인공지능 (Artificial Intelligence)

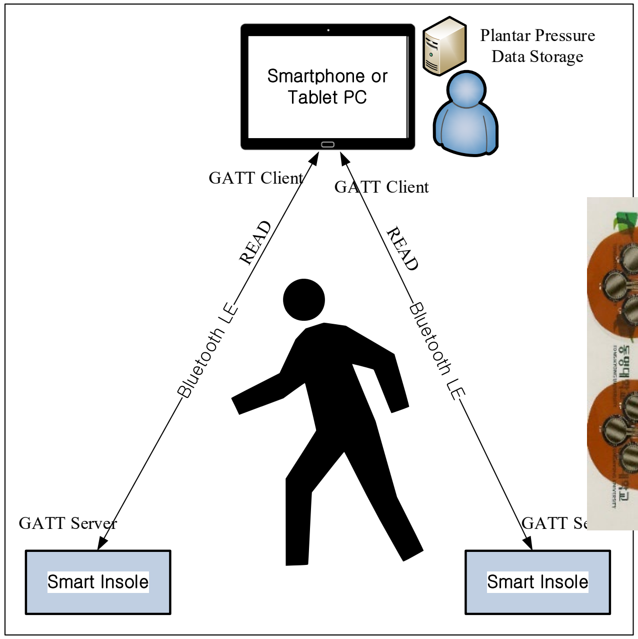
딥러닝

Hyuntae Cho

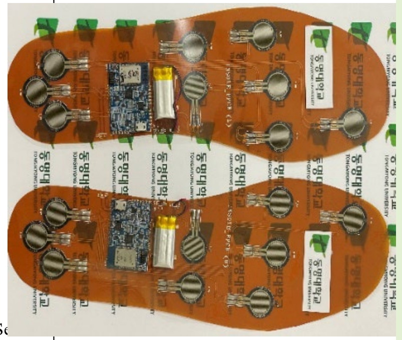
Dept. of Digital Media Engineering

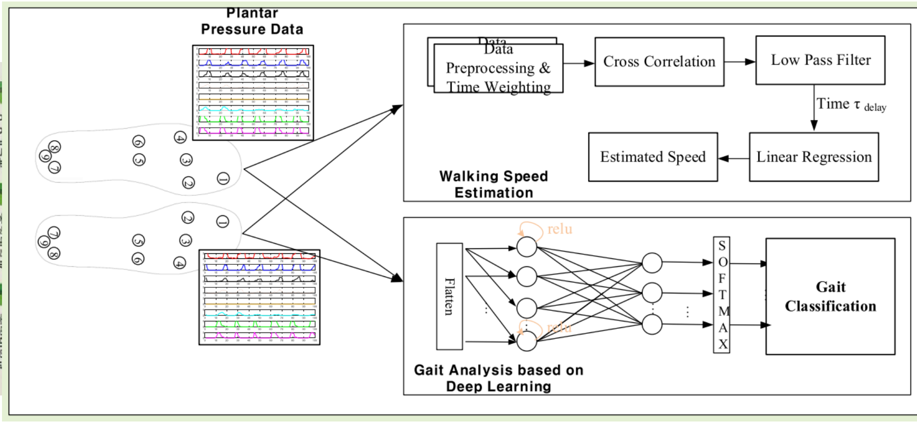
Tongmyong University

걸음 걸이 분석 해보자



Smartphone or Tablet PC
Plantar Pressure Data Storage
GATT Client
GATT Client
READ
READ
Bluetooth LE
Bluetooth LE
GATT Server
GATT Se
Smart Insole
Smart Insole



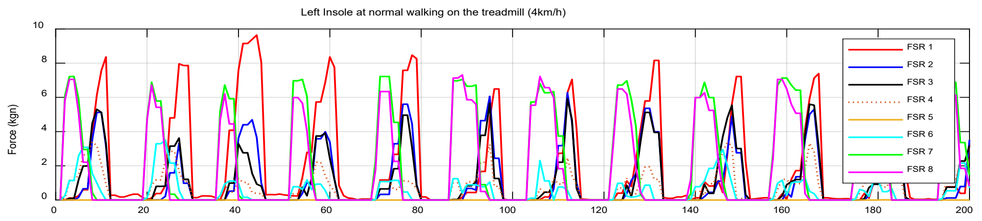


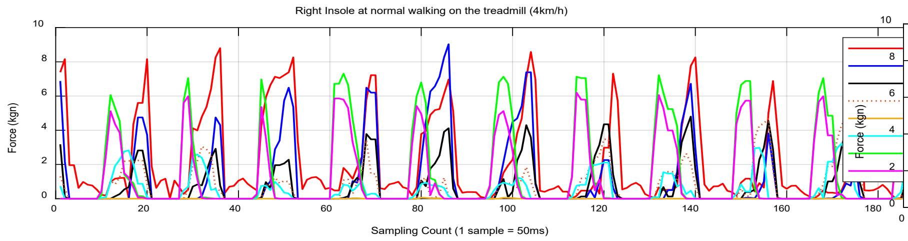
Plantar Pressure Data
Data Preprocessing & Time Weighting
Cross Correlation
Low Pass Filter
Time τ delay
Linear Regression
Estimated Speed
Walking Speed Estimation
Flatten
relu
SOFTMAX
Gait Classification
Gait Analysis based on Deep Learning

