

케라스, 텐서플로우 버전 확인

In [20]:

```
1 import keras
2 keras.__version__
```

Out[20]:

'2.3.1'

In [21]:

```
1 import tensorflow as tf
2 tf.__version__
```

Out[21]:

'2.0.0'

사용 라이브러리 및 이미지 불러오기

In [105]:

```
1 import warnings
2 warnings.filterwarnings('ignore')
3
4 from keras import models, layers
5 from keras.callbacks import ModelCheckpoint, EarlyStopping
6 import cv2
7 from glob import glob
8 import os
9 import numpy as np
10 from IPython.display import SVG
11 from keras.utils.vis_utils import model_to_dot
12 import tensorflow as tf
13 from tensorflow import keras
14
15 from keras import regularizers
16 from sklearn.model_selection import train_test_split
17 from tensorflow.keras.utils import to_categorical
18 from keras.models import Sequential
19 from keras.layers import Dense, Activation, Dropout, Flatten, Conv2D, MaxPooling2D, BatchNormalization
20 from keras.callbacks import ModelCheckpoint, EarlyStopping
21 import matplotlib.pyplot as plt
```

In [106]:

```
1 img_data = glob('C:\Users\WW82106\W\Desktop\WWsw_0601\WWpokemon\lww*.jpg')
2 class_name = ['Charmander', 'Gastly', 'Goldeen', 'Gyarados', 'Horsea', 'Mew', 'Mewtwo', 'Pikachu', 'Poliwag', 'Squirtle']
3 dic = {'Charmander':0, 'Gastly':1, 'Goldeen':2, 'Gyarados':3, 'Horsea':4, 'Mew':5, 'Mewtwo':6, 'Pikachu':7, 'Poliwag':8, 'Squirtle':9}
4 dic2 = {0:'Charmander', 1:'Gastly', 2:'Goldeen', 3:'Gyarados', 4:'Horsea', 5:'Mew', 6:'Mewtwo', 7:'Pikachu', 8:'Poliwag', 9:'Squirtle'}
```

이미지, 레이블들을 저장

In [133]:

```
1 #데이터들을 담을 리스트 정의
2 X_all = list()
3 #레이블들을 담을 리스트 정의
4 Y_all = list()
5
6
7 for imagename in img_data:
8     try:
9         img = cv2.imread(imagename)
10        img = cv2.resize(img, dsize=(64, 64))
11        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
12
13        image = np.array(img)
14        X_all.append(img)
15
16        label = imagename.split('WW')
17        label = label[6]
18        label = label.split('.')
19        label = str(label[0])
20        label = dic[label]
21        Y_all.append(label)
22    except :
23        pass # 예외
24
25
26 # X, Y리스트들을 NP형식의 배열로 생성
27 X_all = np.array(X_all)
28 Y_all = np.array(Y_all)
29
30 print(X_all)
31 print(Y_all)
32 print('X_all shape: ', X_all.shape)
33 print('Y_all shape: ', Y_all.shape)
```

```
[[ 0  0  0]
 [ 0  1  0]
 [ 0  0  0]
 ...
 [ 0  0  0]
 [ 0  0  0]
 [ 0  1  0]]

[[ 0  0  0]
 [ 1  0  0]
 [ 0  1  0]
 ...
 [ 0  0  0]
 [ 0  2  0]
 [ 0  0  0]]]
[0 0 0 ... 9 9 9]
X_all shape:  (20159, 64, 64, 3)
Y_all shape:  (20159,)
```

train, test 데이터셋 분리

In [134]:

```
1 X_train,X_test,Y_train,Y_test = train_test_split(X_all, Y_all, test_size = 0.2, shuffle=True, random_state=44)
2 print(X_train.shape)
3 print(X_test.shape)
4 print(Y_train.shape)
5 print(Y_test.shape)
```

```
(16127, 64, 64, 3)
(4032, 64, 64, 3)
(16127,)
(4032,)
```

