빅데이터 기반 AI 응용 솔루션 개발자 전문과정

교과목명: 딥러닝알고리즘 구현

- 평가일: 2022. 11. 18
- 성명: 신창훈
- 점수:70

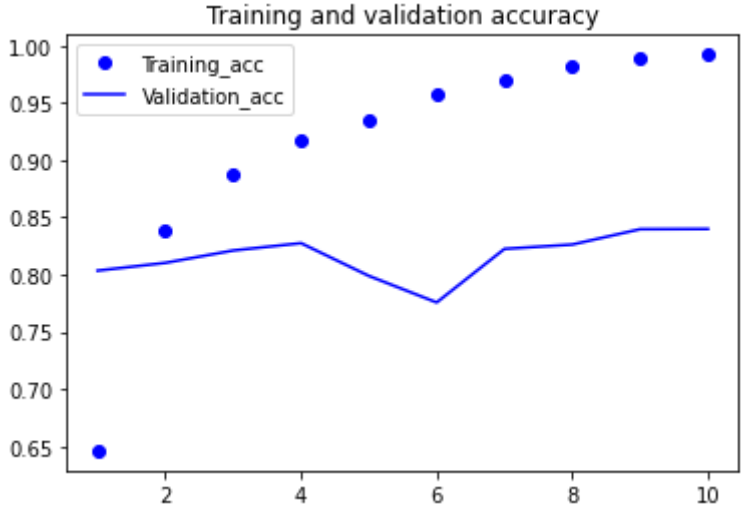
Q1. 사람이 문장을 읽는 것처럼 이전에 나온 것을 기억하면서 단어별로 또는 한눈에 들어오는 만큼씩 처리하여 문장에 있은 의미를 자연스럽게 표현하려는 목적으로 과거 정보를 사용하고 새롭게 얻은 정보를 계속 업데이트하는 방식이 순환 신경망(RNN) 이다. SimpleRNN을 활용하여 IMDB 영화 리뷰 데이터에 대하여 아래 사항을 수행하세요.

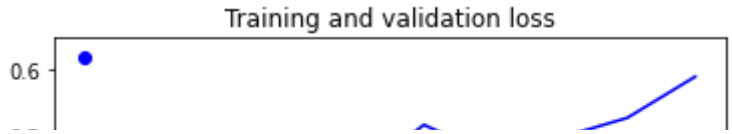
- 데이터 전처리 : max_features 10000, maxlen = 500, batch_size 32
- 케라스를 사용하여 입력 시퀀스에 대한 마지막 출력만 반환하는 방식으로 모델링. (embedding 층 입력 (max_features, 32))
- 학습 및 검증 옵션 : epochs 10, batch_size 128, 검증 데이터 20% ※ 학습시간 20분
- 훈련과 검증의 손실과 정확도를 그래프로 표현
- 검증 정확도를 확인하고 동 사례에 SimpleRNN 모델의 적합 여부 및 개선 방안에 대하여 기술하세요.

```
from keras.datasets import imdb
from tensorflow.keras.preprocessing import sequence

max_features = 10000
maxlen = 500
batch_size = 32

(input_train, y_train), (input_test, y_test) = imdb.load_data(num_words = max_features)

input_train = sequence.pad_sequences(input_train, maxlen=maxlen)
input_test = sequence.pad_sequences(input_test, maxlen=maxlen)


from keras.layers import Dense, SimpleRNN, Embedding
from keras.models import Sequential

model = Sequential()
model.add(Embedding(max_features, 32))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['acc'])
history = model.fit(input_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

```
==========================] - 73s 442ms/step - loss: 0.6162 - acc: 0.6461 - val_loss:
==========================] - 67s 429ms/step - loss: 0.3843 - acc: 0.8386 - val_loss:
==========================] - 67s 428ms/step - loss: 0.2873 - acc: 0.8874 - val_loss:
==========================] - 67s 429ms/step - loss: 0.2159 - acc: 0.9177 - val_loss:
==========================] - 68s 432ms/step - loss: 0.1882 - acc: 0.9344 - val_loss:
==========================] - 66s 421ms/step - loss: 0.1225 - acc: 0.9579 - val_loss:
==========================] - 66s 423ms/step - loss: 0.0890 - acc: 0.9701 - val_loss:
==========================] - 68s 430ms/step - loss: 0.0602 - acc: 0.9818 - val_loss:
==========================] - 68s 435ms/step - loss: 0.0379 - acc: 0.9884 - val_loss:
==========================] - 67s 424ms/step - loss: 0.0283 - acc: 0.9920 - val_loss:
```