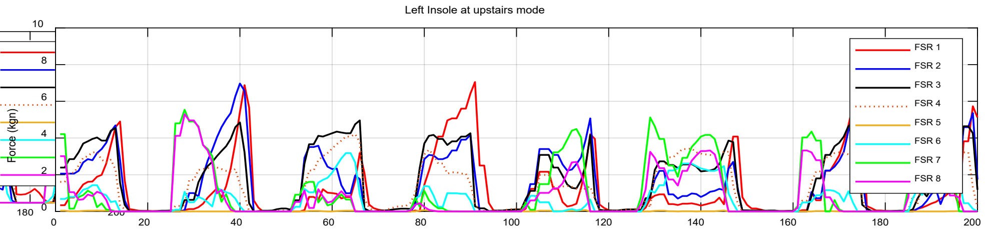
걸음 걸이 분석 해보자

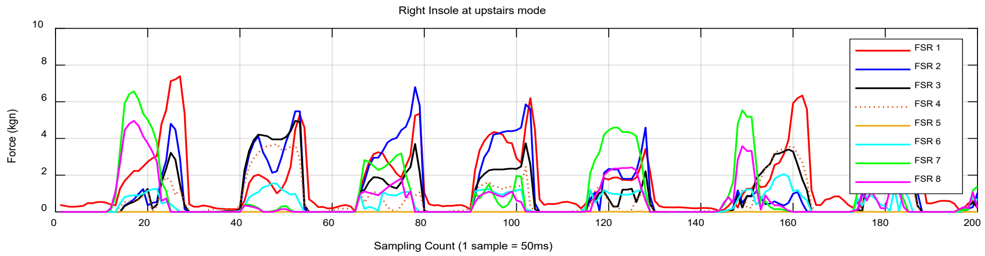
- 9종의 걸음걸이가 있음
  - 걸음걸이에 대한 패턴 가시화
  - 분류를 해보자