정규화 및 원핫인코딩

```
In [135]: 1 X_train = X_train.reshape(X_train.shape[0], 64, 64, 3)
2 X_test = X_test.reshape(X_test.shape[0], 64, 64, 3)
3 X_train = X_train.astype('float') / 255
4 X_test = X_test.astype('float') / 255
5
6 print('X_train_shape: ', X_train.shape)
7 print('X_test_shape: ', X_test.shape)
8 print(X_train[:5])
9 print(X_test[:5])
```

```
...
[[1. 1. 1. ]
 [1. 1. 1. ]
 [1. 1. 1. ]]

...
[[1. 1. 1. ]
 [1. 1. 1. ]
 [1. 1. 1. ]

...
[[1. 1. 1. ]
 [1. 1. 1. ]
 [1. 1. 1. ]

...
[[1. 1. 1. ]
 [1. 1. 1. ]
 [1. 1. 1. ]]]
```

```
In [136]: 1 Y_train = to_categorical(Y_train, 10)
2 Y_test = to_categorical(Y_test, 10)
3 print('Y_train_shape:', Y_train.shape)
4 print('Y_test_shape', Y_test.shape)
```

Y_train_shape: (16127, 10)
Y_test_shape (4032, 10)

Autoencoder - Unsupervised Learning

```
In [137]: 1 from tensorflow.keras.models import Sequential, Model
2 from tensorflow.keras.layers import Input, Dense, Conv2D, MaxPooling2D, UpSampling2D, Flatten, Reshape
3
4 autoencoder = Sequential()
5
6 # 인코딩 부분입니다.
7 autoencoder.add(Conv2D(16, kernel_size=3, padding='same', input_shape=(64,64,3), activation='relu'))
8 autoencoder.add(MaxPooling2D(pool_size=2, padding='same'))
9 autoencoder.add(Conv2D(8, kernel_size=3, activation='relu', padding='same'))
10 autoencoder.add(Conv2D(8, kernel_size=3, strides=2, padding='same', activation='relu'))
11
12 # 디코딩 부분이 이어집니다.
13 autoencoder.add(Conv2D(8, kernel_size=3, padding='same', activation='relu'))
14 autoencoder.add(UpSampling2D())
15 autoencoder.add(Conv2D(8, kernel_size=3, padding='same', activation='relu'))
16 autoencoder.add(UpSampling2D())
17 #autoencoder.add(Conv2D(16, kernel_size=3, activation='relu'))
18 #autoencoder.add(UpSampling2D())
19 autoencoder.add(Conv2D(3, kernel_size=3, padding='same', activation='sigmoid'))
20
21 # 전체 구조를 확인해 봅니다.
22 autoencoder.summary()
23
24 # 컴파일 및 학습을 하는 부분입니다.
25 autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
26 autoencoder.fit(X_train, X_train, epochs=20, batch_size=50, validation_data=(X_test, X_test))
```

Model: "sequential_71"

Layer (type)	Output Shape	Param #
=====		
conv2d_385 (Conv2D)	(None, 64, 64, 16)	448
=====		
max_pooling2d_108 (MaxPoolin	(None, 32, 32, 16)	0
=====		
conv2d_386 (Conv2D)	(None, 32, 32, 8)	1160
=====		
conv2d_387 (Conv2D)	(None, 16, 16, 8)	584
=====		
conv2d_388 (Conv2D)	(None, 16, 16, 8)	584
=====		
up_sampling2d_165 (UpSamplin	(None, 32, 32, 8)	0
=====		
conv2d_389 (Conv2D)	(None, 32, 32, 8)	584
=====		
up_sampling2d_166 (UpSamplin	(None, 64, 64, 8)	0
=====		
conv2d_390 (Conv2D)	(None, 64, 64, 3)	219
=====		
Total params: 3,579		
Trainable params: 3,579		
Non-trainable params: 0		

Train on 16127 samples, validate on 4032 samples

Epoch 1/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.4495 - val_loss: 0.3925

Epoch 2/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3900 - val_loss: 0.3818

