, 2))
41  for i in range(filters):
42      plt.subplot(2, filters//2, i+1)
43      ax = plt.gca()
44      ax.get_xaxis().set_visible(False)
45      ax.get_yaxis().set_visible(False)
46      plt.imshow(C[i])
47  plt.suptitle('Convolution result', fontsize=12)
48  plt.show()
```



Section 03 하이퍼파라미터

❑ kernel_size(합성곱층)

- **합성곱층의** kernel_size **파라미터로 커널의 크기를 지정한다**.
- **다음 그림의 첫 번째 합성곱층에서** kernel_size**는** 5 x 5**이다**.



- kernel_size**가 너무 작으면 극히 작은 특징도 검출할 수 없게 되어 제대로 학습할 수 없다**.
- **반대로 너무 크면 원래 작은특징의 모임으로 검출될 예정이었던 큰 특징까지 검출되어 계층구조파악에 자신 있는 신경망모델의 강점을 살리지 못하고 비효율적인 모델이 되어버린다**.

Section 03 하이퍼파라미터

❑ kernel_size(합성곱층)

- ex11_7.ipynb

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   import urllib.request
4
5   class Conv:
6     def __init__(self, filters, kernel_size):
7       self.filters = filters
8       self.kernel_size = kernel_size
9       self.W = np.random.rand(filters, kernel_size[0], kernel_size[1])
10
11    def f_prop(self, X):
12      k_h, k_w = self.kernel_size
13      out = np.zeros((filters, X.shape[0] - k_h + 1, X.shape[1] - k_w + 1))
14      for k in range(self.filters):
15        for i in range(out[0].shape[0]):
16          for j in range(out[0].shape[1]):
17            x = X[i:i+k_h, j:j+k_w]
18            out[k, i, j] = np.dot(self.W[k].flatten(), x.flatten())
19      return out
20
```

Section 03 하이퍼파라미터

❑ kernel_size(합성곱층)

- ex11_7.ipynb

```
21  local_filename, headers = urllib.request.urlretrieve('https://aidemystorageprd.blob.core.windows.net/data/5100_
    cnn_data/circle.npy')
22  X = np.load(local_filename)
23
24  filters = 4
25  kernel_size = (3, 3)
26
27  conv1 = Conv(filters=filters, kernel_size=kernel_size)
28  C1 = conv1.f_prop(X)
29
30  filters = 4
31  kernel_size = (6, 6)
32
33  conv2 = Conv(filters=filters, kernel_size=kernel_size)
34  C2 = conv2.f_prop(X)
35
36  plt.imshow(X)
37  plt.title("The original image", fontsize=12)
38  plt.show()
39
```

Section 03 하이퍼파라미터

❑ kernel_size(합성곱층)

- ex11_7.ipynb

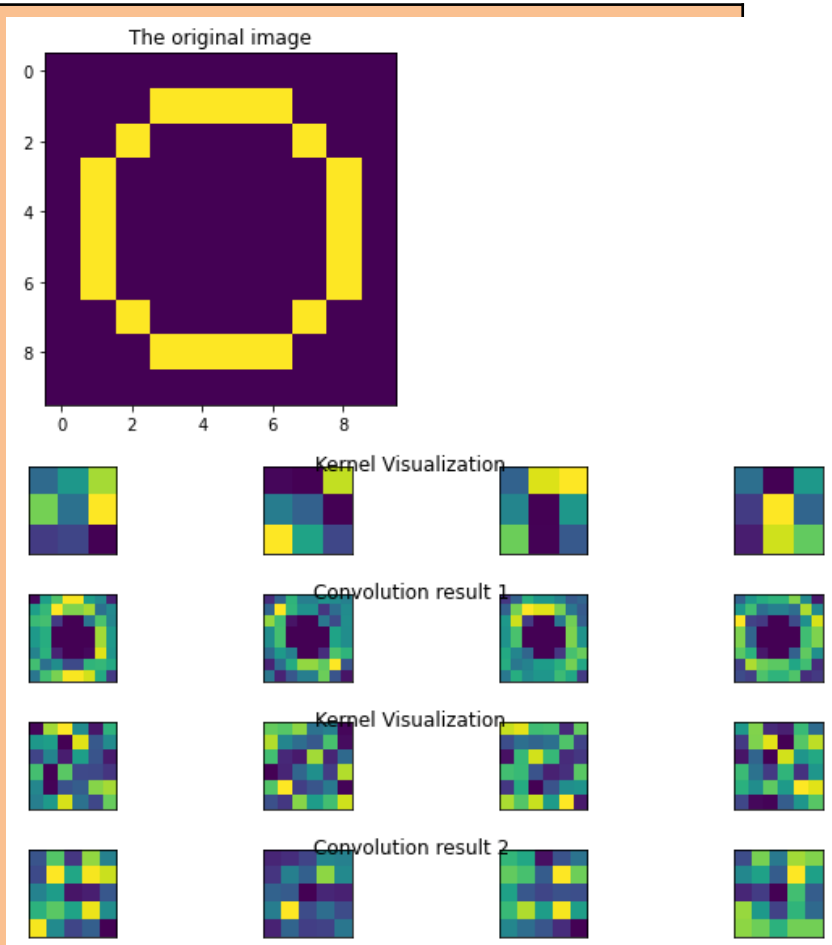
```
40  plt.figure(figsize=(10, 1))
41  for i in range(filters):
42      plt.subplot(1, filters, i+1)
43      ax = plt.gca()
44      ax.get_xaxis().set_visible(False)
45      ax.get_yaxis().set_visible(False)
46      plt.imshow(conv1.W[i])
47  plt.suptitle('Kernel Visualization', fontsize=12)
48  plt.show()
49
50  plt.figure(figsize=(10, 1))
51  for i in range(filters):
52      plt.subplot(1, filters, i+1)
53      ax = plt.gca()
54      ax.get_xaxis().set_visible(False)
55      ax.get_yaxis().set_visible(False)
56      plt.imshow(C1[i])
57  plt.suptitle('Convolution result 1', fontsize=12)
58  plt.show()
59
```

Section 03 하이퍼파라미터

❑ kernel_size(합성곱층)

- ex11_7.ipynb

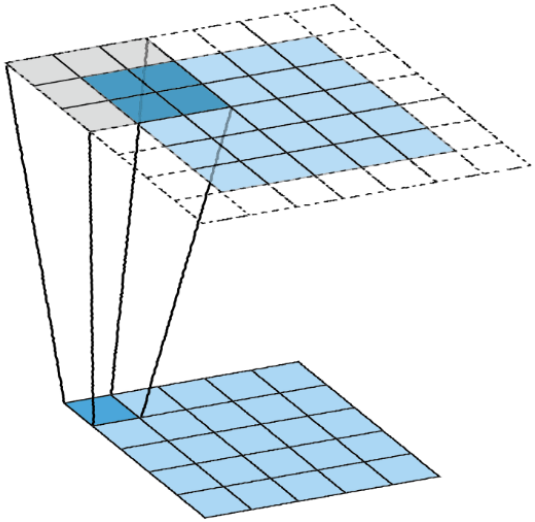
```
60  plt.figure(figsize=(10, 1))
61  for i in range(filters):
62      plt.subplot(1, filters, i+1)
63      ax = plt.gca()
64      ax.get_xaxis().set_visible(False)
65      ax.get_yaxis().set_visible(False)
66      plt.imshow(conv2.W[i])
67  plt.suptitle('Kernel Visualization', fontsize=12)
68  plt.show()
69
70  plt.figure(figsize=(10, 1))
71  for i in range(filters):
72      plt.subplot(1, filters, i+1)
73      ax = plt.gca()
74      ax.get_xaxis().set_visible(False)
75      ax.get_yaxis().set_visible(False)
76      plt.imshow(C2[i])
77  plt.suptitle('Convolution result 2', fontsize=12)
78  plt.show()
```



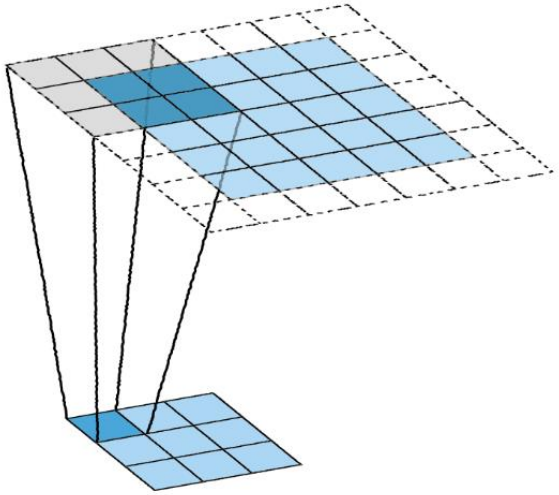
Section 03 하이퍼파라미터

❑ strides(합성곱층)

- **합성곱층의** strides **파라미터는 특징을 추출하는 간격, 즉 커널을 이동하는 거리를 지정한다.**
- strides**가 작을수록 세부적인 특징량을 추출할 수 있지만 이미지 내의 동일한 위치에 같은 특징을 여러 번 감지해버리는 등 불필요한 계산이 많아질 수 있다.**
- **그러나 일반적으로** strides **는 작은 편이 좋다고 여겨져** Keras**의** Conv2D **층에서** strides**는 기본으로** (1, 1)**로 되어 있다.**



strides : (1, 1)



strides : (2, 2)

Section 03 하이퍼파라미터

❑ strides(합성곱층)

- ex11_8.ipynb

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   import urllib.request
4
5   class Conv:
6       def __init__(self, filters, kernel_size, strides):
7           self.filters = filters
8           self.kernel_size = kernel_size
9           self.strides = strides
10          self.W = np.random.rand(filters, kernel_size[0], kernel_size[1])
11
```

Section 03 하이퍼파라미터

❑ strides(합성곱층)

- ex11_8.ipynb

```
12      def f_prop(self, X):
13          k_h = self.kernel_size[0]
14          k_w = self.kernel_size[1]
15          s_h = self.strides[0]
16          s_w = self.strides[1]
17          out = np.zeros((filters, (X.shape[0]-k_h)//s_h+1,  (X.shape[1]-k_w)//s_w+1))
18          for k in range(self.filters):
19              for i in range(out[0].shape[0]):
20                  for j in range(out[0].shape[1]):
21                      x = X[i*s_h:i*s_h+k_h, j*s_w:j*s_w+k_w]
22                      out[k, i, j] = np.dot(self.W[k].flatten(), x.flatten())
23          return out
24
25  local_filename, headers = urllib.request.urlretrieve('https://aidemystorageprd.blob.core.windows.net/data/
    5100_cnn_data/circle.npy')
26  X = np.load(local_filename)
27
```

Section 03 하이퍼파라미터

❑ strides(합성곱층)

- ex11_8.ipynb

```
28  filters = 4
29  kernel_size = (3, 3)
30  strides = (1,1)
31  conv1 = Conv(filters=filters, kernel_size=kernel_size, strides=strides)
32  C1 = conv1.f_prop(X)
33
34  filters = 4
35  kernel_size = (3,3)
36  strides = (2,2)
37  conv2 = Conv(filters=filters, kernel_size=kernel_size, strides=strides)
38  conv2.W = conv1.W
39  C2 = conv2.f_prop(X)
40
41  plt.imshow(X)
42  plt.title("The original image", fontsize=12)
43  plt.show()
44
45  plt.figure(figsize=(10, 1))
```

Section 03 하이퍼파라미터

❑ strides(합성곱층)

- ex11_8.ipynb

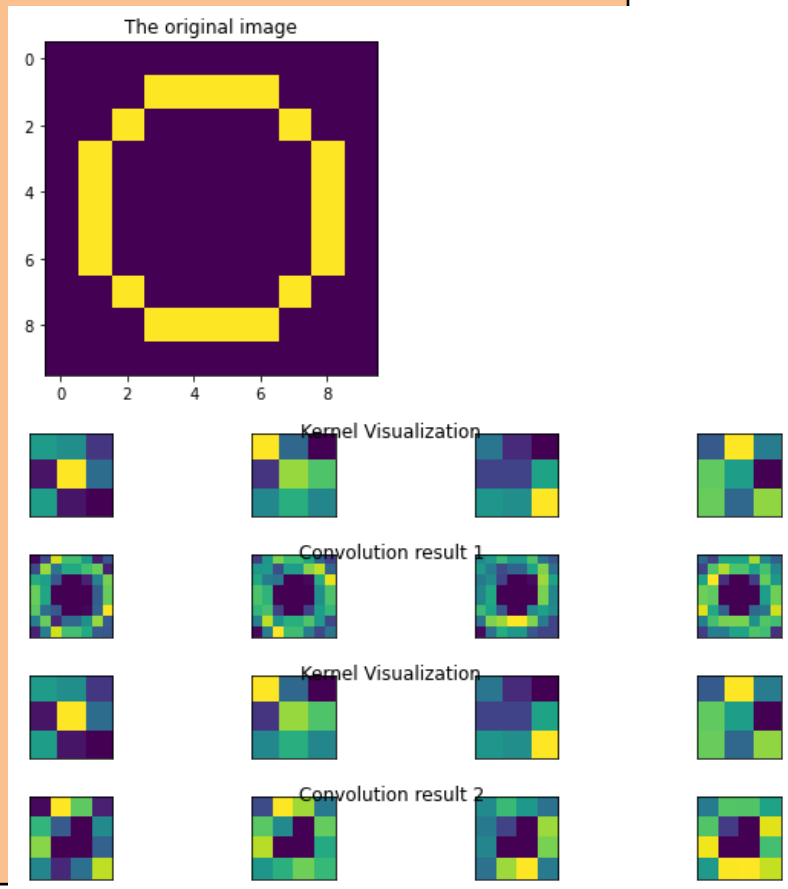
```
46  for i in range(filters):
47      plt.subplot(1, filters, i+1)
48      ax = plt.gca()
49      ax.get_xaxis().set_visible(False)
50      ax.get_yaxis().set_visible(False)
51      plt.imshow(conv1.W[i])
52  plt.suptitle('Kernel Visualization', fontsize=12)
53  plt.show()
54
55  plt.figure(figsize=(10, 1))
56  for i in range(filters):
57      plt.subplot(1, filters, i+1)
58      ax = plt.gca()
59      ax.get_xaxis().set_visible(False)
60      ax.get_yaxis().set_visible(False)
61      plt.imshow(C1[i])
62  plt.suptitle('Convolution result 1', fontsize=12)
63  plt.show()
64
```

Section 03 하이퍼파라미터

❑ strides(합성곱층)

- ex11_8.ipynb

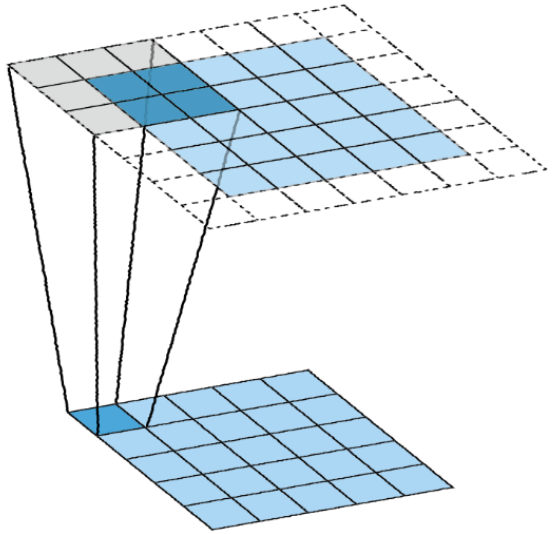
```
65  plt.figure(figsize=(10, 1))
66  for i in range(filters):
67      plt.subplot(1, filters, i+1)
68      ax = plt.gca()
69      ax.get_xaxis().set_visible(False)
70      ax.get_yaxis().set_visible(False)
71      plt.imshow(conv2.W[i])
72  plt.suptitle('Kernel Visualization', fontsize=12)
73  plt.show()
74
75  plt.figure(figsize=(10, 1))
76  for i in range(filters):
77      plt.subplot(1, filters, i+1)
78      ax = plt.gca()
78      ax.get_xaxis().set_visible(False)
75      ax.get_yaxis().set_visible(False)
76      plt.imshow(C2[i])
77  plt.suptitle('Convolution result 2', fontsize=12)
78  plt.show()
```



Section 03 하이퍼파라미터

❑ padding(합성곱층)

- 패딩은 입력 이미지의 주변을 0으로 채우는 것을 말한다.
- 패딩에 의해 가장자리 데이터의 특징도 잘 고려되게 된다.
- 그 외에도 데이터 갱신 빈도가 올라가고, 각 층의 입출력 유닛 수를 조정할 수 있는 장점이 존재한다.
- 다음 그림의 하늘색 패널 주위에 흰 테두리는 패딩을 표현한 것이다.
- 상하로 1, 좌우로 1의 패딩을 한 그림이다.



Section 03 하이퍼파라미터

❑ padding(합성곱층)

- Keras**의** Conv2D **층에서** padding=valid, padding=same**과 같은 패딩 방법을 지정한다**.
- padding=valid**의 경우 패딩은 수행되지 않으며**, padding=same**의 경우 출력되는 특징 맵이 입력 크기와 일치하도록 입력에 패딩을 수행한다**.

Section 03 하이퍼파라미터

❑ padding(합성곱층)

- ex11_9.ipynb

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   import urllib.request
4
5   class Conv:
6     def __init__(self, filters, kernel_size, strides, padding):
7       self.filters = filters
8       self.kernel_size = kernel_size
9       self.strides = strides
10      self.padding = padding
11      self.W = np.random.rand(filters, kernel_size[0], kernel_size[1])
12
13    def f_prop(self, X):
14      k_h, k_w = self.kernel_size
15      s_h, s_w = self.strides
16      p_h, p_w = self.padding
17      s_w = self.strides[1]
18      out = np.zeros((filters, (X.shape[0]+p_h*2-k_h)//s_h+1, (X.shape[1]+p_w*2-k_w)//s_w+1))
19      X = np.pad(X, ((p_h, p_h), (p_w, p_w)), 'constant', constant_values=((0,0),(0,0)))
20      self.X = X
```