16개의 센서가 활용



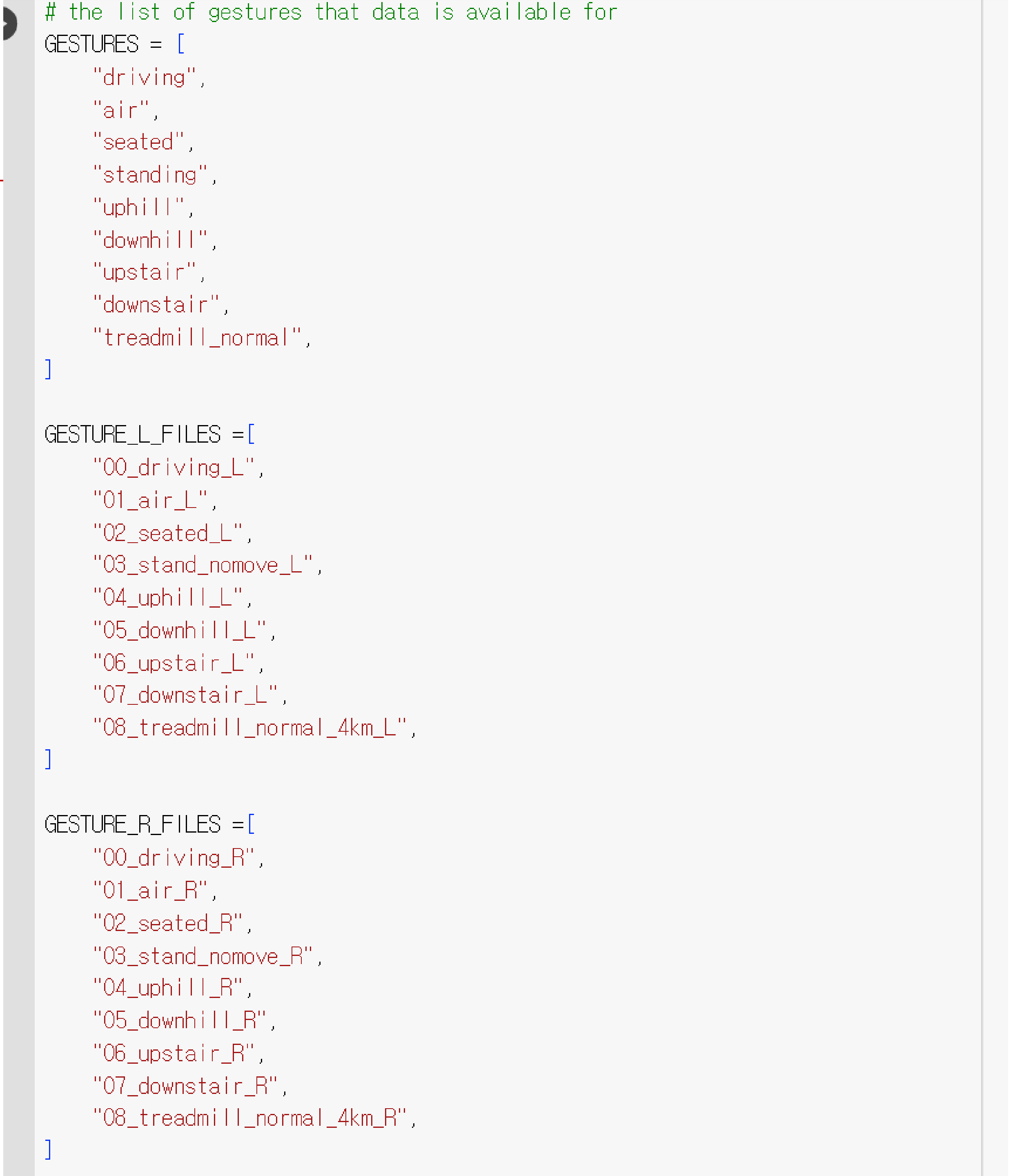






데이터 종류

- 오른쪽 9종류

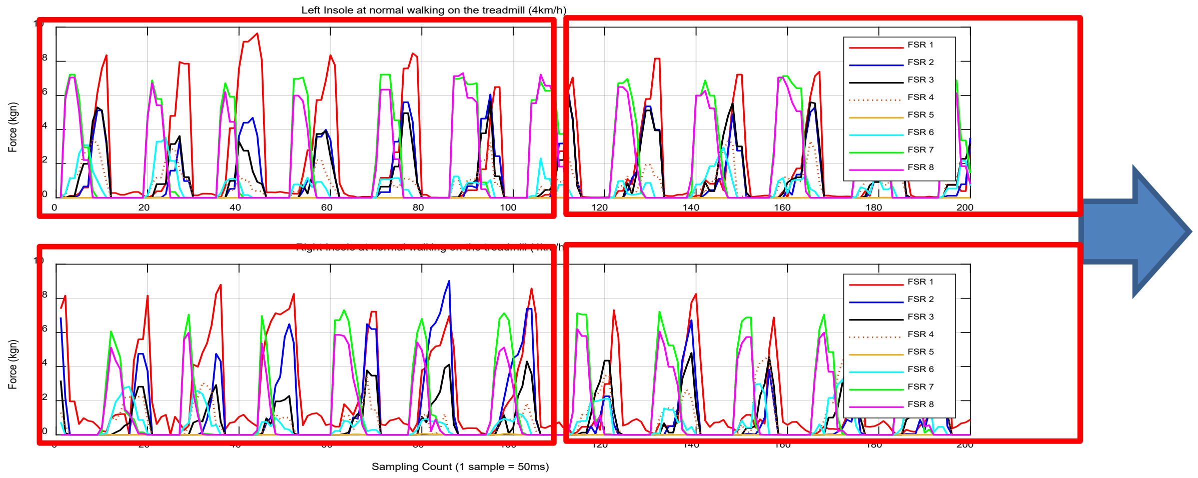


```
# the list of gestures that data is available for
GESTURES = [
    "driving",
    "air",
    "seated",
    "standing",
    "uphill",
    "downhill",
    "upstair",
    "downstair",
    "treadmill_normal",
]

GESTURE_L_FILES =[
    "00_driving_L",
    "01_air_L",
    "02_seated_L",
    "03_stand_nomove_L",
    "04_uphill_L",
    "05_downhill_L",
    "06_upstair_L",
    "07_downstair_L",
    "08_treadmill_normal_4km_L",
]

GESTURE_R_FILES =[
    "00_driving_R",
    "01_air_R",
    "02_seated_R",
    "03_stand_nomove_R",
    "04_uphill_R",
    "05_downhill_R",
    "06_upstair_R",
    "07_downstair_R",
    "08_treadmill_normal_4km_R",
]
```