```
import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc)+1)

plt.plot(epochs, acc, 'bo', label='Training_acc')
plt.plot(epochs, val_acc, 'b', label='Validation_acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training_loss')
plt.plot(epochs, val_loss, 'b', label='Validation_loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```





위의 사례 검증정확도 : 83%

- 위의 모델의 성능을 개선하기 위해선 텍스트 갯수를 500개로 한정하지 않고 더 늘려서 데이터의 양을 늘리는 방안과 네트워크의 크기를 늘려 정확도를 높이는 방안이 있다.

0.2]

Q2. Q1 문제를 LSTM 모델을 적용하여 수행하세요

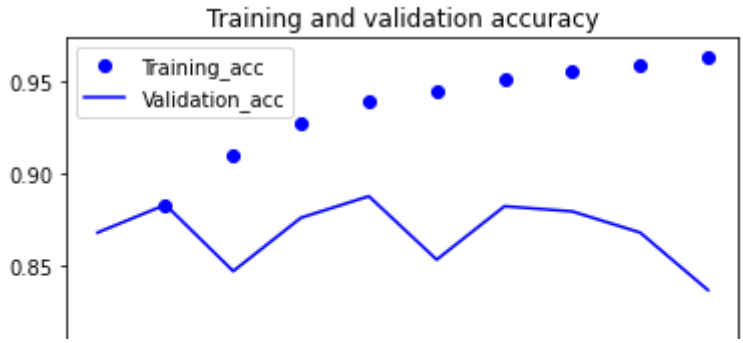
- 모델링, 학습 및 검증
- 결과 시각화

```
from keras.layers import LSTM

model = Sequential()
model.add(Embedding(max_features, 32))
model.add(LSTM(32))
model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])

history = model.fit(input_train, y_train,
                    epochs=10,
                    batch_size=128,
                    validation_split=0.2)
```

```
Epoch 1/10
157/157 [==============================] - 9s 24ms/step - loss: 0.5252 - acc: 0.7490
Epoch 2/10
157/157 [==============================] - 3s 20ms/step - loss: 0.2973 - acc: 0.8824
Epoch 3/10
157/157 [==============================] - 4s 23ms/step - loss: 0.2317 - acc: 0.9101
Epoch 4/10
157/157 [==============================] - 3s 20ms/step - loss: 0.2011 - acc: 0.9275
Epoch 5/10
```

```
157/157 [==============================] - 3s 20ms/step - loss: 0.1721 - acc: 0.9385
Epoch 6/10
157/157 [==============================] - 3s 21ms/step - loss: 0.1550 - acc: 0.9449
Epoch 7/10
157/157 [==============================] - 3s 20ms/step - loss: 0.1385 - acc: 0.9512
Epoch 8/10
157/157 [==============================] - 3s 21ms/step - loss: 0.1273 - acc: 0.9552
Epoch 9/10
157/157 [==============================] - 3s 20ms/step - loss: 0.1183 - acc: 0.9582
Epoch 10/10
157/157 [==============================] - 3s 20ms/step - loss: 0.1111 - acc: 0.9625
```

```
import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc)+1)

plt.plot(epochs, acc, 'bo', label='Training_acc')
plt.plot(epochs, val_acc, 'b', label='Validation_acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training_loss')
plt.plot(epochs, val_loss, 'b', label='Validation_loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```



Q3. MNIST 숫자 이미지 데이터에 대하여 CNN 모델을 사용하여 아래사항을 수행하세요

- Conv2D와 MaxPooling2D 층을 사용하여 컨브넷을 생성(채널의 수 32개 또는 64개)
- 출력 텐서를 완전 연결 네트워크에 주입
- 10개의 클래스 분류하기 위한 분류기 추가
- 컨브넷 학습 및 평가