Section 03 하이퍼파라미터

❑ padding(합성곱층)

- ex11_9.ipynb

```
21          for k in range(self.filters):
22              for i in range(out[0].shape[0]):
23                  for j in range(out[0].shape[1]):
24                      x = X[i*s_h:i*s_h+k_h, j*s_w:j*s_w+k_w]
25                      out[k, i, j] = np.dot(self.W[k].flatten(), x.flatten())
26          return out
27
28  local_filename, headers = urllib.request.urlretrieve('https://aidemystorageprd.blob.core.windows.net/data/
    5100_cnn_data/circle.npy')
29  X = np.load(local_filename)
30
31  filters = 4
32  kernel_size = (3, 3)
33  strides = (1,1)
34  padding = (0,0)
35  conv1 = Conv(filters=filters, kernel_size=kernel_size, strides=strides, padding=padding)
36  C1 = conv1.f_prop(X)
37
```

Section 03 하이퍼파라미터

❑ padding(합성곱층)

- ex11_9.ipynb

```
filters = 4
kernel_size = (3,3)
strides = (1, 1)
padding = (2,2)
conv2 = Conv(filters=filters, kernel_size=kernel_size, strides=strides, padding=padding)
conv2.W = conv1.W
C2 = conv2.f_prop(X)

plt.imshow(conv1.X)
plt.title('Padding result of convolution 1', fontsize=12)
plt.show()

plt.figure(figsize=(10, 1))
for i in range(filters):
    plt.subplot(1, filters, i+1)
    ax = plt.gca()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    plt.imshow(conv1.W[i])
plt.suptitle('Visualization of the convolution 1 kernel', fontsize=12)
plt.show()
```

Section 03 하이퍼파라미터

❑ padding(합성곱층)

- ex11_9.ipynb

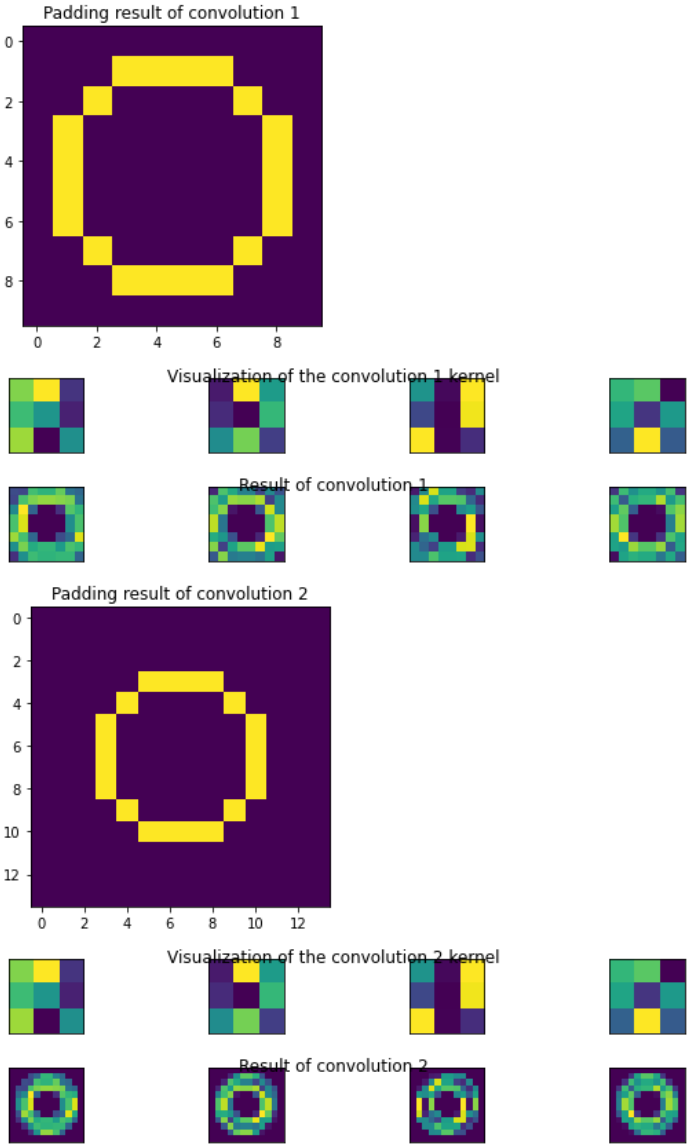
```
59
60  plt.figure(figsize=(10, 1))
61  for i in range(filters):
62      plt.subplot(1, filters, i+1)
63      ax = plt.gca()
64      ax.get_xaxis().set_visible(False)
65      ax.get_yaxis().set_visible(False)
66      plt.imshow(C1[i])
67  plt.suptitle('Result of convolution 1', fontsize=12)
68  plt.show()
69
70  plt.imshow(conv2.X)
71  plt.title('Padding result of convolution 2', fontsize=12)
72  plt.show()
73
74  plt.figure(figsize=(10, 1))
```

Section 03 하이퍼파라미터

❑ padding(합성곱층)

- ex11_9.ipynb

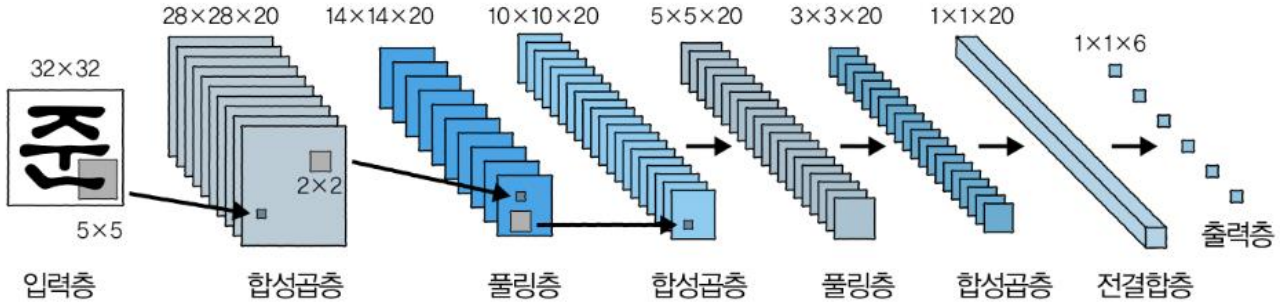
```
75  for i in range(filters):
76      plt.subplot(1, filters, i+1)
77      ax = plt.gca()
78      ax.get_xaxis().set_visible(False)
79      ax.get_yaxis().set_visible(False)
80      plt.imshow(conv2.W[i])
81  plt.suptitle('Visualization of the convolution 2 kernel', fontsize=12)
82  plt.show()
83
84  plt.figure(figsize=(10, 1))
85  for i in range(filters):
86      plt.subplot(1, filters, i+1)
87      ax = plt.gca()
88      ax.get_xaxis().set_visible(False)
89      ax.get_yaxis().set_visible(False)
90      plt.imshow(C2[i])
91  plt.suptitle('Result of convolution 2', fontsize=12)
92  plt.show()
```



Section 03 하이퍼파라미터

❑ pool_size(풀링층)

- **풀링층의** pool_size **파라미터는 풀링의 거칠기를 지정하는 파라미터이다**.
- **다음 그림에서는 첫 풀링의 크기가** 2 x 2**로 되어 있다**.
- pool_size**를 크게 하면 위치에 대한 견고성** (**강건성**, robustness)**이 상승**(**이미지 내에서 개체의 위치가 다소 변화해도 출력이 변화하지 않는 것**)**하지만 기본적으로** pool_size**는**2 x 2**로 하면 좋다고 알려져 있다**.



Section 03 하이퍼파라미터

❑ pool_size(풀링층)

- ex11_10.ipynb

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   import urllib.request
4
5   class Conv:
6     def __init__(self, W, filters, kernel_size):
7       self.filters = filters
8       self.kernel_size = kernel_size
9       self.W = np.random.rand(filters, kernel_size[0], kernel_size[1])
10
11    def f_prop(self, X):
12      k_h, k_w = self.kernel_size
13      out = np.zeros((filters, X.shape[0]-k_h+1, X.shape[1]-k_w+1))
14      for k in range(self.filters):
15        for i in range(out[0].shape[0]):
16          for j in range(out[0].shape[1]):
17            x = X[i:i+k_h, j:j+k_w]
18            out[k,i,j] = np.dot(self.W[k].flatten(), x.flatten())
19      return out
20
```

Section 03 하이퍼파라미터

❑ pool_size(풀링층)

- ex11_10.ipynb

```
21  class Pool:
22      def __init__(self, pool_size):
23          self.pool_size = pool_size
24
25      def f_prop(self, X):
26          k_h, k_w = self.pool_size
27          out = np.zeros((X.shape[0]-k_h+1, X.shape[1]-k_w+1))
28          for i in range(out.shape[0]):
29              for j in range(out.shape[1]):
30                  out[i,j] = np.max(X[i:i+k_h, j:j+k_w])
31          return out
32
33  local_filename, headers = urllib.request.urlretrieve('https://aidemystorageprd.blob.core.windows.net/data/5100_cnn_data/circle.npy')
34  X = np.load(local_filename)
35  local_filename_w, headers = urllib.request.urlretrieve('https://aidemystorageprd.blob.core.windows.net/data/5100_cnn_data/weight.npy')
36  W = np.load(local_filename_w)
37
```

Section 03 하이퍼파라미터

❑ pool_size(풀링층)

- ex11_10.ipynb