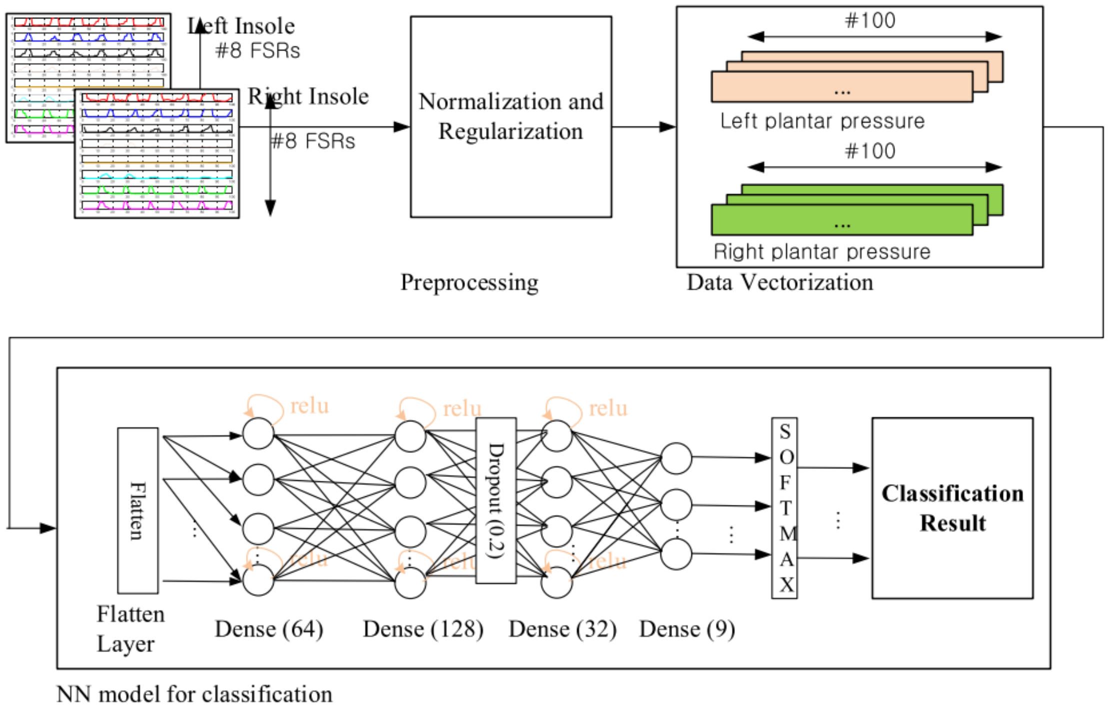
데이터 가공

- **전체 수집된 데이터를**
  - 16개 센서 x 100개 샘플을 하나의 데이터로 분류시켜 학습

  - 예측할 때도 16x 100개 샘플 데이터 확보



전체 절차


Left Insole
#8 FSRs
Right Insole
#8 FSRs
Normalization and Regularization
Preprocessing
#100
...
Left plantar pressure
#100
...
Right plantar pressure
Data Vectorization
relu
relu
relu
relu
relu
relu
Flatten
Dropout (0.2)
SOFTMAX
Classification Result
Flatten Layer
Dense (64)
Dense (128)
Dense (32)
Dense (9)
NN model for classification

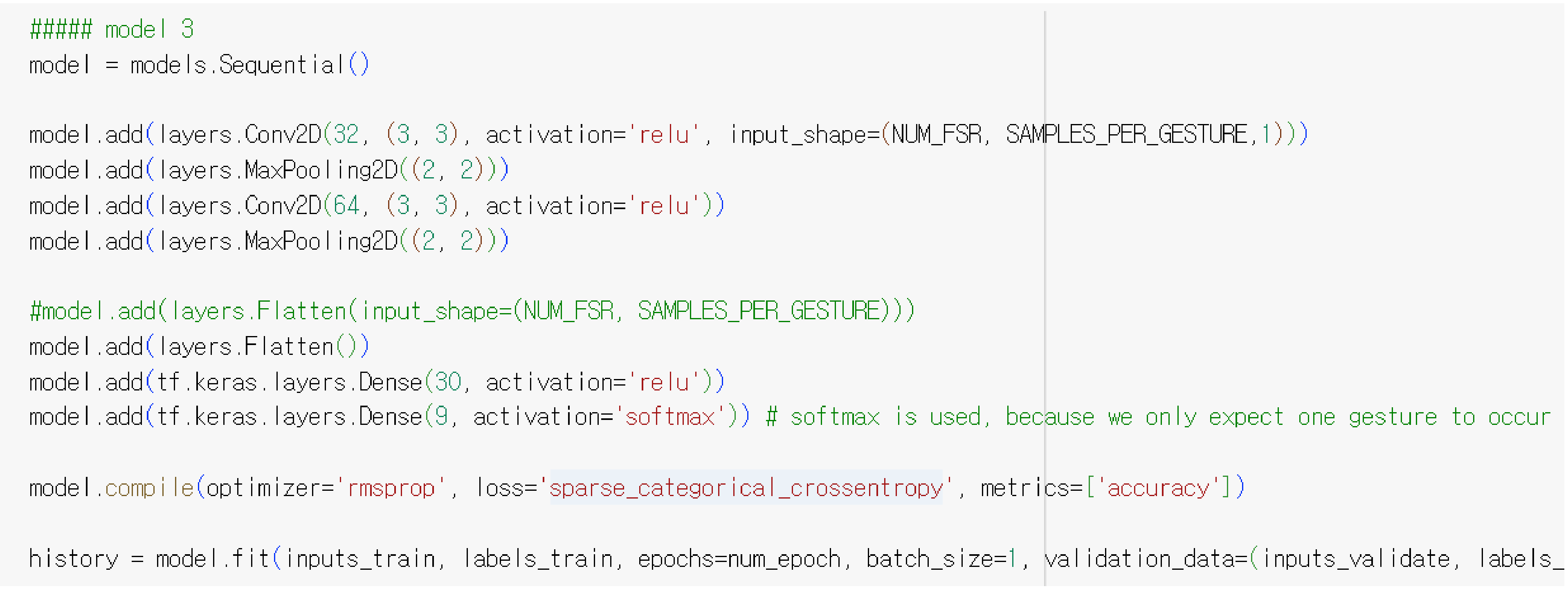
데이터 로드 및 분리

```
# Randomize the order of the inputs, so they can be evenly distributed for training, testing, and validation
# https://stackoverflow.com/a/37710486/2020087
# 전체 raw data를 랜덤하게 섞음
num_inputs = len(inputs)
randomize = np.arange(num_inputs)
np.random.shuffle(randomize)

# Swap the consecutive indexes (0, 1, 2, etc) with the randomized indexes
# 인덱스 삽입
inputs = inputs[randomize]
labels = labels[randomize]



# Split the recordings (group of samples) into three sets: training, testing and validation
TRAIN_SPLIT = int(0.6 * num_inputs)            #훈련용 60%
TEST_SPLIT = int(0.2 * num_inputs + TRAIN_SPLIT)#테스트용 20%

# 6:2:2로 데이터 나눔
inputs_train, inputs_test, inputs_validate = np.split(inputs, [TRAIN_SPLIT, TEST_SPLIT])      # RAW DATA
labels_train, labels_test, labels_validate = np.split(labels, [TRAIN_SPLIT, TEST_SPLIT])  # 정답 labels

print("Data set randomization and splitting complete.")
```