Epoch 3/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3830 - val_loss: 0.3771

Epoch 4/20
16127/16127 [=====] - 51s 3ms/sample - loss: 0.3783 - val_loss: 0.3727

Epoch 5/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3740 - val_loss: 0.3690

Epoch 6/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3714 - val_loss: 0.3700

Epoch 7/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3700 - val_loss: 0.3661

Epoch 8/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3689 - val_loss: 0.3654

Epoch 9/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3676 - val_loss: 0.3638

Epoch 10/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3667 - val_loss: 0.3631

Epoch 11/20
16127/16127 [=====] - 51s 3ms/sample - loss: 0.3663 - val_loss: 0.3624

Epoch 12/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3656 - val_loss: 0.3621

Epoch 13/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3651 - val_loss: 0.3631

Epoch 14/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3644 - val_loss: 0.3625

Epoch 15/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3639 - val_loss: 0.3607

Epoch 16/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3637 - val_loss: 0.3609

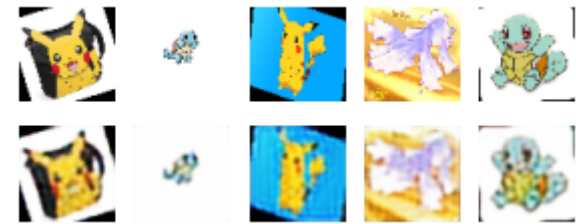
Epoch 17/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3631 - val_loss: 0.3601

Epoch 18/20
16127/16127 [=====] - 50s 3ms/sample - loss: 0.3628 - val_loss: 0.3593
Epoch 19/20
16127/16127 [=====] - 51s 3ms/sample - loss: 0.3627 - val_loss: 0.3590
Epoch 20/20
16127/16127 [=====] - 51s 3ms/sample - loss: 0.3619 - val_loss: 0.3585

Out[137]: <tensorflow.python.keras.callbacks.History at 0x170378159e8>

결과 출력

```
In [139]: 1 #학습된 결과를 출력하는 부분입니다.
2 random_test = np.random.randint(X_test.shape[0], size=5) #테스트할 이미지를 랜덤하게 불러옵니다.
3 ae_imgs = autoencoder.predict(X_test) #앞서 만든 오토인코더 모델에 집어 넣습니다.
4
5 plt.figure(figsize=(7, 2)) #출력될 이미지의 크기
6
7 for i, image_idx in enumerate(random_test): #랜덤하게 뽑은 이미지를 차례로 나열
8     ax = plt.subplot(2, 7, i + 1) ### VGG16 - Transfer Learning
9     plt.imshow(X_test[image_idx].reshape(64, 64, 3)) #테스트할 이미지
10    ax.axis('off')
11    ax = plt.subplot(2, 7, 7 + i + 1)
12    plt.imshow(ae_imgs[image_idx].reshape(64, 64, 3)) #오토인코딩 결과를 다음열에 출력
13    ax.axis('off')
14
15 plt.show()
```



```
In [141]: 1 #학습된 결과를 출력하는 부분입니다.
2 random_test = np.random.randint(X_test.shape[0], size=5) #테스트할 이미지를 랜덤하게 불러옵니다.
3 ae_imgs = autoencoder.predict(X_test) #앞서 만든 오토인코더 모델에 집어 넣습니다.
4
5 plt.figure(figsize=(7, 2)) #출력될 이미지의 크기
6
7 for i, image_idx in enumerate(random_test): #랜덤하게 뽑은 이미지를 차례로 나열
8     ax = plt.subplot(2, 7, i + 1)
9     plt.imshow(X_test[image_idx].reshape(64, 64, 3)) #테스트할 이미지
10    ax.axis('off')
11    ax = plt.subplot(2, 7, 7 + i + 1)
12    plt.imshow(ae_imgs[image_idx].reshape(64, 64, 3)) #오토인코딩 결과를 다음열에 출력
13    ax.axis('off')
14
15 plt.show()
```