```
38  filters = 4
39  kernel_size = (3,3)
40  conv = Conv(W=W, filters=filters, kernel_size=kernel_size)
41  C = conv.f_prop(X)
42
43  pool_size = (2,2)
44  pool1 = Pool(pool_size)
45  P1 = [pool1.f_prop(C[i]) for i in range(len(C))]
46
47  pool_size = (4,4)
48  pool2 = Pool(pool_size)
49  P2 = [pool2.f_prop(C[i]) for i in range(len(C))]
50
51  plt.imshow(X)
52  plt.title('The original image', fontsize=12)
53  plt.show()
54
55  plt.figure(figsize=(10,1))
```

Section 03 하이퍼파라미터

❑ pool_size(풀링층)

- ex11_10.ipynb

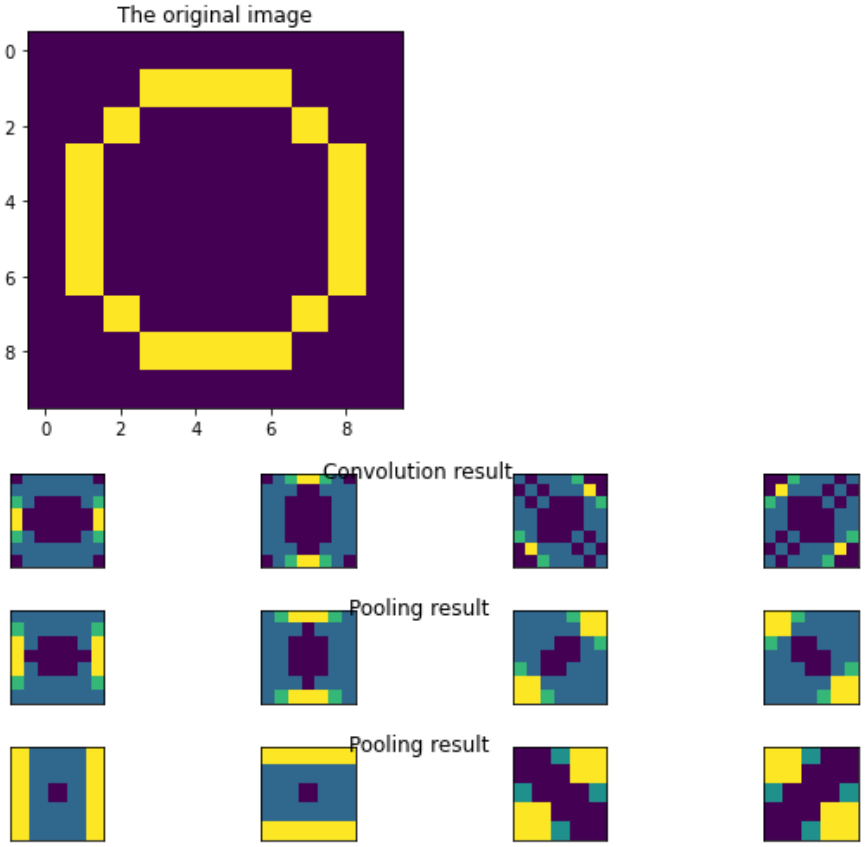
```
for i in range(filters):
    plt.subplot(1, filters, i+1)
    ax = plt.gca()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    plt.imshow(C[i])
plt.suptitle('Convolution result', fontsize=12)
plt.show()
plt.figure(figsize=(10,1))
for i in range(filters):
    plt.subplot(1, filters, i+1)
    ax = plt.gca()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    plt.imshow(P1[i])
plt.suptitle('Pooling result', fontsize=12)
plt.show()

```

Section 03 하이퍼파라미터

❑ pool_size(풀링층)

- ex11_10.ipynb

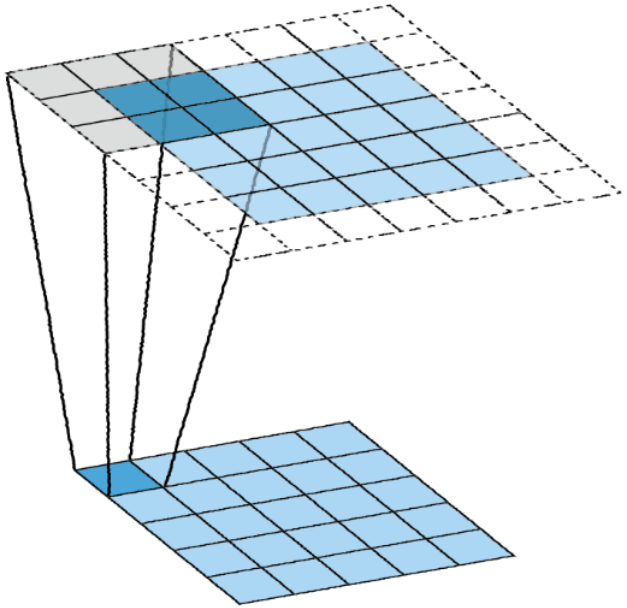
```
74  plt.figure(figsize=(10,1))
75  for i in range(filters):
76      plt.subplot(1, filters, i+1)
77      ax = plt.gca()
78      ax.get_xaxis().set_visible(False)
79      ax.get_yaxis().set_visible(False)
80      plt.imshow(P2[i])
81  plt.suptitle('Pooling result', fontsize=12)
82  plt.show()
```



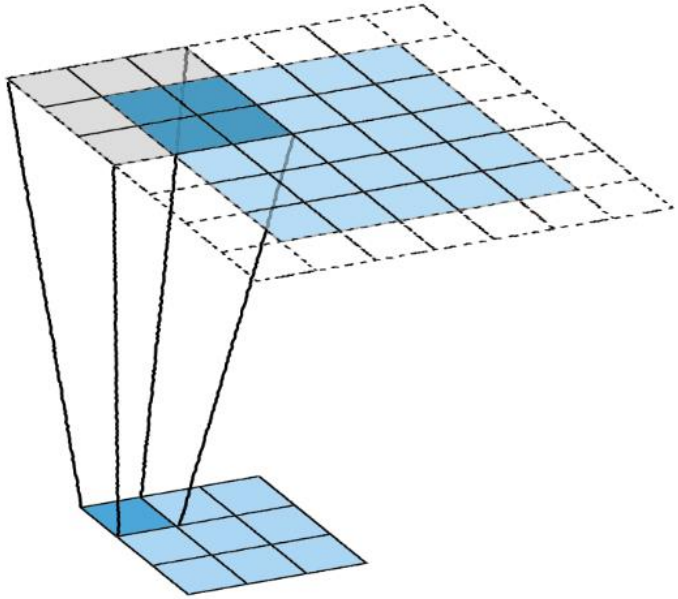
Section 03 하이퍼파라미터

❑ strides(풀링층)

- **풀링층의** strides **파라미터는 합성곱층의** strides **파라미터와 마찬가지로 특징 맵을 풀링하는 간격을 지정한다**.
- **텐서플로우의** Conv2D **층에서** strides**는 기본적으로** pool_size**와 일치하도록 되어 있다**.



Strides : (1, 1)



Strides : (2, 2)

Section 03 하이퍼파라미터

❑ strides(풀링층)

- ex11_11.ipynb

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   import urllib.request
4
5   class Conv:
6       def __init__(self, W, filters, kernel_size):
7           self.filters = filters
8           self.kernel_size = kernel_size
9           self.W = np.random.rand(filters, kernel_size[0], kernel_size[1])
10
11      def f_prop(self, X):
12          k_h, k_w = self.kernel_size
13          out = np.zeros((filters, X.shape[0]-k_h+1, X.shape[1]-k_w+1))
14          for k in range(self.filters):
15              for i in range(out[0].shape[0]):
16                  for j in range(out[0].shape[1]):
17                      x = X[i:i+k_h, j:j+k_w]
18                      out[k,i,j] = np.dot(self.W[k].flatten(), x.flatten())
19          return out
20
```

Section 03 하이퍼파라미터

❑ strides(풀링층)

- ex11_11.ipynb

```
21  class Pool:
22      def __init__(self, pool_size, strides):
23          self.pool_size = pool_size
24          self.strides = strides
25
26      def f_prop(self, X):
27          k_h, k_w = self.pool_size
28          s_h, s_w = self.strides
29          out = np.zeros(((X.shape[0]-k_h)//s_h+1, (X.shape[1]-k_w)//s_w+1))
30          for i in range(out.shape[0]):
31              for j in range(out.shape[1]):
32                  out[i,j] = np.max(X[i*s_h:i*s_h+k_h, j*s_w:j*s_w+k_w])
33          return out
34
35  local_filename, headers = urllib.request.urlretrieve('https://aidemystorageprd.blob.core.windows.net/data/5100_cnn_da
    ta/circle.npy')
36  X = np.load(local_filename)
37  local_filename_w, headers = urllib.request.urlretrieve('https://aidemystorageprd.blob.core.windows.net/data/5100_cnn_
    data/weight.npy')
38  W = np.load(local_filename_w)
```

Section 03 하이퍼파라미터

❑ strides(풀링층)

- ex11_11.ipynb

```
39
40  filters = 4
41  kernel_size = (3,3)
42  conv = Conv(W=W, filters=filters, kernel_size=kernel_size)
43  C = conv.f_prop(X)
44
45  pool_size = (2,2)
46  strides = (1,1)
47  pool1 = Pool(pool_size, strides)
48  P1 = [pool1.f_prop(C[i]) for i in range(len(C))]
49
50  pool_size = (3,3)
51  strides = (2,2)
52  pool2 = Pool((3,3), (2,2))
53  P2 = [pool2.f_prop(C[i]) for i in range(len(C))]
54
55  plt.imshow(X)
56  plt.title('The original image', fontsize=12)
57  plt.show()
58
```

Section 03 하이퍼파라미터

❑ strides(풀링층)

- ex11_11.ipynb

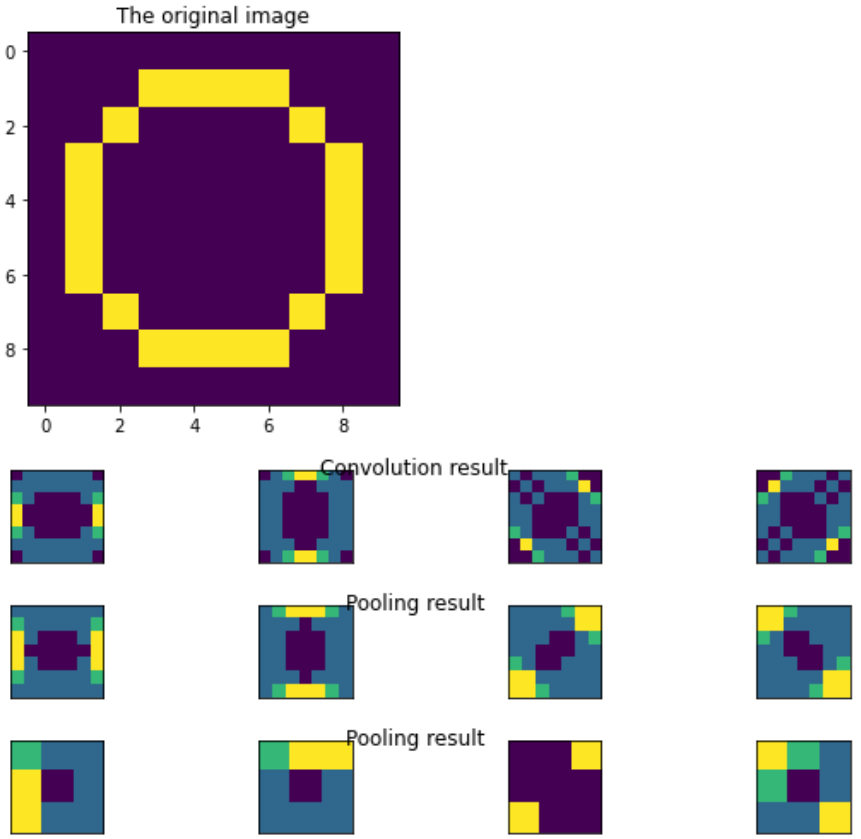
```
plt.figure(figsize=(10,1))
for i in range(filters):
    plt.subplot(1, filters, i+1)
    ax = plt.gca()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    plt.imshow(C[i])
plt.suptitle('Convolution result', fontsize=12)
plt.show()

plt.figure(figsize=(10,1))
for i in range(filters):
    plt.subplot(1, filters, i+1)
    ax = plt.gca()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
    plt.imshow(P1[i])
plt.suptitle('Pooling result', fontsize=12)
plt.show()

```

Section 03 하이퍼파라미터

❑ strides(풀링층)

- ex11_11.ipynb

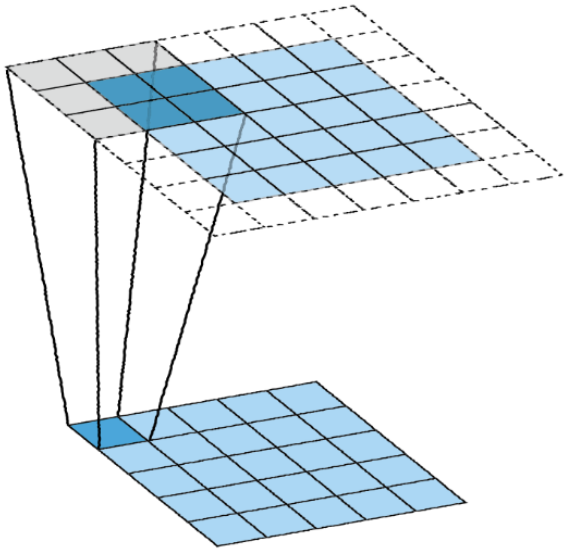
```
79  plt.figure(figsize=(10,1))
80  for i in range(filters):
81      plt.subplot(1, filters, i+1)
82      ax = plt.gca()
83      ax.get_xaxis().set_visible(False)
84      ax.get_yaxis().set_visible(False)
85      plt.imshow(P2[i])
86  plt.suptitle('Pooling result', fontsize=12)
87  plt.show()
```



Section 03 하이퍼파라미터

❑ padding(풀링층)

- **합성곱층의** padding**과 마찬가지로 풀링층의** padding **파라미터는 패딩 방식을 지정한다**.
- **텐서플로우의** MaxPooling2D **층에서는** padding=valid, padding=same**과 같이 패딩 방법을 지정한다**.
- padding=valid**의 경우 패딩은 수행되지 않으며**, padding=same**의 경우 출력되는 특징 맵이 입력 크기와 일치하도록 입력에 패딩을 수행한다**.