동명대학교
TONGMYONG UNIVERSITY

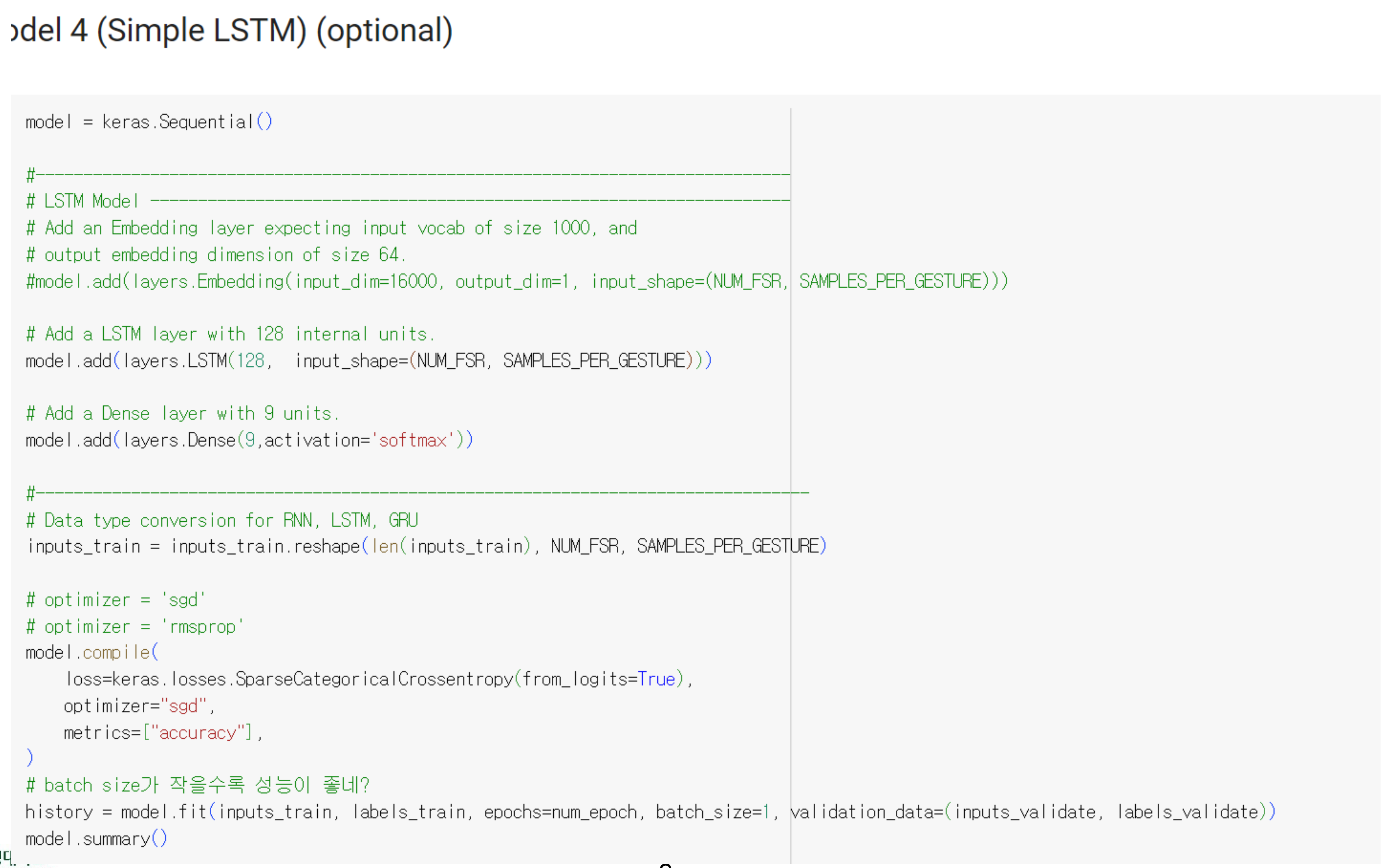
모델 설계

- **CNN**



```
##### model 3
model = models.Sequential()

model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(NUM_FSR, SAMPLES_PER_GESTURE,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.Flatten(input_shape=(NUM_FSR, SAMPLES_PER_GESTURE)))
model.add(layers.Flatten())
model.add(tf.keras.layers.Dense(30, activation='relu'))
model.add(tf.keras.layers.Dense(9, activation='softmax')) # softmax is used, because we only expect one gesture to occur

model.compile(optimizer='rmsprop', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

history = model.fit(inputs_train, labels_train, epochs=num_epoch, batch_size=1, validation_data=(inputs_validate, labels_
```