```
from keras import layers
from keras import models

model = models.Sequential()
model.add(layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.MaxPooling2D((2,2)))
model.add(layers.Conv2D(64, (3,3), activation='relu'))
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))
```

```
from keras.datasets import mnist
from keras.utils import to_categorical

(train_images, train_labels),(test_images, test_labels) = mnist.load_data()

train_images = train_images.reshape((60000, 28,28,1))
train_images = train_images.astype('float32')/255

test_images = test_images.reshape((10000, 28,28,1))
test_images = test_images.astype('float32')/255

train_labels = to_categorical(train_labels)
test_labels = to_categorical(test_labels)

model.compile(optimizer='rmsprop', loss='categorical_crossentropy', metrics=['acc'])
model.fit(train_images, train_labels, epochs=5, batch_size = 64)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mni
11490434/11490434 [==============================] - 2s 0us/step
Epoch 1/5
938/938 [==============================] - 9s 5ms/step - loss: 0.1666 - acc: 0.9479
```

```
Epoch 2/5
938/938 [==============================] - 4s 5ms/step - loss: 0.0448 - acc: 0.9857
Epoch 3/5
938/938 [==============================] - 4s 5ms/step - loss: 0.0316 - acc: 0.9903
Epoch 4/5
938/938 [==============================] - 4s 5ms/step - loss: 0.0240 - acc: 0.9927
Epoch 5/5
938/938 [==============================] - 4s 5ms/step - loss: 0.0195 - acc: 0.9937
<keras.callbacks.History at 0x7f3a9eb67ad0>
```

```
test_loss, test_acc = model.evaluate(test_images, test_labels)
test_acc
```

```
313/313 [==============================] - 1s 3ms/step - loss: 0.0273 - acc: 0.9924
0.9923999905586243
```

Q4. cats_and_dogs_small으로 축소한 데이터 셋으로 사전 훈련된 네트워크를 사용하여 강아지 고양이 분류 과제를 아래와 같이 수행하세요.

- ImageNet 데이터셋에 훈련된 VGG16 네트워크의 합성곱 기반 층을 사용하여 유용한 특성 추출하고 이 특성으로 분류기 훈련
- ImageDataGenerator 사용 (※ 소요시간 20분)
- VGG 매개변수
  - weights는 모델을 초기화할 가중치 체크포인트를 지정 : 'imagenet'
  - include_top은 네트워크의 최상위 완전 연결 분류기를 포함할지 안할지를 지정 : False
  - input_shape은 네트워크에 주입할 이미지 텐서의 크기 :(150.150,3)
- 데이터 증식을 사용하지 않는 방법으로 수행
- 완전 연결 분류기를 정의하고 규제를 위해 드롭아웃 사용 : 0.5

```
from keras.applications import VGG16

conv_base = VGG16(weights='imagenet',
                  include_top=False,
                  input_shape=(150,150,3))
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vg
58889256/58889256 [==============================] - 3s 0us/step
```

```
!pwd
```

```
/content
```

```
%cd /content/drive/MyDrive/m9_딥러닝기본
```

```
/content/drive/MyDrive/m9_딥러닝기본
```

```
from google.colab import drive
```

```
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# 데이터 증식 사용X
import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator

base_dir = './cats_and_dogs_small'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
test_dir = os.path.join(base_dir, 'test')

datagen = ImageDataGenerator(rescale=1./255)
batch_size=20
```

```
def extract_features(directory, sample_count):
  features = np.zeros(shape=(sample_count, 4,4,512))
  labels = np.zeros(shape=(sample_count))
  generator = datagen.flow_from_directory(
      directory,
      target_size=(150,150),
      batch_size = batch_size,
      class_mode = 'binary'
  )
  i = 0
  for inputs_batch, labels_batch in generator:
    features_batch = conv_base.predict(inputs_batch)
    features[i*batch_size : (i+1)*batch_size] = features_batch
    labels[i*batch_size : (i+1)*batch_size] = labels_batch
    i+=1
    if i*batch_size >= sample_count:
      break
  return features, labels
```

```
train_features, train_labels = extract_features(train_dir, 2000)
validation_features, validation_labels = extract_features(validation_dir, 1000)
test_features, test_labels = extract_features(test_dir, 1000)
```