Section 03 하이퍼파라미터

❑ padding(풀링층)

- ex11_12.ipynb

```
1   import numpy as np
2   import matplotlib.pyplot as plt
3   import urllib.request
4
5   class Conv:
6       def __init__(self, W, filters, kernel_size):
7           self.filters = filters
8           self.kernel_size = kernel_size
9           self.W = np.random.rand(filters, kernel_size[0], kernel_size[1])
10
11      def f_prop(self, X):
12          k_h, k_w = self.kernel_size
13          out = np.zeros((filters, X.shape[0]-k_h+1, X.shape[1]-k_w+1))
14          for k in range(self.filters):
15              for i in range(out[0].shape[0]):
16                  for j in range(out[0].shape[1]):
17                      x = X[i:i+k_h, j:j+k_w]
18                      out[k,i,j] = np.dot(self.W[k].flatten(), x.flatten())
19          return out
20
```

Section 03 하이퍼파라미터

❑ padding(풀링층)

- ex11_12.ipynb

```
21  class Pool:
22    def __init__(self, pool_size, strides):
23      self.pool_size = pool_size
24      self.strides = strides
25      self.padding = padding
26
27    def f_prop(self, X):
28      k_h, k_w = self.pool_size
29      s_h, s_w = self.strides
30      p_h, p_w = self.padding
31      out = np.zeros(((X.shape[0]-k_h)//s_h+1, (X.shape[1]-k_w)//s_w+1))
32      X = np.pad(X, ((p_h,p_h),(p_w,p_w)), 'constant', constant_values=((0,0),(0,0)))
33      for i in range(out.shape[0]):
34        for j in range(out.shape[1]):
35          out[i,j] = np.max(X[i*s_h:i*s_h+k_h, j*s_w:j*s_w+k_w])
36      return out
37
38  local_filename, headers = urllib.request.urlretrieve('https://aidemystorageprd.blob.core.windows.net/data/
    5100_cnn_data/circle.npy')
39  X = np.load(local_filename)
```

Section 03 하이퍼파라미터

❑ padding(풀링층)

- ex11_12.ipynb

```
40  local_filename_w, headers = urllib.request.urlretrieve('https://aidemystorageprd.blob.core.windows.net/data/5100_
    cnn_data/weight.npy')
41  W = np.load(local_filename_w)
42
43  filters = 4
44  kernel_size = (3,3)
45  conv = Conv(W=W, filters=filters, kernel_size=kernel_size)
46  C = conv.f_prop(X)
47
48  pool_size = (2, 2)
49  strides = (2, 2)
50  padding = (0, 0)
51  pool1 = Pool(pool_size=pool_size, strides=strides, padding=padding)
52  P1 = [pool1.f_prop(C[i]) for i in range(len(C))]
53
54  pool_size = (2,2)
55  strides = (2,2)
56  padding = (1,1)
57  pool2 = Pool(pool_size=pool_size, strides=strides, padding=padding)
58  P2 = [pool2.f_prop(C[i]) for i in range(len(C))]
59
```

Section 03 하이퍼파라미터

❑ padding(풀링층)

- ex11_12.ipynb

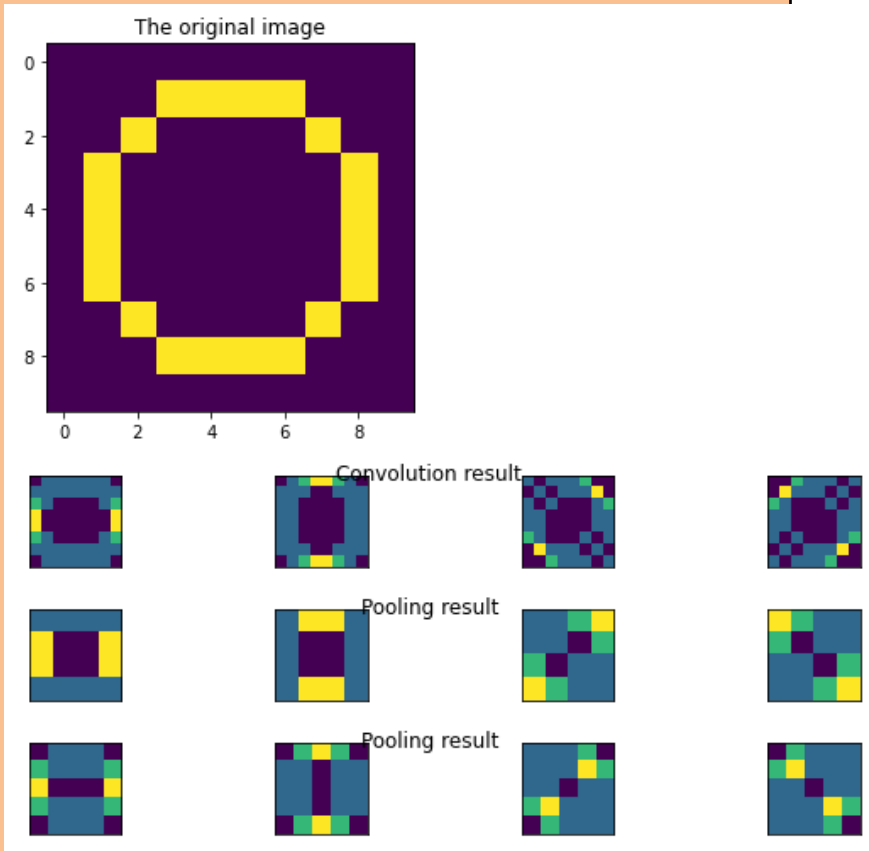
```
60  plt.imshow(X)
61  plt.title('The original image', fontsize=12)
62  plt.show()
63
64  plt.figure(figsize=(10,1))
65  for i in range(filters):
66      plt.subplot(1, filters, i+1)
67      ax = plt.gca()
68      ax.get_xaxis().set_visible(False)
69      ax.get_yaxis().set_visible(False)
70      plt.imshow(C[i])
71  plt.suptitle('Convolution result', fontsize=12)
72  plt.show()
73
```

Section 03 하이퍼파라미터

❑ padding(풀링층)

- ex11_12.ipynb

```
74  plt.figure(figsize=(10,1))
75  for i in range(filters):
76      plt.subplot(1, filters, i+1)
77      ax = plt.gca()
78      ax.get_xaxis().set_visible(False)
79      ax.get_yaxis().set_visible(False)
80      plt.imshow(P1[i])
81  plt.suptitle('Pooling result', fontsize=12)
82  plt.show()
83
84  plt.figure(figsize=(10,1))
85  for i in range(filters):
86      plt.subplot(1, filters, i+1)
87      ax = plt.gca()
88      ax.get_xaxis().set_visible(False)
89      ax.get_yaxis().set_visible(False)
90      plt.imshow(P2[i])
91  plt.suptitle('Pooling result', fontsize=12)
92  plt.show()
```



Section 03 하이퍼파라미터

❑ 우편번호 인식기 v2

- 앞 장에서 만든 우편번호 인식기 v1은 성능이 꽤 실망스러웠다.
- 컨볼루션 신경망으로 필기 숫자에 대한 정확률을 개선하여 버전 업그레이드를 시도해보자.
- 예제11-13의 목적은 우편번호 인식기 v1을 업그레이드하기 위해 높은 정확률의 필기 숫자 인식기를 확보하는 것이다.
- 우편번호 인식기 v1의 신경망 구조와 학습에 관련된 하이퍼파라미터를 그대로 사용했다.
- LeNet-5보다 층이 더 깊고 Dropout 층이 추가되었다.
- 34행은 학습한 모델을 'cnn_v2.h5' 파일에 저장하여 우편번호 인식기 v1을 업그레이드하는 데 쓸 수 있게 준비한다.

Section 03 하이퍼파라미터

❑ 우편번호 인식기 v2

- ex11_13.ipynb

```
1   import tensorflow as tf
2   import numpy as np
3   import matplotlib.pyplot as plt
4
5   from tensorflow.keras.datasets import mnist
6   from tensorflow.keras.models import Sequential
7   from tensorflow.keras.layers import Conv2D,MaxPooling2D,Flatten,Dropout,Dense
8   from tensorflow.keras.utils import to_categorical
9
10  (x_train, y_train), (x_test, y_test) = mnist.load_data()
11  x_train=x_train.reshape(60000,28,28,1)
12  x_test=x_test.reshape(10000,28,28,1)
13  x_train = x_train.astype(np.float32) / 255.0
14  x_test = x_test.astype(np.float32) / 255.0
15  y_train = to_categorical(y_train, 10)
16  y_test = to_categorical(y_test, 10)
17
```

Section 03 하이퍼파라미터

```
18  cnn=Sequential()
19  cnn.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28,28,1)))
20  cnn.add(Conv2D(32, (3, 3), activation='relu'))
21  cnn.add(MaxPooling2D(pool_size=(2, 2)))
22  cnn.add(Dropout(0.25))
23  cnn.add(Conv2D(64, (3, 3), activation='relu'))
24  cnn.add(Conv2D(64, (3, 3), activation='relu'))
25  cnn.add(MaxPooling2D(pool_size=(2, 2)))
26  cnn.add(Dropout(0.25))
27  cnn.add(Flatten())
28  cnn.add(Dense(units=512, activation='relu'))
29  cnn.add(Dropout(0.5))
30  cnn.add(Dense(units=10, activation='softmax'))
31
32  cnn.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['acc'])
33  history=cnn.fit(x_train, y_train, batch_size=128, epochs=100, validation_data=(x_test, y_test), verbose=2)
34
35  cnn.save('cnn_v2.h5')
36
37  res=cnn.evaluate(x_test, y_test, verbose=0)
38
39  print('정확도=',res[1]*100)
```

Section 03 하이퍼파라미터

❑ 우편번호 인식기 v2

- **생성한 딥러닝 모델 파일인** cnn_v2.h5**파일을 라즈베리 파이의** /home/pi/Smart_Car2/data **폴더로 이동시킨다**.
- 10**장의 우편번호 인식기** v.1**을 컨볼루션 신경망 버전으로 업그레이드하려면** 07**행과** 37**행만** ex11_14.py**처럼 수정하면 된다**.
- 6**행은** ex11_13.py**의** 35**행이 저장해둔 모델 파일인 'cnn_v2.h5'를 읽어온다**.
- 35**행은 샘플의 텐서를 컨볼루션 신경망의 입력층에 맞추는 일을 한다**.

Section 03 하이퍼파라미터

❑ 우편번호 인식기 v2

- ex11_14.ipynb

```
1   import tensorflow as tf
2   import numpy as np
3   import matplotlib.pyplot as plt
4   import cv2
5
6   model = tf.keras.model.load_model('./data/cnn_v2.h5')
7
8   def reset():
9     global img
10    img = np.ones((200, 520, 3), dtype=np.uint8) * 255
11    for i in range(5):
12      cv2.rectangle(img, (10 + i * 100, 50), (10 + (i + 1) * 100, 150), (0, 0, 255))
13    cv2.putText(img, 'e:erase s:show r:recognition q:quit', (10, 40), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255,
    0, 0), 1)
14
```

Section 03 하이퍼파라미터

❑ 우편번호 인식기 v2

- ex11_14.ipynb

```
15  def grab_numerals():
16      numerals = []
17      for i in range(5):
18          roi = img[51:149, 11 + i * 100:9 + (i + 1) * 100, 0]
19          roi = 255 - cv2.resize(roi, (28, 28), interpolation = cv2.INTER_CUBIC)
20          numerals.append(roi)
21      numerals = np.array(numerals)
22      return numerals
23
24  def show():
25      numerals = grab_numerals()
26      plt.figure(figsize = (25, 5))
27      for i in range(5):
28          plt.subplot(1, 5, i + 1)
29          plt.imshow(numerals[i], cmap='gray')
30          plt.xticks([]); plt.yticks([])
31      plt.show()
32
```

Section 03 하이퍼파라미터

❑ 우편번호 인식기 v2

- ex11_14.ipynb

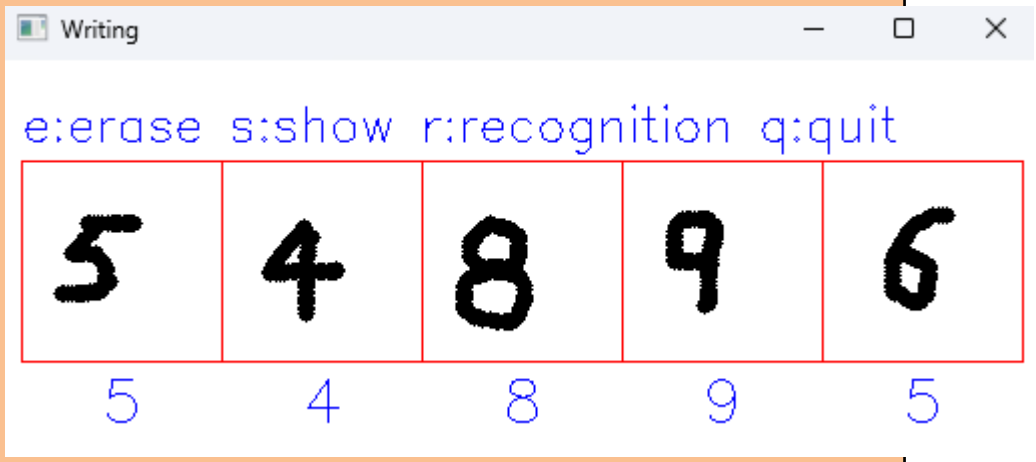
```
33  def recognition():
34      numerals = grab_numerals()
35      numerals = numerals.reshape(5, 28, 28, 1)
36      epochs = 10
37      numerals = numerals.astype(np.float32) / 255.0
38      pred = model.predict(numerals)
39      class_id = np.argmax(pred, axis=1)
40      for i in range(5):
41          cv2.putText(img, str(class_id[i]), (50 + i * 100, 180), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 0, 0), 1)
42
43  BrushSize = 4
44  LColor = (0, 0, 0)
45
46  def writing(event, x, y, flags, param):
47      if event == cv2.EVENT_LBUTTONDOWN:
48          cv2.circle(img, (x, y), BrushSize, LColor, -1)
49      elif event == cv2.EVENT_MOUSEMOVE and flags == cv2.EVENT_FLAG_LBUTTON:
50          cv2.circle(img, (x, y), BrushSize, LColor, -1)
51
```

Section 03 하이퍼파라미터

❑ 우편번호 인식기 v2

- ex11_14.ipynb

```
52  reset()
53  cv2.namedWindow("Writing")
54  cv2.setMouseCallback("Writing", writing)
55  while(True):
56      cv2.imshow('Writing', img)
57      key = cv2.waitKey(1)
58      if key == ord('e'):
59          reset()
60      elif key == ord('s'):
61          show()
62      elif key == ord('r'):
63          recognition()
64      elif key == ord('q'):
65          break
66
67  cv2.destroyAllWindows()
```



Section 03 하이퍼파라미터

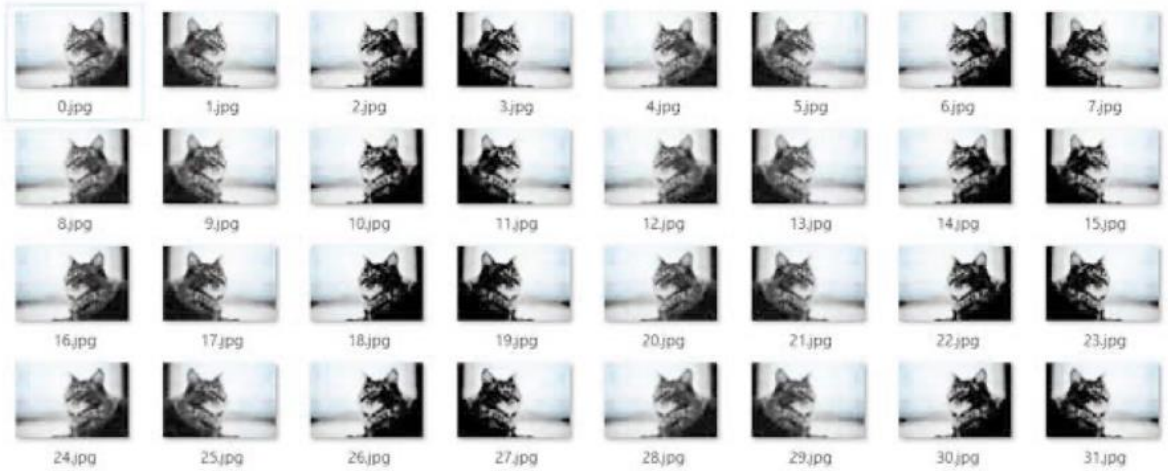
❑ 우편번호 인식기 v2

- 프로그램을 실행해보면, 버전 2가 버전 1보다 월등히 잘 인식하는 것을 확인할 수 있다.
- 우편번호 인식기는 미국인이 종이에 쓴 필기체 숫자 데이터셋 MNIST로 학습하였는데, 한국인이 전자펜으로 화면에 쓴 필기 숫자 데이터셋을 수집하여 다시 학습하면 정확도를 더 높일 수 있다.
- 이 사례에서 볼 수 있는 것처럼 학습에 사용한 데이터와 실제 현장에서 발생한 데이터의 분포가 다른 상황을 데이터셋 시프트(dataset shift)라 부른다.
- 데이터셋 시프트가 발생하면 성능 저하가 나타난다.
- 데이터셋 시프트 문제 해결을 포함하여 여러 방법으로 성능을 높인 다음, 이 시스템을 우체국에 설치하면 근사한 우편번호 인식기가 되어 우체국의 업무 효율을 크게 높일 것이다.

