AND Gate with Keras

```
# tensorflow와 tf.keras를 임포트합니다
import tensorflow as tf
from tensorflow import keras
```

```
tf.__version__
```

```
'2.3.0'
```

```
# 헬퍼(helper) 라이브러리를 임포트합니다
import numpy as np
import matplotlib.pyplot as plt
```

```
#x_data = [[0, 0],
#          [0, 1],
#          [1, 0],
#          [1, 1]]

#y_data = [0,
#          0,
#          0,
#          1]

x_data = np.array([[1, 3], [1, 50], [10, 10], [3.7, 3.5], [7, 5], [9, 4], [2, 8]]) #training_points
y_data = [1, 1, 1, 0, 0, 0, 1]

# AND게이트는 입력값들중 2개의 입력값둘다 조건이 맞아야만 1을 준다.
# 여기서 1을주는 조건은 오른쪽의 숫자가 왼쪽보다 무조건 같거나 커야하다는것이다.

x_data = np.array(x_data, dtype=np.float32)
y_data = np.array(y_data, dtype=np.float32)
```

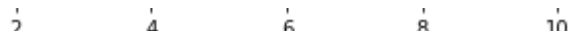
```
x_data.shape, y_data.shape
```

```
((7, 2), (7,))
```

```
plt.scatter(x_data[:, 0], x_data[:, 1], c=y_data)
```

```
<matplotlib.collections.PathCollection at 0x7fd7311b8ef0>
```


```
from tensorflow.keras import layers
from tensorflow.keras import activations
from tensorflow.keras import optimizers
from tensorflow.keras import models
```



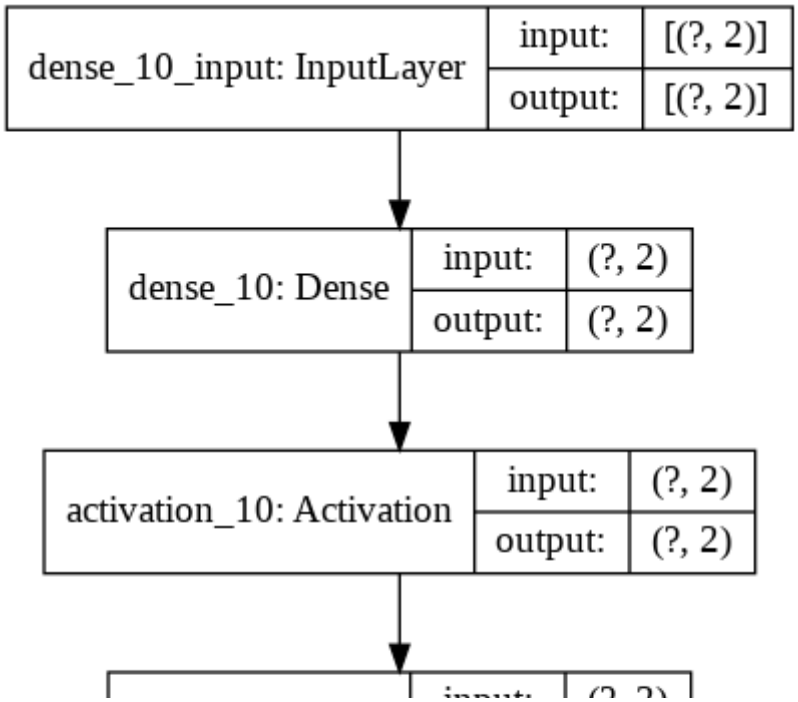

```
model = models.Sequential()
model.add(layers.Dense(2, input_dim=2))
model.add(layers.Activation('tanh'))
model.add(layers.Dense(1))
model.add(layers.Activation('sigmoid'))


sgd = optimizers.SGD(lr=0.1)
model.compile(loss='binary_crossentropy', optimizer=sgd, metrics=['accuracy'])


model.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_10 (Dense)             (None, 2)                 6         
_________________________________________________________________
activation_10 (Activation)   (None, 2)                 0         
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 3         
_________________________________________________________________
activation_11 (Activation)   (None, 1)                 0         
=================================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
```

```
from tensorflow.keras.utils import plot_model
plot_model(model, to_file='model_and.png', show_shapes=True)
```




```
history = model.fit(x_data, y_data,  batch_size=1, epochs=500)
```