```
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 18ms/step
```

```
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 24ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 30ms/step
```

```
train_features = np.reshape(train_features, (2000, 4 * 4 * 512))
validation_features = np.reshape(validation_features, (1000, 4 * 4 * 512))
test_features = np.reshape(test_features, (1000, 4 * 4 * 512))
```

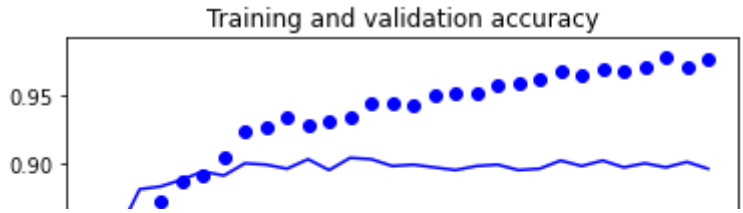
```
from keras import models, layers, optimizers

model = models.Sequential()
model.add(layers.Dense(256, activation='relu', input_dim=4*4*512))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
```

```
model.compile(optimizer = optimizers.RMSprop(lr=2e-5),
              loss = 'binary_crossentropy',
              metrics=['acc'])

history = model.fit(train_features, train_labels,
                    epochs=30,
                    batch_size=20,
                    validation_data = (validation_features, validation_labels))
```

```
Epoch 1/30
100/100 [==============================] - 1s 6ms/step - loss: 0.5769 - acc: 0.687
Epoch 2/30
100/100 [==============================] - 0s 4ms/step - loss: 0.4134 - acc: 0.815
Epoch 3/30
100/100 [==============================] - 0s 4ms/step - loss: 0.3460 - acc: 0.858
Epoch 4/30
100/100 [==============================] - 0s 4ms/step - loss: 0.3170 - acc: 0.871
Epoch 5/30
100/100 [==============================] - 0s 4ms/step - loss: 0.2846 - acc: 0.886
Epoch 6/30
100/100 [==============================] - 0s 5ms/step - loss: 0.2622 - acc: 0.891
Epoch 7/30
100/100 [==============================] - 0s 4ms/step - loss: 0.2456 - acc: 0.904
Epoch 8/30
100/100 [==============================] - 0s 4ms/step - loss: 0.2220 - acc: 0.924
Epoch 9/30
100/100 [==============================] - 0s 4ms/step - loss: 0.2138 - acc: 0.926
Epoch 10/30
100/100 [==============================] - 0s 4ms/step - loss: 0.1944 - acc: 0.934
Epoch 11/30
100/100 [==============================] - 0s 4ms/step - loss: 0.1974 - acc: 0.928
Epoch 12/30
100/100 [==============================] - 0s 5ms/step - loss: 0.1807 - acc: 0.931
Epoch 13/30
100/100 [==============================] - 0s 5ms/step - loss: 0.1785 - acc: 0.934
Epoch 14/30
100/100 [==============================] - 0s 4ms/step - loss: 0.1669 - acc: 0.943
Epoch 15/30
100/100 [==============================] - 0s 4ms/step - loss: 0.1567 - acc: 0.944
Epoch 16/30
100/100 [==============================] - 0s 4ms/step - loss: 0.1504 - acc: 0.942
Epoch 17/30
100/100 [==============================] - 0s 5ms/step - loss: 0.1453 - acc: 0.950
Epoch 18/30
100/100 [==============================] - 0s 4ms/step - loss: 0.1374 - acc: 0.951
Epoch 19/30
100/100 [==============================] - 0s 4ms/step - loss: 0.1360 - acc: 0.951
Epoch 20/30
100/100 [==============================] - 0s 4ms/step - loss: 0.1283 - acc: 0.958
Epoch 21/30
100/100 [==============================] - 0s 5ms/step - loss: 0.1236 - acc: 0.959
Epoch 22/30
100/100 [==============================] - 0s 5ms/step - loss: 0.1180 - acc: 0.961
Epoch 23/30
100/100 [==============================] - 0s 4ms/step - loss: 0.1125 - acc: 0.967
Epoch 24/30
100/100 [==============================] - 0s 4ms/step - loss: 0.1057 - acc: 0.964
Epoch 25/30
```