Section 04 데이터 부풀리기

❑ ImageDataGenerator

- 이미지 인식에는 이미지 데이터 및 그에 대응하는 라벨(지도 데이터)의 조합이 대량으로 필요하다.
- 그러나 충분한 수의 이미지와 라벨의 조합을 준비하는 것은 다양한 비용적 측면에서 어려울 수 있다.
- 따라서 데이터 수를 충분한 양으로 늘릴 때 이미지 부풀리기를 활용할 수 있다.
- 이미지 부풀리기가 단순히 복사해서 양을 늘리는 것이라면 큰 의미가 없다.
- 그러므로 이미지를 뒤집거나 좌우로 조금씩 이동시켜 새로운 데이터를 만들어낸다.



Section 04 데이터 부풀리기

❑ ImageDataGenerator

- **여기서는 텐서플로우의** ImageDataGenerator**를 사용하여 패딩을 실시한다.** ImageDataGenerator**에는 여러 인수가 있어서 다양한 방법으로 데이터를 쉽게 가공할 수 있다. 또한 복수의 가공법을 조합하여 새로운 이미지를 생성할 수 있다.**

```
datagen = ImageDataGenerator(rotation_range=0.,
                             width_shift_range=0.,
                             height_shift_range=0.,
                             shear_range=0.,
                             zoom_range=0.,
                             channel_shift_range=0z
                             horizontal_flip=False,
                             vertical_flip=False)
```

Section 04 데이터 부풀리기

❑ ImageDataGenerator

- **다음은** ImageDataGenerator**에서 널리 쓰이는 옵션이다.**
  - rotation_range: **회전하는 범위**(**단위**: degree)
  - width_shift_range: **수평 이동하는 범위**(**이미지의 가로폭에 대한 비율**)
  - height_shif t_range: **수직 이동하는 범위**(**이미지의 세로폭에 대한 비율**)
  - shear_range: **전단**(shearing) **범위. 크게 하면 더 비스듬하게 찌그러진 이미지가 됨**(**단위**: degree)
  - zoom_range: **이미지를 확대/축소시키는 비율**(**최소**: 1 - zoom_range, **최대**: 1 + zoom_range)
  - channel_shift_range: **입력이** RGB 3 **채널인 이미지의 경우** R, G, B **각각에 임의의 값을 더하거나 뺄 수 있음**(0~255)
  - horizontal_flip : True **로 설정 시 가로로 반전**
  - vertical_flip : True **로 설정 시 세로로 반전**

Section 05 정규화

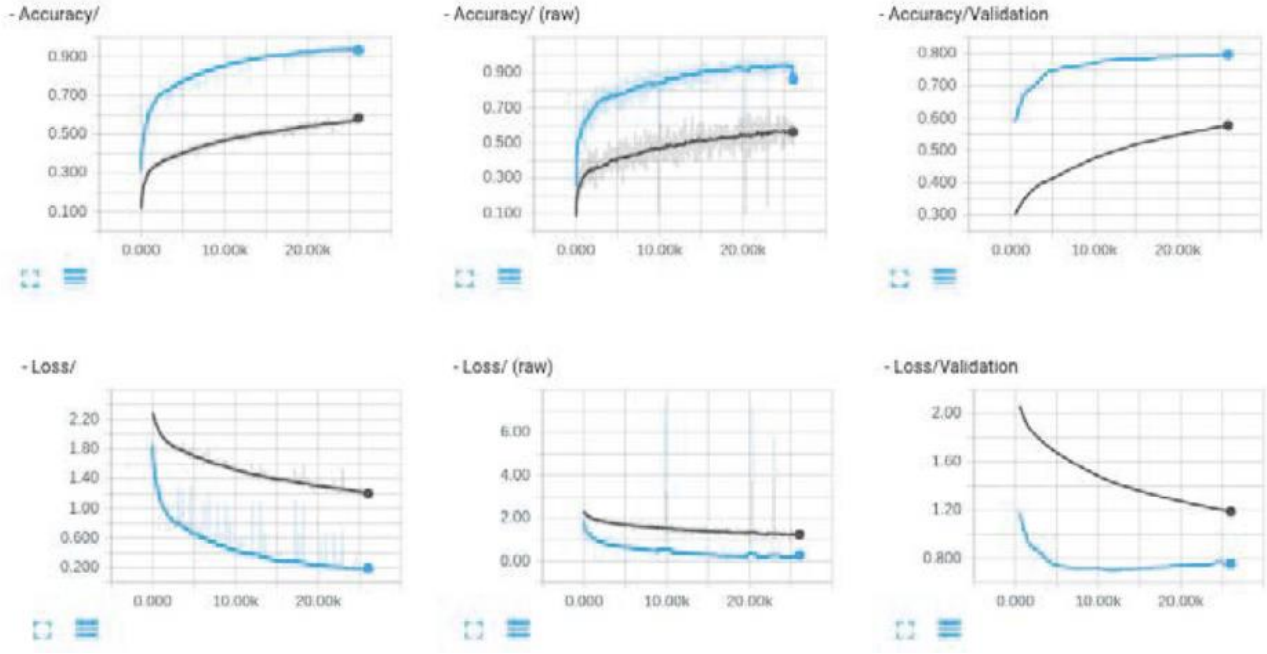
❑ 다양한 정규화 방법

- 다음 그림은 정규화의 예이다.
- 규칙에 따라 데이터를 처리하고, 사용하기 쉽게 하는 것을 정규화라고 한다.
- 그림에서는 정규화로 햇볕이 드는 방법을 통일하여 학습에 직접적인 관계가 없는 데이터 간의 차이를 제거하고 있다.
- 이렇게 하면 학습 효율을 크게 높일 수 있다.

Section 05 정규화

❑ 다양한 정규화 방법

- **다음 그림의 그래프는** cifar10 **분류에 배치 정규화**(batch normalization, BN)**를 수행하여 정확도가 크게 상승한 것을 보여준다**.
- **파란색 선이** BN **없음**, **검은색 선이** BN **있음이다**.
- **최근에는 심층 신경망 모델에서 정규화가 별로 필요 없다고 여겨 지기도 하지만 간단한 모델을 사용하는 경우에는 매우 유용하다**.



Section 05 정규화

❑ 다양한 정규화 방법

- **딥러닝의 정규화방법은 여러 가지가 있으며, 대표적인 것은 다음과 같다.**
  - **배치 정규화**(Batch Normalization, BN)
  - **주성분 분석**(Principal Component Analysis, PCA)
  - **특이값 분해**(Singular Value Decomposition, SVD)
  - **제로위상 성분분석**(Zero—phase Component Analysis, ZCA)
  - **국소 응답 정규화**(Local Response Normalization, LRN)
  - **전역 콘트라스트 정규화**(Global Contrast Normalization, GCN)
  - **국소 콘트라스트 정규화**(Local Contrast Normalization, LCN)

- **이들 정규화 방법은 크게 표준화**(standardization)**와 백색화**(whitening)**로 나눌 수 있다.**

Section 05 정규화

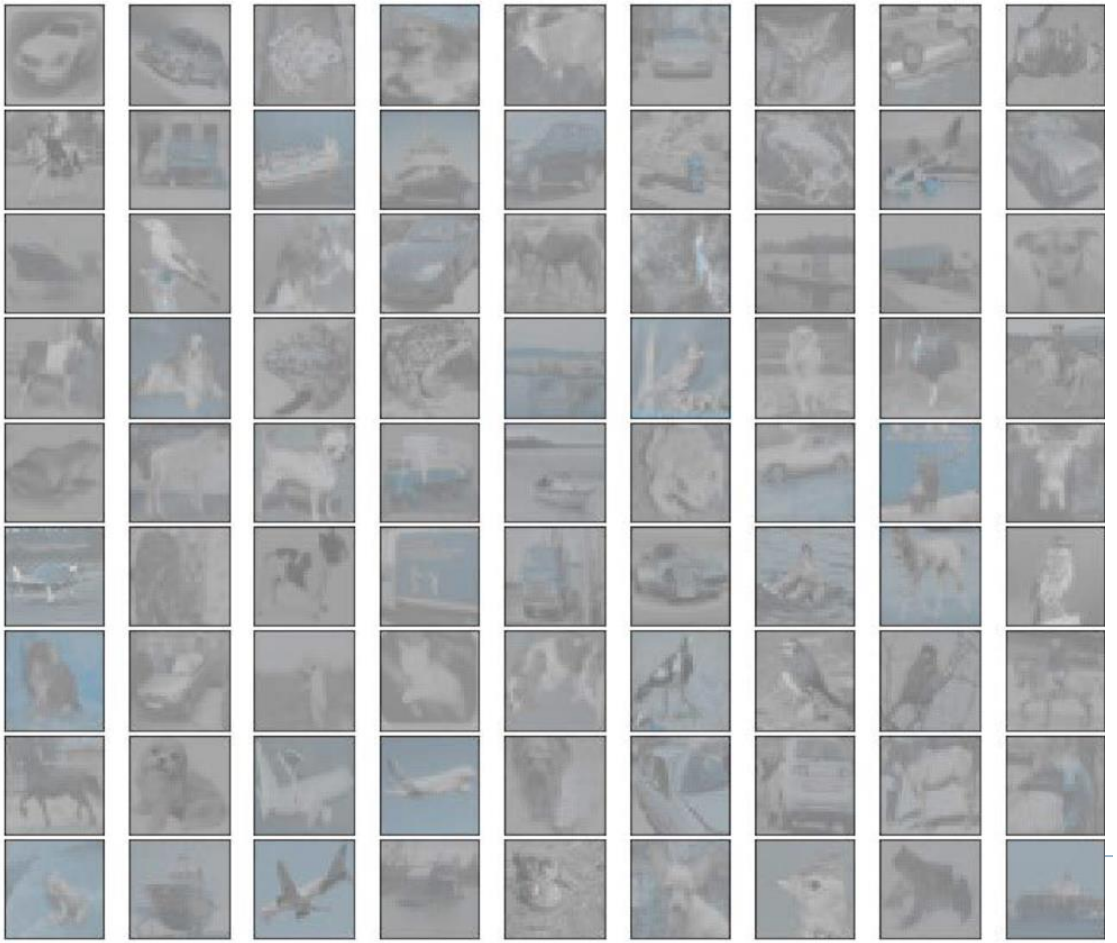
❑ 표준화

- **표준화**(standardization)**는 개별 특징에 대해 평균을** 0**으로,** **분산을** 1**로 하여 특징별 데이터 분포를 좁히는 방법이다.**
- **표준화를 수행하면 색조가 평균적이 되어 회색처럼 보이지만, 반대로 지금까지 눈에 띄지 않았던 색**(R, G, B **중 하나)이 다른 색상과 동일한 수준에서 중요시**(가중치가 붙음)**되므로 숨은 특징을 찾기 쉬워진다.**

Section 05 정규화

❑ 표준화

- **다음 그림은** cifar10 **데이터셋의 각 특징(여기서는**R, G, B**의** 3**채널)에 표준화를 실시한 것이다**.



Section 05 정규화

❑ 표준화

- ex11_15.ipynb

```
1  import matplotlib.pyplot as plt
2  from tensorflow.keras.datasets import cifar10
3  from tensorflow.keras.preprocessing.image import ImageDataGenerator
4
5  (X_train, y_train), (X_test, y_test) = cifar10.load_data()
6
7  for i in range(10):
8     plt.subplot(2, 5, i + 1)
9     plt.imshow(X_train[i])
10 plt.suptitle('The original image', fontsize=12)
11 plt.show()
12
13 datagen = ImageDataGenerator( samplewise_center=True, samplewise_std_normalization=True)
14 g = datagen.flow(X_train, y_train, shuffle=False)
15 X_batch, y_batch = g.next()
16
```

Section 05 정규화

❑ 표준화

- ex11_15.ipynb

```
17  X_batch *= 127.0 / max(abs(X_batch.min()), X_batch.max())
18  X_batch += 127.0
19  X_batch = X_batch.astype('uint8')
20
21  for i in range(10):
22      plt.subplot(2, 5, i + 1)
23      plt.imshow(X_batch[i])
24  plt.suptitle('Standardization result', fontsize=12)
25  plt.show()
```