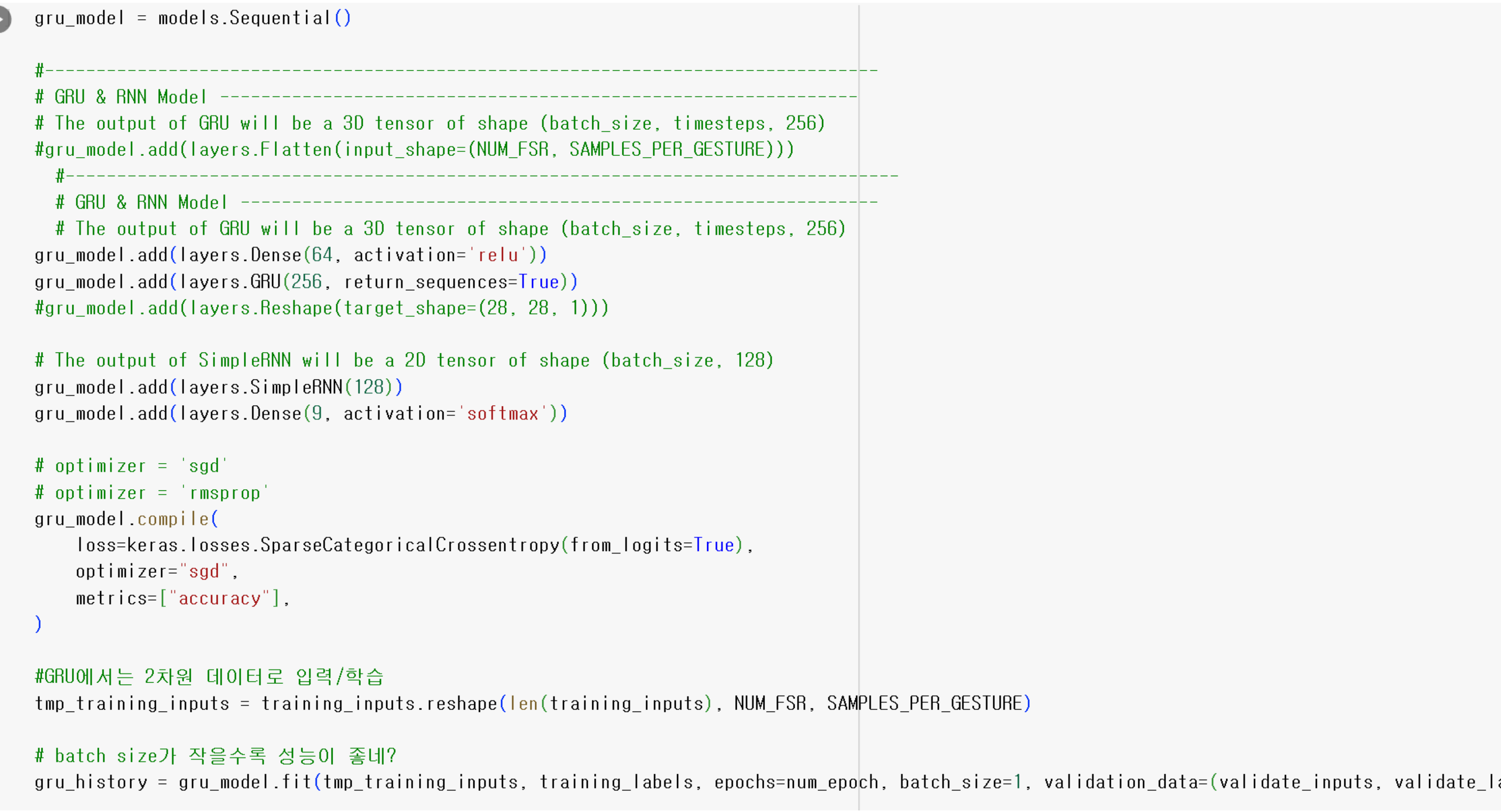
LSTM model



odel 4 (Simple LSTM) (optional)

```
model = keras.Sequential()

#-------------------------------------------------------------------------------
# LSTM Model -------------------------------------------------------------------
# Add an Embedding layer expecting input vocab of size 1000, and
# output embedding dimension of size 64.
#model.add(layers.Embedding(input_dim=16000, output_dim=1, input_shape=(NUM_FSR, SAMPLES_PER_GESTURE)))

# Add a LSTM layer with 128 internal units.
model.add(layers.LSTM(128,  input_shape=(NUM_FSR, SAMPLES_PER_GESTURE)))

# Add a Dense layer with 9 units.
model.add(layers.Dense(9,activation='softmax'))

#---------------------------------------------------------------------------------
# Data type conversion for RNN, LSTM, GRU
inputs_train = inputs_train.reshape(len(inputs_train), NUM_FSR, SAMPLES_PER_GESTURE)

# optimizer = 'sgd'
# optimizer = 'rmsprop'
model.compile(
    loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    optimizer="sgd",
    metrics=["accuracy"],
)
# batch size가 작을수록 성능이 좋네?
history = model.fit(inputs_train, labels_train, epochs=num_epoch, batch_size=1, validation_data=(inputs_validate, labels_validate))
model.summary()
```