```
100/100 [==============================] - 0s 4ms/step - loss: 0.1004 - acc: 0.969
Epoch 26/30
100/100 [==============================] - 0s 4ms/step - loss: 0.0977 - acc: 0.967
Epoch 27/30
100/100 [==============================] - 0s 4ms/step - loss: 0.0941 - acc: 0.970
Epoch 28/30
100/100 [==============================] - 0s 5ms/step - loss: 0.0875 - acc: 0.977
Epoch 29/30
```

```
import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc)+1)

plt.plot(epochs, acc, 'bo', label='Training_acc')
plt.plot(epochs, val_acc, 'b', label='Validation_acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training_loss')
plt.plot(epochs, val_loss, 'b', label='Validation_loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```



Q5. Q4 문제를 데이터 증식을 사용한 방식으로 수행하세요.

```
from keras import models
from keras import layers

model = models.Sequential()
model.add(conv_base)
model.add(layers.Dense(256, activation='relu'))
model.add(layers.Dropout(0.5))
model.add(layers.Dense(1, activation='sigmoid'))
```

```
# 기반층 동결
conv_base.trainable = False
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=20,
      width_shift_range=0.1,
      height_shift_range=0.1,
      shear_range=0.1,
      zoom_range=0.1,
      horizontal_flip=True,
      fill_mode='nearest')

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(150, 150),
        batch_size=20,
        class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
        validation_dir,
        target_size=(150, 150),
        batch_size=20,
        class_mode='binary')

model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=2e-5),
              metrics=['acc'])

history = model.fit_generator(
```

```
 train_generator,
 steps_per_epoch=100,
 epochs=30,
 validation_data=validation_generator,
 validation_steps=50,
 verbose=2)

Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:37: UserWarning: `Model
Epoch 1/30
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-38-7d6c2ad2729d> in <module>
     35       validation_data=validation_generator,
     36       validation_steps=50,
---> 37       verbose=2)

                                  2 frames
/usr/local/lib/python3.7/dist-packages/keras/engine/training.py in
tf__train_function(iterator)
     13                 try:
     14                     do_return = True
---> 15                     retval_ = ag__.converted_call(ag__.ld(step_function),
(ag__.ld(self), ag__.ld(iterator)), None, fscope)
     16                 except:
     17                     do_return = False

ValueError: in user code:

    File "/usr/local/lib/python3.7/dist-packages/keras/engine/training.py", line
1051, in train_function  *
        return step_function(self, iterator)
    File "/usr/local/lib/python3.7/dist-packages/keras/engine/training.py", line
1040, in step_function  **
        outputs = model.distribute_strategy.run(run_step, args=(data,))
    File "/usr/local/lib/python3.7/dist-packages/keras/engine/training.py", line
1030, in run_step  **
        outputs = model.train_step(data)
    File "/usr/local/lib/python3.7/dist-packages/keras/engine/training.py", line
890, in train_step
        loss = self.compute_loss(x, y, y_pred, sample_weight)
    File "/usr/local/lib/python3.7/dist-packages/keras/engine/training.py", line
949, in compute_loss
        y, y_pred, sample_weight, regularization_losses=self.losses)
    File "/usr/local/lib/python3.7/dist-packages/keras/engine/compile_utils.py",
line 201, in __call__
        loss_value = loss_obj(y_t, y_p, sample_weight=sw)
    File "/usr/local/lib/python3.7/dist-packages/keras/losses.py", line 139, in
__call__
        losses = call_fn(y_true, y_pred)
    File "/usr/local/lib/python3.7/dist-packages/keras/losses.py", line 243, in call
**
        return ag_fn(y_true, y_pred, **self._fn_kwargs)
```

```
import matplotlib.pyplot as plt

acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(1, len(acc)+1)

plt.plot(epochs, acc, 'bo', label='Training_acc')
plt.plot(epochs, val_acc, 'b', label='Validation_acc')
plt.title('Training and validation accuracy')
plt.legend()

plt.figure()

plt.plot(epochs, loss, 'bo', label='Training_loss')
plt.plot(epochs, val_loss, 'b', label='Validation_loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```

Colab 유료 제품 - 여기에서 계약 취소

1초 오후 3:13에 완료됨