Section 05 정규화

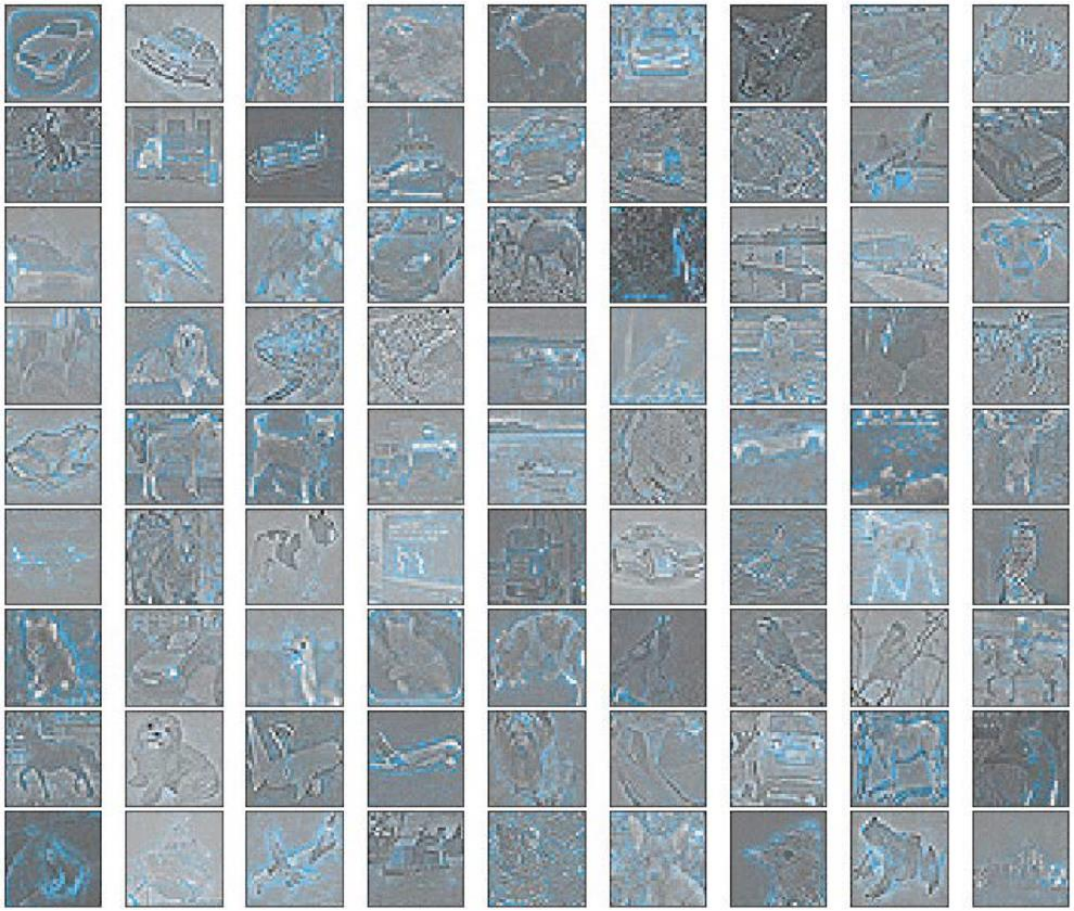
❑ 백색화

- **백색화**(whitening)**는 데이터 성분 사이의 상관관계를 없애는 방법이다.**
- **백색화를 수행하면 전체적으로 어두워지고 가장자리가 강조된 것처럼 보이지만 이는 백색화가 주위의 픽셀 정보로부터 쉽게 상정되는 색상은 무시하는 효과가 있기 때문이다.**
- **백색화로 정보량이 적은 면이나 배경 등이 아니라 정보량이 많은 가장자리 등을 강조함으로써 학습 효율을 높일 수 있다.**

Section 05 정규화

❑ 백색화

- **다음 그림은** cifar10 **데이터셋의 각 특징(여기서는** R, G, B**의** 3**채널)에 백색화를 실시한 것이다**.



Section 05 정규화

❑ 백색화

- ex11_16.ipynb

```
1  import matplotlib.pyplot as plt
2  from tensorflow.keras.datasets import cifar10
3  from tensorflow.keras.preprocessing.image import ImageDataGenerator
4
5  (X_train, y_train), (X_test, y_test) = cifar10.load_data()
6
7  X_train = X_train[:300]
8  X_test = X_test[:100]
9  y_train = y_train[:300]
10 y_test = y_test[:100]
11
12 for i in range(10):
13     plt.subplot(2, 5, i + 1)
14     plt.imshow(X_train[i])
15 plt.suptitle('The original image', fontsize=12)
16 plt.show()
17
```

Section 05 정규화

❑ 백색화

- ex11_16.ipynb

```
18  datagen = datagen = ImageDataGenerator(zca_whitening=True)
19
20  datagen.fit(X_train)
21  g = datagen.flow(X_train, y_train, shuffle=False)
22  X_batch, y_batch = g.next()
23
24  X_batch *= 127.0 / max(abs(X_batch.min()), abs(X_batch.max()))
25  X_batch += 127
26  X_batch = X_batch.astype('uint8')
27  for i in range(10):
28      plt.subplot(2, 5, i + 1)
29      plt.imshow(X_batch[i])
30  plt.suptitle('Whitening result', fontsize=12)
31  plt.show()
```

Section 05 정규화

❑ 배치 정규화

- 딥러닝에서 미니배치 학습을 통해 배치마다 표준화를 수행하는 것을 배치 정규화(batch normalization)라고 한다.
- Keras에서는 전결합층이나 합성곱층, 활성화 함수 등과 마찬가지로 다음과 같이 model의 add () 메서드로 모델에 배치 정규화를 통합할 수 있다.

```
model.add(BatchNormalizat ion())
```

- 배치 정규화는 데이터의 전처리뿐만 아니라 중간층의 출력에도 적용할 수 있다.
- 특히 활성화 함수 ReLU 등 출력 값의 범위가 한정되지 않은 함수의 출력에 배치 정규화를 사용하면 학습이 원활하게 진행되어 큰 효과를 발휘한다.

Section 05 정규화

❑ 배치 정규화

- ex11_17.ipynb

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  from tensorflow.keras.datasets import mnist
4  from tensorflow.keras.layers import Activation, Conv2D, Dense, Flatten, MaxPooling2D, BatchNormalization
5  from tensorflow.keras.models import Sequential
6  from tensorflow.keras.utils import to_categorical
7
8  (X_train, y_train), (X_test, y_test) = mnist.load_data()
9  X_train = np.reshape(a=X_train, newshape=(-1, 28, 28, 1))
10 X_test = np.reshape(a = X_test,newshape=(-1, 28, 28, 1))
11 y_train = to_categorical(y_train)
12 y_test = to_categorical(y_test)
13
14 model1 = Sequential()
15 model1.add(Conv2D(input_shape=(28, 28, 1), filters=32, kernel_size=(2, 2), strides=(1, 1), padding="same"))
16 model1.add(MaxPooling2D(pool_size=(2, 2)))
17 model1.add(Conv2D(filters=32, kernel_size=(2, 2), strides=(1, 1), padding="same"))
18 model1.add(MaxPooling2D(pool_size=(2, 2)))
19 model1.add(Flatten())
```

Section 05 정규화

❑ 배치 정규화

- ex11_17.ipynb

```
20  model1.add(Dense(256))
21  model1.add(Activation('sigmoid'))
22  model1.add(Dense(128))
23  model1.add(Activation('sigmoid'))
24  model1.add(Dense(10))
25  model1.add(Activation('softmax'))
26
27  model1.compile(optimizer='sgd', loss='categorical_crossentropy',  metrics=['acc'])
28
29  history = model1.fit(X_train, y_train, batch_size=32, epochs=3, validation_data=(X_test, y_test))
30
31  plt.plot(history.history['acc'], label='acc', ls='-', marker='o')
32  plt.plot(history.history['val_acc'], label='val_acc', ls='-',  marker='x')
33  plt.ylabel('accuracy')
34  plt.xlabel('epoch')
35  plt.suptitle('model1', fontsize=12)
36  plt.show()
37
```

Section 05 정규화

❑ 배치 정규화

- ex11_17.ipynb

```
38  model2 = Sequential()
39  model2.add(Conv2D(input_shape=(28, 28, 1), filters=32, kernel_size=(2, 2), strides=(1, 1), padding="same"))
40  model2.add(MaxPooling2D(pool_size=(2, 2)))
41  model2.add(Conv2D(filters=32, kernel_size=(2, 2), strides=(1, 1), padding="same"))
42  model2.add(MaxPooling2D(pool_size=(2, 2)))
43  model2.add(Flatten())
44  model2.add(Dense(256))
45  model2.add(Activation('relu'))
46  model2.add(BatchNormalization())
47  model2.add(Dense(128))
48  model2.add(Activation('relu'))
49  model2.add(BatchNormalization())
50  model2.add(Dense(10))
51  model2.add(Activation('softmax'))
52
53  model2.compile(optimizer='sgd', loss='categorical_crossentropy', metrics=['acc'])
54  history = model2.fit(X_train, y_train, batch_size=32, epochs=3, validation_data=(X_test, y_test))
```

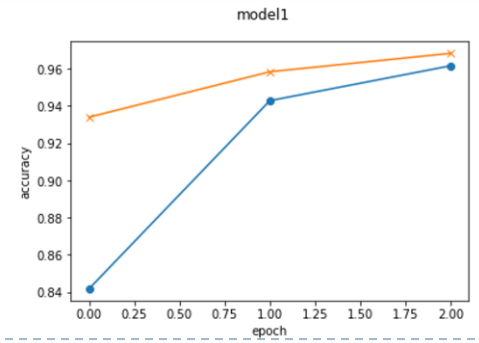
Section 05 정규화

❑ 배치 정규화

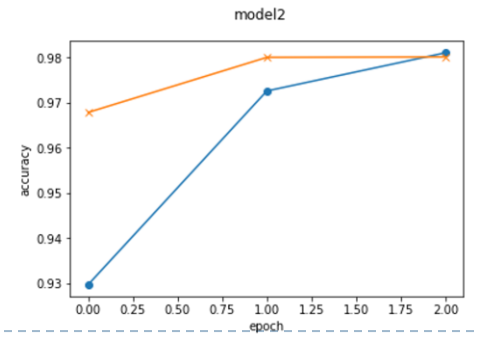
- ex11_17.ipynb

```
55
56  plt.plot(history.history['acc'], label='acc', ls='-', marker='o')
57  plt.plot(history.history['val_acc'], label='val_acc', ls='-', marker='x')
58  plt.ylabel('accuracy')
59  plt.xlabel('epoch')
60  plt.suptitle('model2', fontsize=12)
61  plt.show()
```

```
Epoch 1/3
1875/1875 [==============================] - 10s 5ms/step - loss: 0.8544 - acc: 0.8418 - val_loss: 0.3559 - val_acc: 0.
Epoch 2/3
1875/1875 [==============================] - 9s 5ms/step - loss: 0.2635 - acc: 0.9428 - val_loss: 0.1862 - val_acc: 0.9
Epoch 3/3
1875/1875 [==============================] - 8s 4ms/step - loss: 0.1613 - acc: 0.9616 - val_loss: 0.1253 - val_acc: 0.9
```




```
Epoch 1/3
1875/1875 [==============================] - 11s 6ms/step - loss: 0.2390 - acc: 0.9296 - val_loss: 0.1139 - val_acc: 0.
Epoch 2/3
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0965 - acc: 0.9725 - val_loss: 0.0864 - val_acc: 0.
Epoch 3/3
1875/1875 [==============================] - 10s 5ms/step - loss: 0.0656 - acc: 0.9810 - val_loss: 0.0725 - val_acc: 0.
```

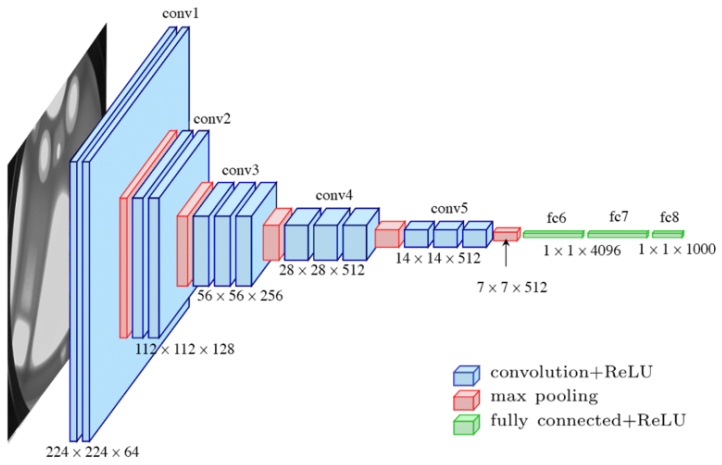


Section 06 전이학습

- 대규모 신경망을 학습시키는 데는 매우 많은 시간이 걸리며, 데이터도 대량으로 필요하다.
- 이런 경우 대량의 데이터로 미리 학습되어 공개된 모델을 이용하는 것이 효과적이다.
- 학습된 모델을 이용하여 새로운 모델을 학습시키는 것을 전이학습(Transfer learning)이라고 한다.
- 텐서플로우에서는 ImageNet(120만장, 1000클래스로 이루어진 거대한 이미지 데이터셋)으로 학습한 이미지 분류 모델과 그 가중치를 다운로드하여 사용할 수 있다.
- 공개된 모델은 여러 가지가 있지만 여기서는 VGG16 모델을 예로 설명한다.

Section 06 전이학습

- VGG 모델은 옥스퍼드 대학의 VGG(Visual Geometry Group) 팀이 만든 네트워크 모델로, 2014년에 열린 ILSVRC(ImageNet Large Scale Visual Recognition Challenge)라는 대규모 이미지 인식 대회에서 준우승을 차지했다.
- 작은 필터를 사용한 합성곱을 2~4회 연속으로 실시하여 풀링을 반복하는 당시로서는 상당히 층을 깊게 한 것이 특징이다.
- VGG 모델은 가중치를 가진 층 (합성곱층과 전결합층)을 16층 거듭한 것과 19층 거듭한 것이 있으며, 각각 VGG16, VGG19라고 부른다.
- VGG16은 합성곱 13층 + 전결합층 3층 = 16층의 신경망으로 되어 있다.



Section 06 전이학습

- 원래 VGG 모델은 1,000클래스의 분류 모델이므로 출력 유닛이 1,000개 있지만, 마지막 전결합층을 제외한 도중까지의 층을 특징 추출용 층으로 사용하여 전이학습에 활용할수 있다.
- 또한 입력 이미지의 크기를 신경 쓸 필요가 없다.
- VGG16 모델은 합성곱층 커널의 크기가 3 X 3으로 작고, 'padding = same'으로 되어 있으며, 극단적으로 입력 이미지가 적지 않은한 13층을 거쳐 추출되는 특징 수가 일정 부분 확보되기 때문이다.

Section 06 전이학습

❑ VGG16

- **텐서플로우로** cifar10 **데이터셋을 전이학습을 이용하여 분류해보자**.
- **지금까지 사용해온** Sequential **유형의 모델에** VGG16 **모델을 조합하겠다**.
- **먼저** VGG **모델을 만든다**.