GRU model



```
gru_model = models.Sequential()

#-----------------------------------------------------------------------------------
# GRU & RNN Model ---------------------------------------------------------------
# The output of GRU will be a 3D tensor of shape (batch_size, timesteps, 256)
#gru_model.add(layers.Flatten(input_shape=(NUM_FSR, SAMPLES_PER_GESTURE)))
  #-----------------------------------------------------------------------------------
  # GRU & RNN Model ---------------------------------------------------------------
  # The output of GRU will be a 3D tensor of shape (batch_size, timesteps, 256)
gru_model.add(layers.Dense(64, activation='relu'))
gru_model.add(layers.GRU(256, return_sequences=True))
#gru_model.add(layers.Reshape(target_shape=(28, 28, 1)))

# The output of SimpleRNN will be a 2D tensor of shape (batch_size, 128)
gru_model.add(layers.SimpleRNN(128))
gru_model.add(layers.Dense(9, activation='softmax'))

# optimizer = 'sgd'
# optimizer = 'rmsprop'
gru_model.compile(
    loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    optimizer="sgd",
    metrics=["accuracy"],
)

#GRU에서는 2차원 데이터로 입력/학습
tmp_training_inputs = training_inputs.reshape(len(training_inputs), NUM_FSR, SAMPLES_PER_GESTURE)

# batch size가 작을수록 성능이 좋네?
gru_history = gru_model.fit(tmp_training_inputs, training_labels, epochs=num_epoch, batch_size=1, validation_data=(validate_inputs, validate_l
```

Bidirectional LSTM

```
blstm_model = models.Sequential()

#-------------------------------------------------------------------------------
# bidirectional LSTM -----------------------------------------------------------
blstm_model.add(
    layers.Bidirectional(layers.LSTM(64, return_sequences=True), input_shape=(NUM_FSR, SAMPLES_PER_GESTURE)))
blstm_model.add(
    layers.Bidirectional(layers.LSTM(128, return_sequences=True), input_shape=(NUM_FSR, SAMPLES_PER_GESTURE)))
blstm_model.add(
    layers.Bidirectional(layers.LSTM(64)))
#model.add(layers.Bidirectional(layers.LSTM(32)))
blstm_model.add(layers.Dense(9))
#-------------------------------------------------------------------------------

# optimizer = 'sgd'
# optimizer = 'rmsprop'
blstm_model.compile(
    loss=keras.losses.SparseCategoricalCrossentropy(from_logits=True),
    optimizer="sgd",
    metrics=["accuracy"],
)
# batch size가 작을수록 성능이 좋네?
blstm_history = blstm_model.fit(training_inputs, training_labels, epochs=num_epoch, batch_size=1, validation_data=(validate_inputs, validate_l
```

모델 확인

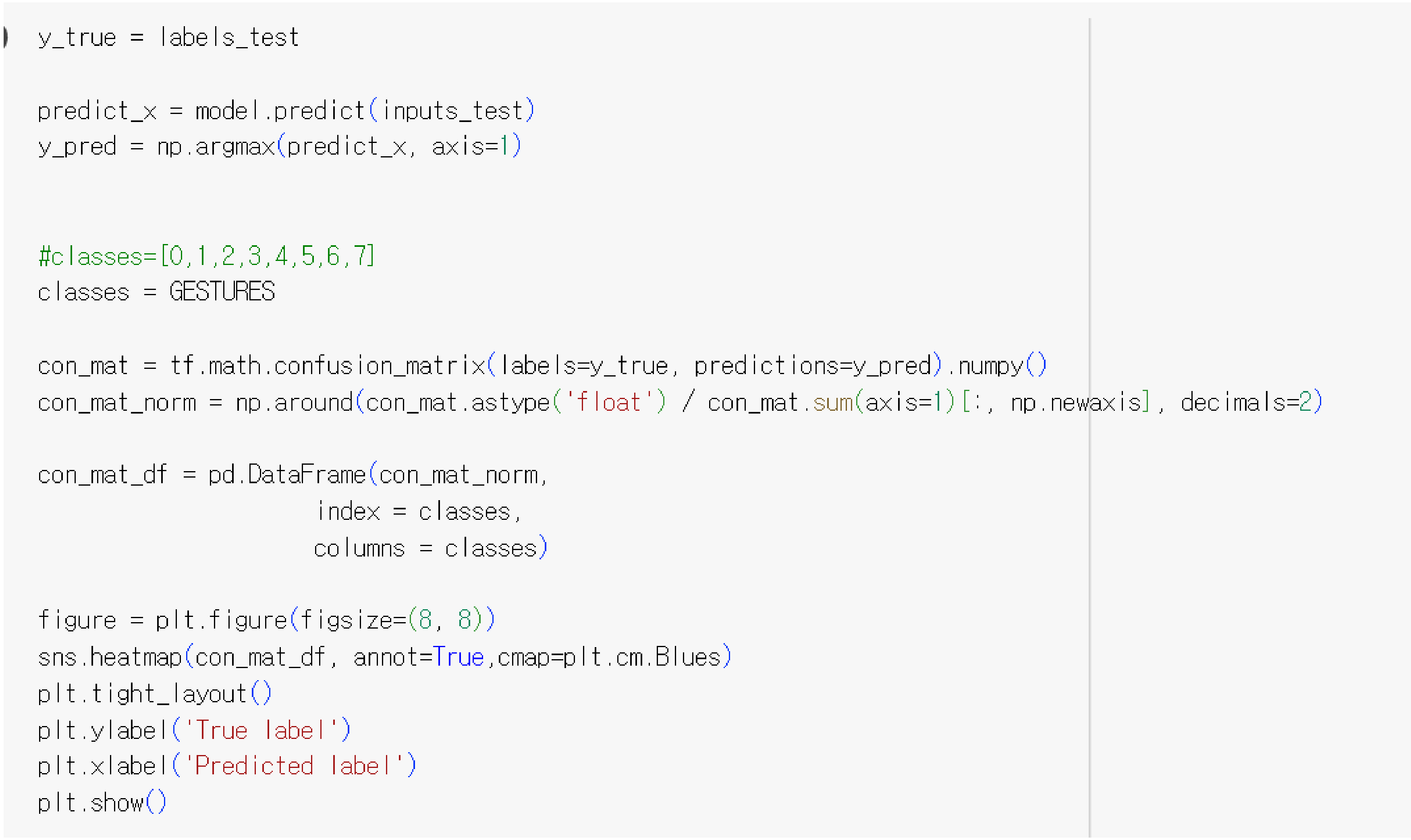
```
from tensorflow import keras
model.summary()
keras.utils.plot_model(model)
```