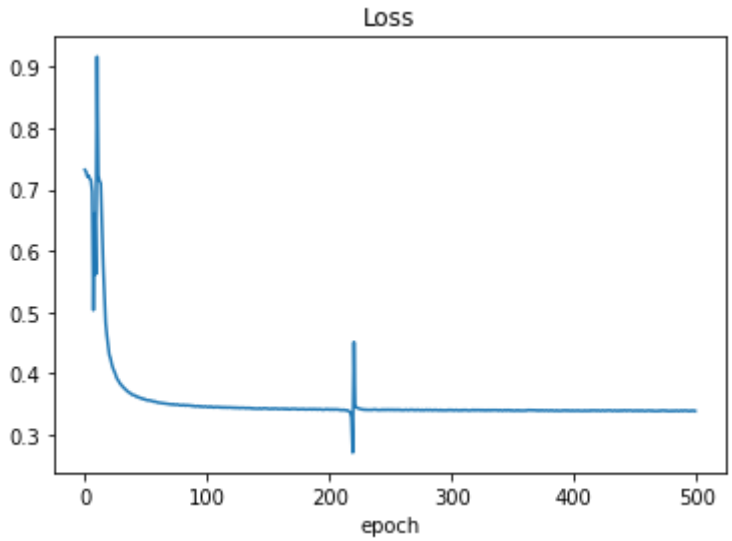
```
Epoch 1/500
7/7 [==============================] - 0s 2ms/step - loss: 0.7319 - accuracy: 0.5714
Epoch 2/500
7/7 [==============================] - 0s 1ms/step - loss: 0.7290 - accuracy: 0.5714
Epoch 3/500
7/7 [==============================] - 0s 2ms/step - loss: 0.7213 - accuracy: 0.5714
Epoch 4/500
7/7 [==============================] - 0s 1ms/step - loss: 0.7226 - accuracy: 0.5714
Epoch 5/500
7/7 [==============================] - 0s 1ms/step - loss: 0.7177 - accuracy: 0.5714
Epoch 6/500
7/7 [==============================] - 0s 1ms/step - loss: 0.7150 - accuracy: 0.4286
Epoch 7/500
7/7 [==============================] - 0s 1ms/step - loss: 0.6963 - accuracy: 0.5714
Epoch 8/500
7/7 [==============================] - 0s 1ms/step - loss: 0.5030 - accuracy: 0.8571
Epoch 9/500
7/7 [==============================] - 0s 1ms/step - loss: 0.6608 - accuracy: 0.7143
Epoch 10/500
7/7 [==============================] - 0s 1ms/step - loss: 0.5618 - accuracy: 0.7143
Epoch 11/500
7/7 [==============================] - 0s 1ms/step - loss: 0.9166 - accuracy: 0.4286
Epoch 12/500
7/7 [==============================] - 0s 1ms/step - loss: 0.7231 - accuracy: 0.5714
Epoch 13/500
7/7 [==============================] - 0s 1ms/step - loss: 0.7134 - accuracy: 0.5714
Epoch 14/500
7/7 [==============================] - 0s 1ms/step - loss: 0.7103 - accuracy: 0.5714
Epoch 15/500
7/7 [==============================] - 0s 1ms/step - loss: 0.6369 - accuracy: 0.5714
Epoch 16/500
7/7 [==============================] - 0s 2ms/step - loss: 0.5767 - accuracy: 0.5714
Epoch 17/500
7/7 [==============================] - 0s 1ms/step - loss: 0.5293 - accuracy: 0.5714
Epoch 18/500
7/7 [==============================] - 0s 1ms/step - loss: 0.4827 - accuracy: 0.8571
Epoch 19/500
```

```
7/7 [==============================] - 0s 1ms/step - loss: 0.4624 - accuracy: 0.7143
Epoch 20/500
7/7 [==============================] - 0s 1ms/step - loss: 0.4462 - accuracy: 0.8571
Epoch 21/500
7/7 [==============================] - 0s 1ms/step - loss: 0.4302 - accuracy: 0.8571
Epoch 22/500
7/7 [==============================] - 0s 1ms/step - loss: 0.4244 - accuracy: 0.8571
Epoch 23/500
7/7 [==============================] - 0s 1ms/step - loss: 0.4150 - accuracy: 0.8571
Epoch 24/500
7/7 [==============================] - 0s 1ms/step - loss: 0.4084 - accuracy: 0.8571
Epoch 25/500
7/7 [==============================] - 0s 2ms/step - loss: 0.4040 - accuracy: 0.8571
Epoch 26/500
7/7 [==============================] - 0s 1ms/step - loss: 0.3988 - accuracy: 0.8571
Epoch 27/500
7/7 [==============================] - 0s 1ms/step - loss: 0.3925 - accuracy: 0.8571
Epoch 28/500
7/7 [==============================] - 0s 2ms/step - loss: 0.3896 - accuracy: 0.8571
Epoch 29/500
7/7 [==============================] - 0s 2ms/step - loss: 0.3863 - accuracy: 0.8571
Epoch 30/500
```

학습 결과 그려보기

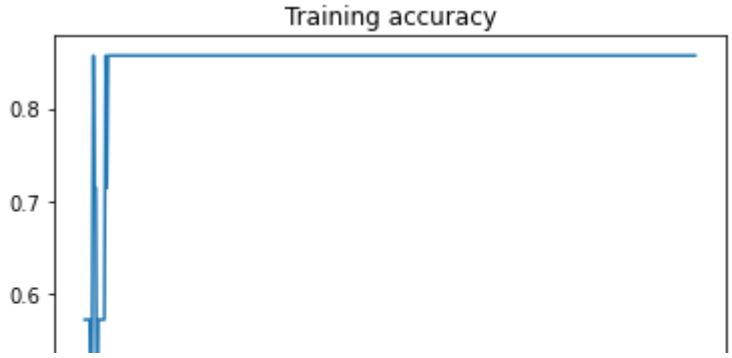
```
plt.plot(history.history['loss'])
plt.title('Loss')
plt.xlabel('epoch')
```

```
Text(0.5, 0, 'epoch')
```




```
plt.plot(history.history['accuracy'])
plt.title('Training accuracy')
plt.xlabel('epoch')
```

```
Text(0.5, 0, 'epoch')
```



확률 찍어보기

```
hypothesis = model.predict(x_data)
print(hypothesis)
```

```
WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_predict_function.<locals
[[0.99624777]
 [0.9977012 ]
 [0.25281313]
 [0.25371447]
 [0.25281313]
 [0.25281313]
 [0.99769723]]
```

```
predicted = hypothesis > 0.5
print(predicted)
```

```
[[ True]
 [ True]
 [False]
 [False]
 [False]
 [False]
 [ True]]
```

위의 결과 값을보면 [10,10]이 TRUE가 나와야 정상인데 1과 50이라는 다른 두수의 차이보다 훨씬 큰 차이를 주었기 때문에 아마 학습이 부정확하게 나오는거같다.

AND게이트는 75퍼센트의 확률로 0을 준다.

조건에 맞는 딱한가지의 값만 1을 주는것이다. 나머지는 모두 0으로 주고.