케라스, 텐서플로우 버전 확인

In [20]:
```
import keras
keras.__version__
```

Out[20]: '2.3.1'

In [21]:
```
import tensorflow as tf
tf.__version__
```

Out[21]: '2.0.0'

사용 라이브러리 및 이미지 불러오기

In [105]:
```
import warnings
warnings.filterwarnings('ignore')

from keras import models, layers
from keras.callbacks import ModelCheckpoint,EarlyStopping
import cv2
from glob import glob
import os
import numpy as np
from IPython.display import SVG
from keras.utils.vis_utils import model_to_dot
import tensorflow as tf
from tensorflow import keras

from keras import regularizers
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Activation,Dropout, Flatten, Conv2D, MaxPooling2D,BatchNormalization
from keras.callbacks import ModelCheckpoint,EarlyStopping
import matplotlib.pyplot as plt
```

In [106]:
```
img_data = glob('C:₩₩Users₩₩82106₩₩Desktop₩₩sw_0601₩₩pokemon_l₩₩*.jpg')
class_name = ['Charmander', 'Gastly', 'Goldeen', 'Gyarados', 'Horsea', 'Mew', 'Mewtwo', 'Pikachu', 'Poliwag', 'Squirtle']
dic = {'Charmander':0,'Gastly':1,'Goldeen':2,'Gyarados':3,'Horsea':4,'Mew':5,'Mewtwo':6,'Pikachu':7,'Poliwag':8,'Squirtle':9}
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 [143]:
```
#데이터들을 담을 리스트 정의
X_all = list()
#레이블들을 담을 리스트 정의
Y_all = list()


for imagename in img_data:
    try:
        img = cv2.imread(imagename)
        img = cv2.resize(img, dsize=(128, 128))
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

        image = np.array(img)
        X_all.append(img)

        label = imagename.split('WW')
        label = label[6]
        label = label.split('.')
        label = str(label[0])
        label = dic[label]
        Y_all.append(label)
    except :
        pass # 예외


# X, Y리스트들을 NP형식의 배열로 생성
X_all = np.array(X_all)
Y_all = np.array(Y_all)

print(X_all)
print(Y_all)
print('X_all shape: ', X_all.shape)
print('Y_all shape: ', Y_all.shape)
```

```
  [[  0   0   0]
   [  0   0   0]
   [  6   1   1]
   ...
   [  1   4   1]
   [  0   0   0]
   [  0   0   0]]

  [[  0   0   0]
   [  0   0   0]
   [  0   0   0]
   ...
   [  4   7   3]
   [  0   0   0]
   [  0   0   0]]]]
[0 0 0 ... 9 9 9]
X_all shape:  (20159, 128, 128, 3)
Y_all shape:  (20159,)
```

train, test 데이터셋 분리

In [144]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X_all, Y_all, test_size = 0.2, shuffle=True, random_state=44)
print(X_train.shape)
print(X_test.shape)
print(Y_train.shape)
print(Y_test.shape)
```

```
(16127, 128, 128, 3)
(4032, 128, 128, 3)
(16127,)
(4032,)
```

정규화 및 원핫인코딩

In [145]:
```
X_train = X_train.reshape(X_train.shape[0], 128, 128, 3)
X_test = X_test.reshape(X_test.shape[0], 128, 128, 3)
X_train = X_train.astype('float') / 255
X_test = X_test.astype('float') / 255

print('X_train_shape: ', X_train.shape)
print('X_test_shape: ', X_test.shape)
print(X_train[:5])
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.        ]
 ...
 [1.         1.         1.        ]
 [1.         1.         1.        ]
 [1.         1.         1.        ]]]]
```

In [146]:
```
Y_train = to_categorical(Y_train, 10)
Y_test = to_categorical(Y_test, 10)
print('Y_train_shape:', Y_train.shape)
print('Y_test_shape', Y_test.shape)
```