```
from keras.applications.vgg16 import VGG16
input_tensor = Input(shape=(32, 32, 3))
vgg16 = VGG16(include_top=False, weights='imagenet', input_tensor=input, tensor)
```

- input_tersor**로 입력 형태를 부여한다**.
- include_top**은 원래 모델의 최후 전결합층을 사용할지 여부이다**.
- **이를** False**로 하면 원래 모델의 합성곱층의 특징 추출부분만 사용하여 이후 층에는 스스로 작성한 모델을 추가할 수 있다**.
- weights**에** imagenet**을 지정하면** ImageNet**에서 학습한 가중치를 사용하고**, None**을 지정하면 임의의 가중치를 사용하게 된다**.

Section 06 전이학습

❑ VGG16

- **특징 추출 부분 이후에 새로운 다른 층을 추가하려면 미리 VGG와 다른 모델 (여기서는 top_model)을 정의하고 다음과 같이 결합한다.**

```
top_model = vgg16.output
top_model = Flatten(input_shape=vgg16.output_shape[l : ])(top_model)
top_model = Dense(256, activation=1 sigmoid')(top_model)
top_model = Dropout(0.5)(top_model)
top_model = Dense(10, activation='softmax')(top_model)
model = Model(inputs=vgg16.input, outputs=top_model)
```

- **VGG16에 의한 특징 추출부분의 가중치는 갱신되면 흐트러져버리므로 다음 코드처럼 고정시킨다.**

```
for layer in model.layers[ : 19] :
layer.trainable = False
```

- **컴파일과 학습은 동일하게 이뤄지지만 전이학습의 경우 최적화에 SGD를 선택하는 것이 좋다고 알려져 있다.**

```
model.compile(loss='categorical_crossentropy',
optimizer=optimizers.SGD(lr=1e-4, momentum=0.9),
metrics=[‘acc'])
```

Section 06 전이학습

❑ 배치 정규화

- **다음 예제는 cifar10 분류 모델을 VGG16을 사용하여 생성하고 전이학습하는 코드이다.**
- ex11_18.ipynb

```
1   from tensorflow.keras import optimizers
2   from tensorflow.keras.applications.vgg16 import VGG16
3   from tensorflow.keras.datasets import cifar10
4   from tensorflow.keras.layers import Dense, Dropout, Flatten, Input
5   from tensorflow.keras.models import Model, Sequential
6   from tensorflow.keras.utils import to_categorical
7   import matplotlib.pyplot as plt
8   import numpy as np
9
10  (X_train, y_train), (X_test, y_test) = cifar10.load_data()
11  y_train = to_categorical(y_train)
12  y_test = to_categorical(y_test)
13
14  input_tensor = Input(shape=(32, 32, 3))
15  vgg16 = VGG16(include_top=False, weights='imagenet', input_tensor=input_tensor)
16
```

Section 06 전이학습

❑ 배치 정규화

- ex11_18.ipynb

```
17  top_model = vgg16.output
18  top_model = Flatten(input_shape=vgg16.output_shape[1:])(top_model)
19  top_model = Dense(256, activation='sigmoid')(top_model)
20  top_model = Dropout(0.5)(top_model)
21  top_model = Dense(10, activation='softmax')(top_model)
22
23  model = Model(inputs=vgg16.input, outputs=top_model)
24
25  for layer in model.layers[:19]:
26     layer.trainable = False
27
28  model.summary()
29
30  model.compile(loss='categorical_crossentropy',
                  optimizer=optimizers.SGD(lr=1e-4, momentum=0.9),
                  metrics=['acc'])
31  model.fit(X_train, y_train, validation_data=(X_test, y_test), batch_size=32, epochs=3)
32
```

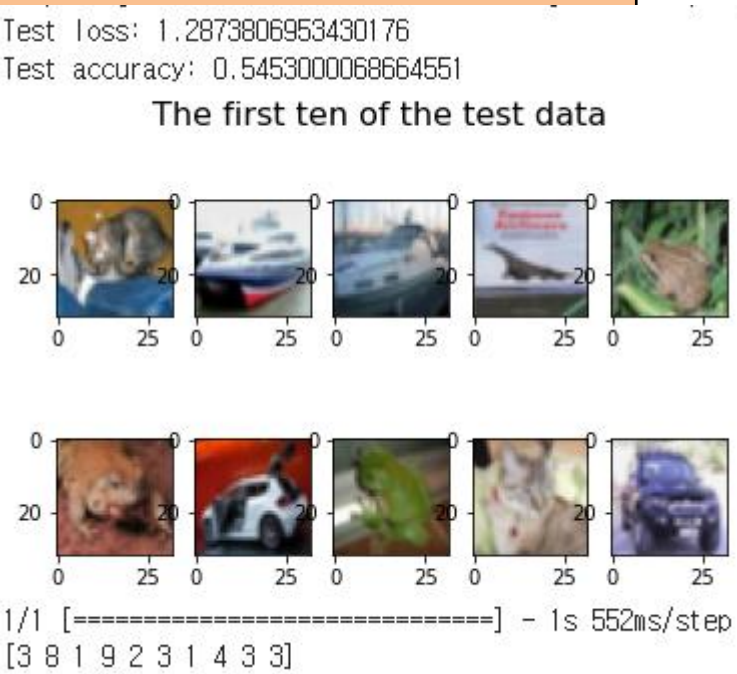
Section 06 전이학습

❑ 배치 정규화

- ex11_18.ipynb

```
33  model.save_weights('param_vgg.hdf5')
34
35  scores = model.evaluate(X_test, y_test, verbose=1)
36  print('Test loss:', scores[0])
37  print('Test accuracy:', scores[1])
38
39  for i in range(10):
40      plt.subplot(2, 5, i+1)
41      plt.imshow(X_test[i])
42  plt.suptitle("The first ten of the test data",fontsize=16)
43  plt.show()
44
45  pred = np.argmax(model.predict(X_test[0:10]), axis=1)
46  print(pred)
```

```
Test loss: 1.2873806953430176
Test accuracy: 0.5453000068664551
```



Section 07 견종 인식 프로그램

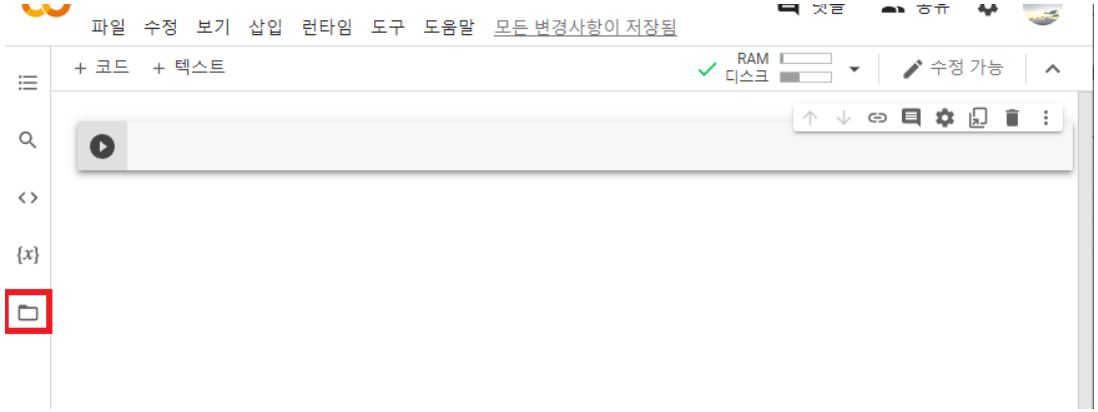
❑ Stanford dogs 데이터셋 설치

- **예제** 11-19**는 사전 학습된** DenseNet121 **모델을** Stanford dogs **데이터셋으로 전이 학습한다**.
- Stanford dogs**는 같은 부류에 속하는 영상이 심하게 달라 부류 내 변화**(intra-class variation)**가 크고 다른 부류 영상이 비슷한 경우가 많아 부류 간 변화**(inter-class variation)**가 작은 대표적인 미세분류**(fine-grained classification) **문제다**.
- **이 데이터셋에는 부류**(**개의 품종**)**가** 120**개**, **영상이** 20,580**장 있다**.
- **텐서플로에서 기본으로 제공하지 않기 때문에 웹사이트에 접속해 다운로드해야 한다**.
- http://vision.stanford.edu/aditya86/ImageNetDogs **에 접속해** images.tar, annotations.tar, list.tar **파일을 다운로드한다**.

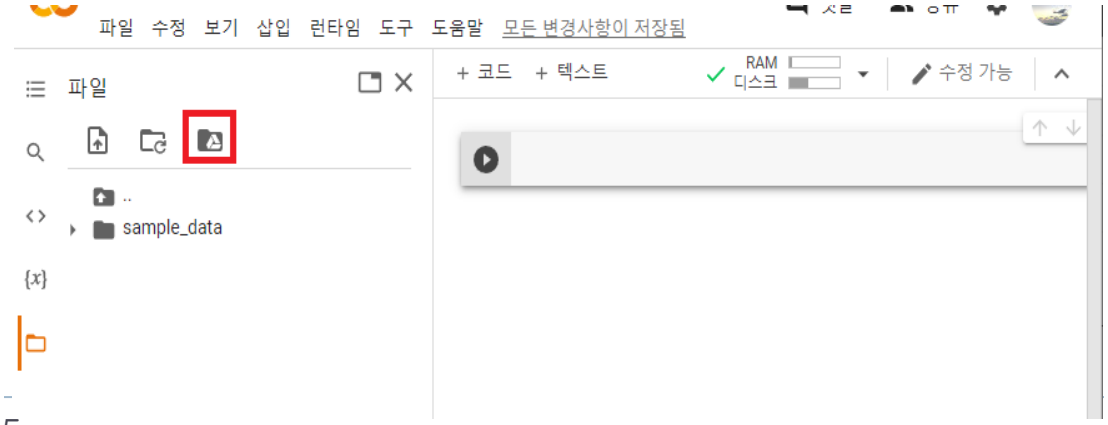
Section 07 견종 인식 프로그램

❑ 구글 드라이브 마운트

- 구글 코랩 노트에 왼쪽 중간의 '파일'버튼을 클릭한다.



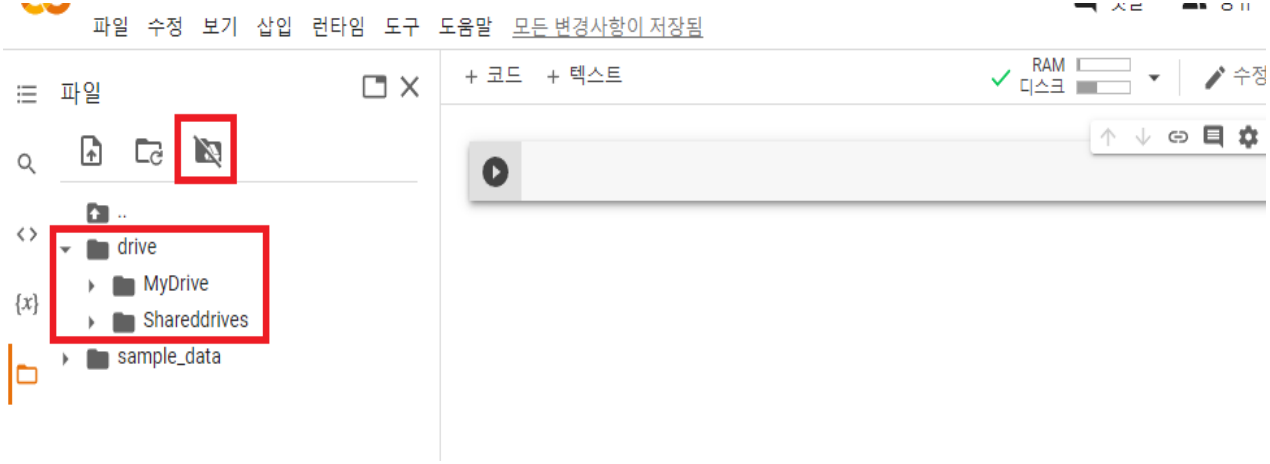
- 아래와 같은 화면이 나타나면 표시의 '드라이브 마운트' 아이콘을 클릭한다.



Section 07 견종 인식 프로그램

❑ 구글 드라이브 마운트

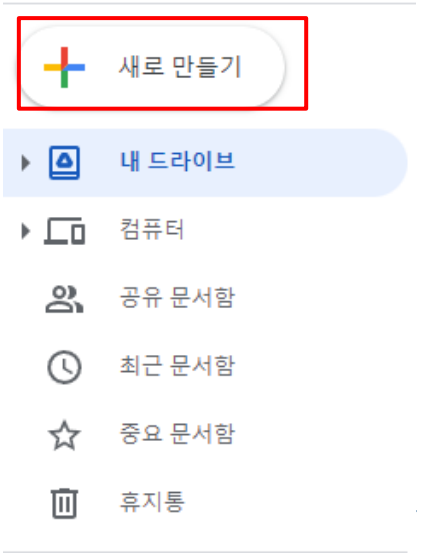
- '드라이브 마운트' 아이콘에 대각선이 생기며 구글 드라이브 연동이 완료된다.



Section 07 견종 인식 프로그램

❑ 구글 드라이브 마운트

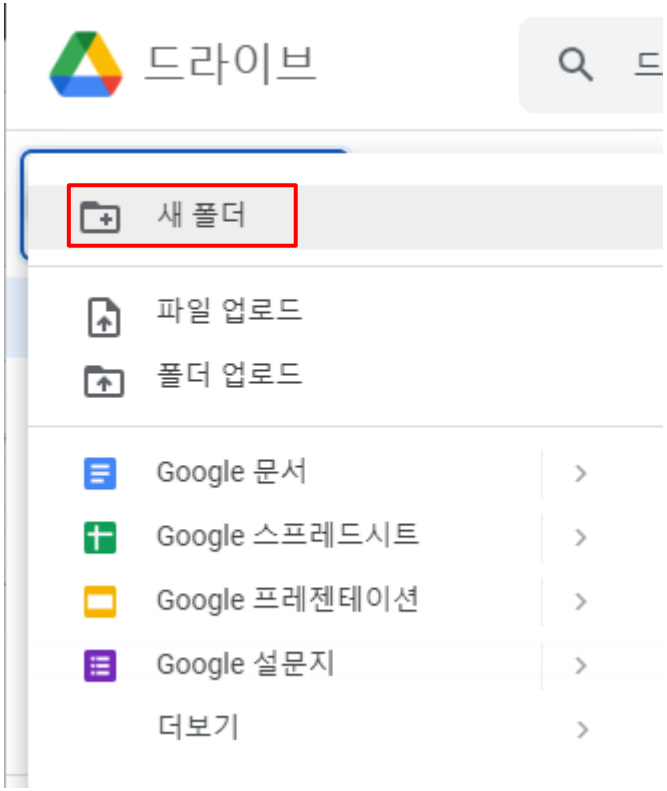
- **크롬 브라우저에서 새로운 탭을 열어** https://drive.google.com **에 접속 후 로그인한다.**
- **코랩과 같은 계정을 사용한다면 코랩에서 사용한 파일이 저장된** [Colab Notebooks] **폴더가 생성되어 있다.**
- [Colab Notebooks] **폴더는 코랩에서 사용한 코드가 저장된 폴더이다.**
- [**새로 만들기**] **버튼을 클릭한다.**



Section 07 견종 인식 프로그램

❑ 구글 드라이브 마운트

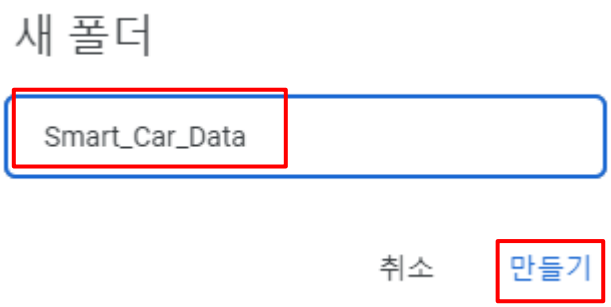
- [새폴더] 버튼을 클릭한다.



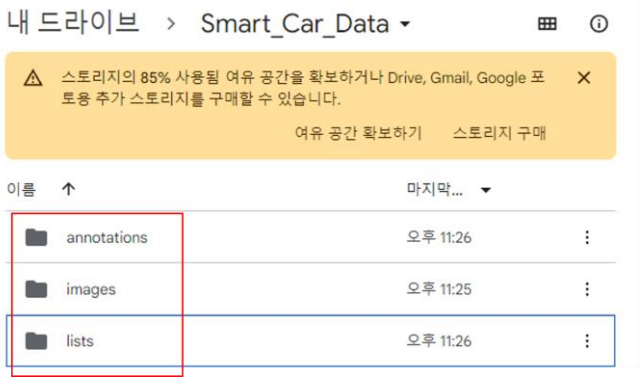
Section 07 견종 인식 프로그램

❑ 구글 드라이브 마운트

- [Smart_Car_Data]**의 이름으로 폴더를 만든다**.



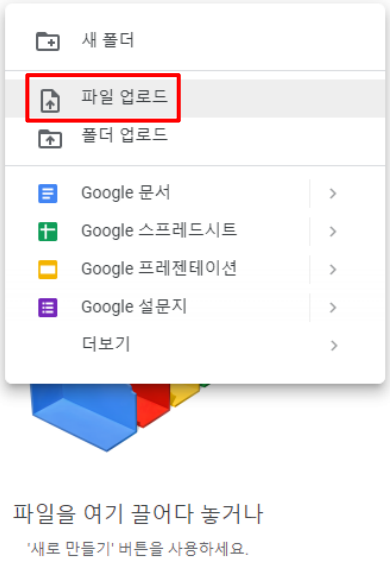
- [Smart_Car_Data] **폴더가 생성되었다**.
- [Smart_Car_Data] **폴더를 더블클릭하여 접속한다**.
- **여기서** images, annotations, lists **폴더를 새로 생성한다**.

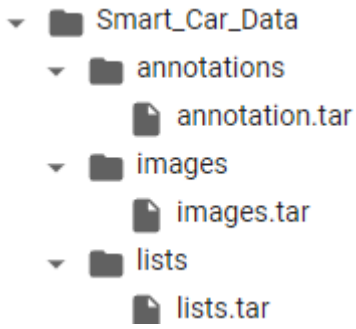


Section 07 견종 인식 프로그램

❑ 구글 드라이브 마운트

- **파일을 끌어다 놓거나 마우스 오른쪽을 눌러 [파일 업로드]를 눌러 파일을 업로드 할 수 있다.**





- **학습 데이터인** images.tar, annotations.tar, list.tar**파일을 각각** images**폴더**, annotations **폴더**, lists **폴더에 업로드한다.**

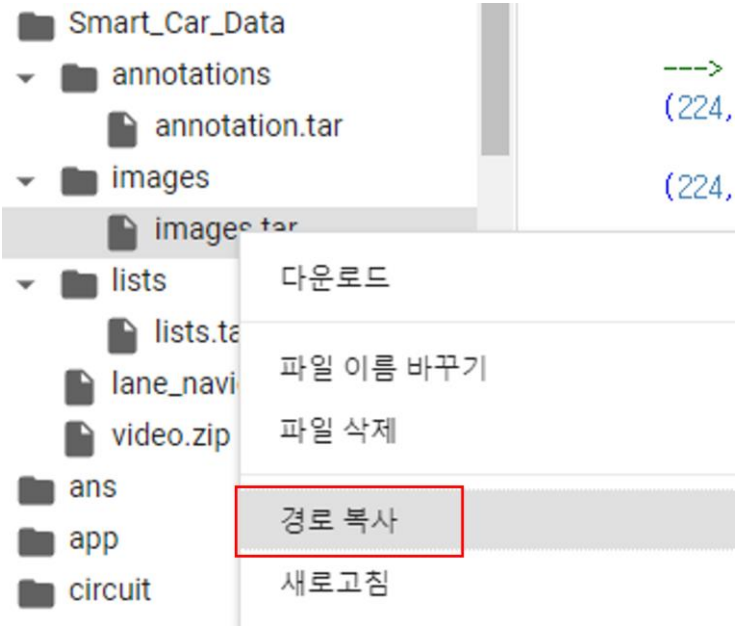
Section 07 견종 인식 프로그램

❑ 압축 풀기

- **구글 코랩 셀에 다음과 같이 입력하여 경로에 있는 파일의 압축을 풀어준다.**

```
!tar -xvf "/content/drive/MyDrive/Smart_Car_Data/images/images.tar"
```

- images.tar **파일의 경로를 복사하는 방법은 코랩에서** images.tar **파일에 마우스 오른쪽 버튼을 클릭 후 [경로 복사] 버튼을 클릭하면 된다.**



Section 07 견종 인식 프로그램

❑ 압축 풀기

- [shift + enter]**키를 입력하여 구글 코랩 셀을 실행한다**.
- [**폴더 새로고침**] **아이콘을 클릭하면 압축이 풀린** [Images] **폴더가 보여진다**.
- [Images] **폴더 안에는 매우 많은 사진 데이터가 있다**.
- **나머지 압축파일도 동일한 방식으로 해제한다**.

```
!tar -xvf '/content/drive/MyDrive/Smart_Car_Data/annotations/annotation.tar'
```

Section 07 견종 인식 프로그램

❑ 미세 조정 방식의 전이 학습: 견종 인식하기

- **예제** 11-19**는 사전 학습된** DenseNet121 **모델을** Stanford dogs **데이터셋으로 전이 학습한다**.
- ex11_19.ipynb

```
1   from tensorflow.keras.models import Sequential
2   from tensorflow.keras.layers import Flatten,Dense,Dropout,Rescaling
3   from tensorflow.keras.optimizers import Adam
4   from tensorflow.keras.applications.densenet import DenseNet121
5   from tensorflow.keras.utils import image_dataset_from_directory
6   import pathlib
7
8   data_path=pathlib.Path('./Images')
9
10  train_ds=image_dataset_from_directory(data_path,validation_split=0.2,subset='training',seed=123,
                                          image_size=(224,224),batch_size=16)
11  test_ds=image_dataset_from_directory(data_path,validation_split=0.2,subset='validation',seed=123,
                                          image_size=(224,224),batch_size=16)
12
13  base_model=DenseNet121(weights='imagenet',include_top=False,input_shape=(224,224,3))
14  cnn=Sequential()
```

Section 07 견종 인식 프로그램

❑ 미세 조정 방식의 전이 학습: 견종 인식하기

- ex11_19.ipynb

```
15  cnn.add(Rescaling(1.0/255.0))
16  cnn.add(base_model)
17  cnn.add(Flatten())
18  cnn.add(Dense(1024,activation='relu'))
19  cnn.add(Dropout(0.75))
20  cnn.add(Dense(units=120,activation='softmax'))
21
22  cnn.compile(loss='sparse_categorical_crossentropy',optimizer=Adam(learning_rate=0.000001),
                metrics=['acc'])
23  hist=cnn.fit(train_ds,epochs=200,validation_data=test_ds,verbose=2)
24  print('정확률=',cnn.evaluate(test_ds,verbose=0)[1]*100)
25
26  cnn.save('cnn_for_stanford_dogs.h5')          # 미세 조정된 모델을 파일에 저장
27
28  import pickle
29  f=open('dog_species_names.txt','wb')
30  pickle.dump(train_ds.class_names,f)
31  f.close()
32
```

Section 07 견종 인식 프로그램

❑ 미세 조정 방식의 전이 학습: 견종 인식하기

- ex11_19.ipynb

```
33  import matplotlib.pyplot as plt
34
35  plt.plot(hist.history['accuracy'])
36  plt.plot(hist.history['val_accuracy'])
37  plt.title('Accuracy graph')
38  plt.ylabel('Accuracy')
39  plt.xlabel('Epoch')
40  plt.legend(['Train','Validation'])
41  plt.grid()
42  plt.show()
43
44  plt.plot(hist.history['loss'])
45  plt.plot(hist.history['val_loss'])
46  plt.title('Loss graph')
47  plt.ylabel('Loss')
48  plt.xlabel('Epoch')
49  plt.legend(['Train','Validation'])
50  plt.grid()
51  plt.show()
```

Section 07 견종 인식 프로그램

❑ 미세 조정 방식의 전이 학습: 견종 인식하기

- 예제 11-19에서 구축한 cnn 객체는 ImageNet으로 학습된 모델을 Stanford dogs 데이터로 전이 학습할 준비가 된 신경망이다.
- 이 신경망을 학습할 때 주의할 점이 있다.
- 이 신경망의 앞쪽에 배치된 base_model에 해당하는 층은 ImageNet으로 사전 학습된 상태고, 뒤에 새로 추가한 완전연결층은 학습되지 않은, 즉 난수를 설정한 상태다.
- 22~23행은 compile과 fit 함수로 학습을 실행한다.
- 22행은 교차 엔트로피 손실 함수를 쓰기 위해 loss='sparse_categorical_crossentropy'로 설정했다.
- sparse_categorical_crossentropy는 앞에서 사용했던 categorical_crossentropy와 동작이 같은데, 단지 부류 정보를 원핫 코드가 아니라 정수로 표현한 경우에 사용한다.

Section 07 견종 인식 프로그램

❑ 미세 조정 방식의 전이 학습: 견종 인식하기

- optimizer=Adam(learning_rate=0,000001) **인수는 학습률을 기본값** (0.001)**보다 훨씬 작은 값으로 설정했다**.
- **크게 설정하면 사전 학습된 앞부분이 흐트러져 오히려 성능이 낮아지기 때문이다**.
- **대신** fit **함수에서** epochs=200**으로 설정하여 세대를 길게 진행한다**.
- 25**행은** 11 **행에서 마련한 테스트 집합으로 정확률을 측정하고 출력한다**.
- **지금까지 수행한 바와 같이, 사전 학습된 모델 뒤에 새로운 층을 붙여 신경망을 구성하고 학습률을 낮게 설정해 다시 학습하는 방식을 미세 조정**(fine-tuning)**이라 부른다**.

Section 07 견종 인식 프로그램

❑ 미세 조정 방식의 전이 학습: 견종 인식하기

- Stanford dogs 데이터셋은MNIST나 CIFAR-10과 같은 장난감 데이터셋과 달리 학습에 많은 시간이 소요된다.
- GPU를 설치한 PC에서 한 세대를 학습하는 데 170초 가량 걸려서 200세대로 설정한 [프로그램 8-기은 9시간 30분 가량 걸렸다.
- 따라서 학습을 마치면 모델을 저장해 필요할 때 불러 쓸 수 있게 한다.
- 27행은 save 함수를 이용하여 모델을 저장한다.
- 이 모델 파일은 다음 예제에서 견종을 인식하는 비전 에이전트를 제작하는 데 쓰인다.

Section 07 견종 인식 프로그램

❑ 미세 조정 방식의 전이 학습: 견종 인식하기

- ex11_20.ipynb

```
1   import cv2
2   import numpy as np
3   import tensorflow as tf
4   import pickle
5   import sys
6   from PyQt5.QtWidgets import *
7
8   cnn=tf.keras.models.load_model('cnn_for_stanford_dogs.h5')          # 모델 읽기
9   dog_species=pickle.load(open('dog_species_names.txt','rb'))         # 견종 이름
10
11  class DogSpeciesRecognition(QMainWindow):
12      def __init__(self) :
13          super().__init__()
14          self.setWindowTitle('견종 인식')
15          self.setGeometry(200,200,700,100)
16
17          fileButton=QPushButton('강아지 사진 열기',self)
18          recognitionButton=QPushButton('품종 인식',self)
19          quitButton=QPushButton('나가기',self)
20
```

Section 07 견종 인식 프로그램

❑ 미세 조정 방식의 전이 학습: 견종 인식하기

- ex11_20.ipynb

```
21        fileButton.setGeometry(10,10,100,30)
22        recognitionButton.setGeometry(110,10,100,30)
23        quitButton.setGeometry(510,10,100,30)
24
25        fileButton.clicked.connect(self.pictureOpenFunction)
26        recognitionButton.clicked.connect(self.recognitionFunction)
27        quitButton.clicked.connect(self.quitFunction)
28
29    def pictureOpenFunction(self):
30        fname=QFileDialog.getOpenFileName(self,'강아지 사진 읽기','./')
31        self.img=cv2.imread(fname[0])
32        if self.img is None: sys.exit('파일을 찾을 수 없습니다.')
33
34        cv2.imshow('Dog image',self.img)
35
```

Section 07 견종 인식 프로그램

❑ 미세 조정 방식의 전이 학습: 견종 인식하기

- ex11_20.ipynb

```
36      def recognitionFunction(self):
37          x=np.reshape(cv2.resize(self.img,(224,224)),(1,224,224,3))
38          res=cnn.predict(x)[0]                    # 예측
39          top5=np.argsort(-res)[:5]
40          top5_dog_species_names=[dog_species[i] for i in top5]
41          for i in range(5):
42              prob='('+str(res[top5[i]])+')'
43              name=str(top5_dog_species_names[i]).split('-')[1]
44              cv2.putText(self.img,prob+name,(10,100+i*30),cv2.FONT_HERSHEY_SIMPLEX,0.7,(255,255,255),2)
45          cv.imshow('Dog image',self.img)
46
47      def quitFunction(self):
48          cv.destroyAllWindows()
49          self.close()
50
51  app=QApplication(sys.argv)
52  win=DogSpeciesRecognition()
53  win.show()
54  app.exec_()
```

Q&A