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

Autoencoder - Unsupervised Learning

In [147]:
```
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Input, Dense, Conv2D, MaxPooling2D, UpSampling2D, Flatten, Reshape

autoencoder = Sequential()

# 인코딩 부분입니다.
autoencoder.add(Conv2D(16, kernel_size=3, padding='same', input_shape=(128,128,3), activation='relu'))
autoencoder.add(MaxPooling2D(pool_size=2, padding='same'))
autoencoder.add(Conv2D(8, kernel_size=3, activation='relu', padding='same'))
autoencoder.add(Conv2D(8, kernel_size=3, strides=2, padding='same', activation='relu'))

# 디코딩 부분이 이어집니다.
autoencoder.add(Conv2D(8, kernel_size=3, padding='same', activation='relu'))
autoencoder.add(UpSampling2D())
autoencoder.add(Conv2D(8, kernel_size=3, padding='same', activation='relu'))
autoencoder.add(UpSampling2D())
#autoencoder.add(Conv2D(16, kernel_size=3, activation='relu'))
#autoencoder.add(UpSampling2D())
autoencoder.add(Conv2D(3, kernel_size=3, padding='same', activation='sigmoid'))

# 전체 구조를 확인해 봅니다.
autoencoder.summary()

# 컴파일 및 학습을 하는 부분입니다.
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')
autoencoder.fit(X_train, X_train, epochs=20, batch_size=50, validation_data=(X_test, X_test))
```

```
Model: "sequential_72"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_391 (Conv2D)          (None, 128, 128, 16)      448
_________________________________________________________________
max_pooling2d_109 (MaxPoolin (None, 64, 64, 16)        0
_________________________________________________________________
conv2d_392 (Conv2D)          (None, 64, 64, 8)         1160
_________________________________________________________________
conv2d_393 (Conv2D)          (None, 32, 32, 8)         584
_________________________________________________________________
conv2d_394 (Conv2D)          (None, 32, 32, 8)         584
_________________________________________________________________
up_sampling2d_167 (UpSamplin (None, 64, 64, 8)         0
_________________________________________________________________
conv2d_395 (Conv2D)          (None, 64, 64, 8)         584
_________________________________________________________________
up_sampling2d_168 (UpSamplin (None, 128, 128, 8)       0
_________________________________________________________________
conv2d_396 (Conv2D)          (None, 128, 128, 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 [==============================] - 190s 12ms/sample - loss: 0.4325 - val_loss: 0.3718
Epoch 2/20
16127/16127 [==============================] - 190s 12ms/sample - loss: 0.3673 - val_loss: 0.3596
Epoch 3/20
16127/16127 [==============================] - 191s 12ms/sample - loss: 0.3611 - val_loss: 0.3565
Epoch 4/20
16127/16127 [==============================] - 192s 12ms/sample - loss: 0.3586 - val_loss: 0.3545
Epoch 5/20
16127/16127 [==============================] - 201s 12ms/sample - loss: 0.3560 - val_loss: 0.3517
Epoch 6/20
16127/16127 [==============================] - 211s 13ms/sample - loss: 0.3548 - val_loss: 0.3500
Epoch 7/20
16127/16127 [==============================] - 201s 12ms/sample - loss: 0.3511 - val_loss: 0.3465
Epoch 8/20
16127/16127 [==============================] - 194s 12ms/sample - loss: 0.3486 - val_loss: 0.3446
Epoch 9/20
16127/16127 [==============================] - 195s 12ms/sample - loss: 0.3479 - val_loss: 0.3442
Epoch 10/20
16127/16127 [==============================] - 192s 12ms/sample - loss: 0.3468 - val_loss: 0.3449
Epoch 11/20
16127/16127 [==============================] - 193s 12ms/sample - loss: 0.3459 - val_loss: 0.3428
Epoch 12/20
16127/16127 [==============================] - 196s 12ms/sample - loss: 0.3456 - val_loss: 0.3441
Epoch 13/20
16127/16127 [==============================] - 197s 12ms/sample - loss: 0.3450 - val_loss: 0.3423
Epoch 14/20
16127/16127 [==============================] - 197s 12ms/sample - loss: 0.3448 - val_loss: 0.3411
Epoch 15/20
16127/16127 [==============================] - 196s 12ms/sample - loss: 0.3443 - val_loss: 0.3430
Epoch 16/20
16127/16127 [==============================] - 202s 13ms/sample - loss: 0.3444 - val_loss: 0.3442
Epoch 17/20
16127/16127 [==============================] - 204s 13ms/sample - loss: 0.3436 - val_loss: 0.3404
```

```
Epoch 18/20
16127/16127 [==============================] - 200s 12ms/sample - loss: 0.3432 - val_loss: 0.3399
Epoch 19/20
16127/16127 [==============================] - 203s 13ms/sample - loss: 0.3431 - val_loss: 0.3407
Epoch 20/20
16127/16127 [==============================] - 199s 12ms/sample - loss: 0.3429 - val_loss: 0.3399
```

Out[147]: <tensorflow.python.keras.callbacks.History at 0x170326baf28>

결과 출력

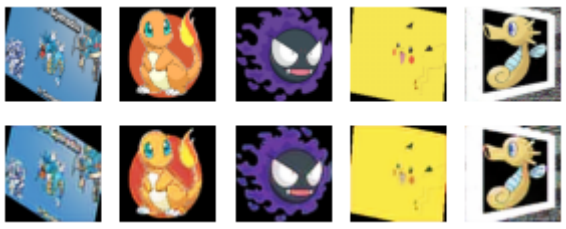
In [148]:

```
#학습된 결과를 출력하는 부분입니다.
random_test = np.random.randint(X_test.shape[0], size=5)  #테스트할 이미지를 랜덤하게 불러옵니다.
ae_imgs = autoencoder.predict(X_test)  #앞서 만든 오토인코더 모델에 집어 넣습니다.

plt.figure(figsize=(7, 2))  #출력될 이미지의 크기

for i, image_idx in enumerate(random_test):    #랜덤하게 뽑은 이미지를 차례로 나열
    ax = plt.subplot(2, 7, i + 1)
    plt.imshow(X_test[image_idx].reshape(128, 128, 3))  #테스트할 이미지
    ax.axis('off')
    ax = plt.subplot(2, 7, 7 + i +1)
    plt.imshow(ae_imgs[image_idx].reshape(128, 128, 3))  #오토인코딩 결과를 다음열에 출력
    ax.axis('off')

plt.show()
```



In [150]:

```
#학습된 결과를 출력하는 부분입니다.
random_test = np.random.randint(X_test.shape[0], size=5)  #테스트할 이미지를 랜덤하게 불러옵니다.
ae_imgs = autoencoder.predict(X_test)  #앞서 만든 오토인코더 모델에 집어 넣습니다.

plt.figure(figsize=(7, 2))  #출력될 이미지의 크기

for i, image_idx in enumerate(random_test):    #랜덤하게 뽑은 이미지를 차례로 나열
    ax = plt.subplot(2, 7, i + 1)
    plt.imshow(X_test[image_idx].reshape(128, 128, 3))  #테스트할 이미지
    ax.axis('off')
    ax = plt.subplot(2, 7, 7 + i +1)
    plt.imshow(ae_imgs[image_idx].reshape(128, 128, 3))  #오토인코딩 결과를 다음열에 출력
    ax.axis('off')

plt.show()
```

In [151]:
```
# 오토인코더로 생성된 사진을 저장하는 부분 # test 사진이 4032개 였으므로 오토 인코더로 생성되는 사진의 개수도 동일

if not os.path.exists("./auto_images"):
    os.makedirs("./auto_images")

random_test = np.random.randint(X_test.shape[0], size=5)
ae_imgs = autoencoder.predict(X_test)

plt.figure(figsize=(7, 2))

for i, image_idx in enumerate(random_test):
    ax = plt.subplot(2, 7, i + 1)
    plt.imshow(X_test[image_idx].reshape(128, 128, 3))
    ax.axis('off')
    ax = plt.subplot(2, 7, 7 + i +1)
    plt.imshow(ae_imgs[image_idx].reshape(128, 128, 3))
    ax.axis('off')

plt.show()

import matplotlib.pyplot as plt

count = 0

for img in ae_imgs:
    plt.imshow(img[0])
    plt.imsave(str(count) + '.' + 'jpg', img)
    count+=1
```