검증

```python
from sklearn.model_selection import cross_validate

print('Validatation data: ')
test_loss, test_acc = model.evaluate(inputs_validate, labels_validate, verbose=2)
print(test_acc)

print('Test data: ')

test_loss, test_acc = model.evaluate(inputs_test, labels_test, verbose=2)
print(test_acc)
```

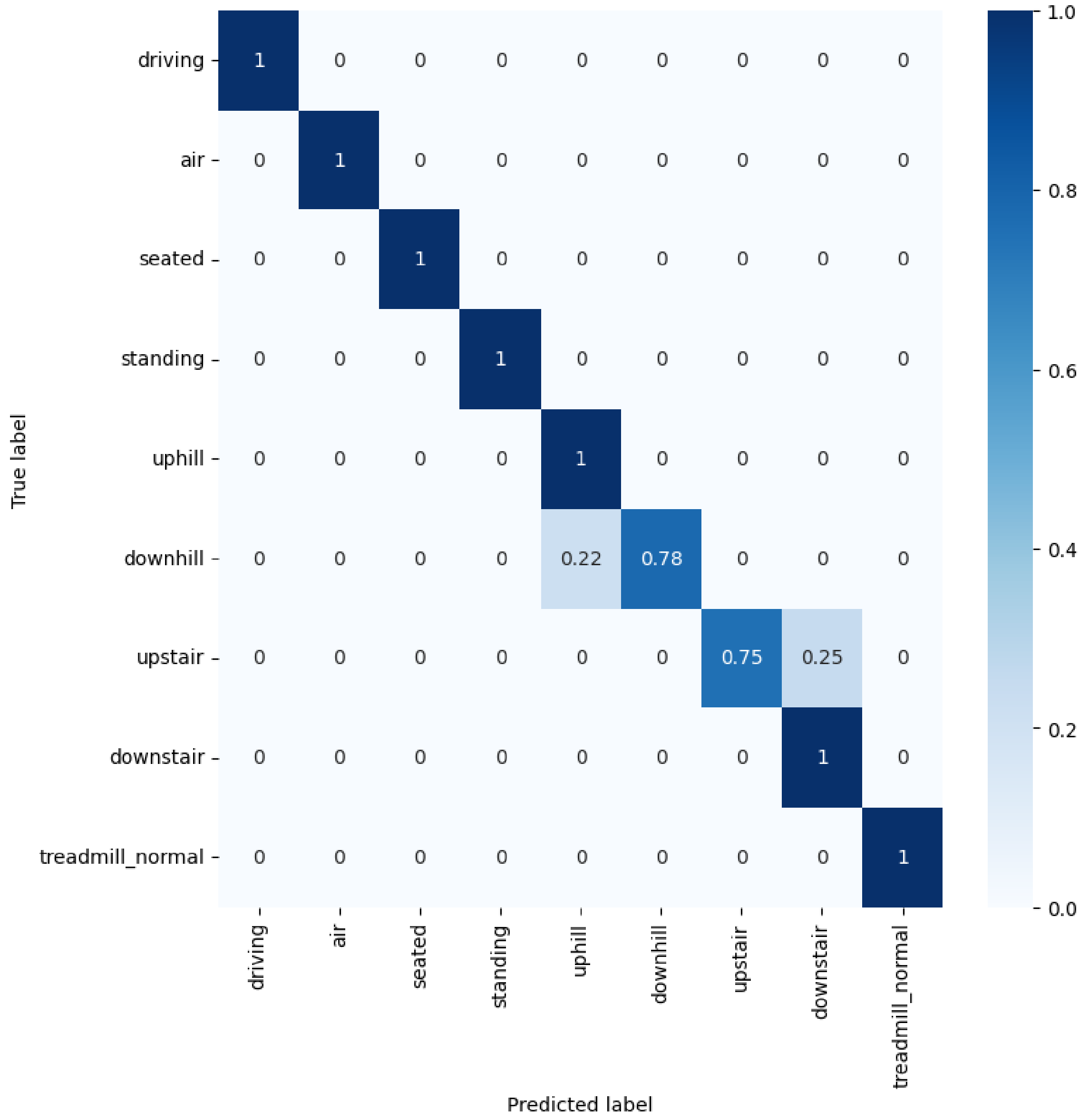
```
Validatation data:
5/5 - 0s - loss: 0.0973 - accuracy: 0.9708 - 237ms/epoch - 47ms/step
0.9708029031753540
Test data:
5/5 - 0s - loss: 0.1703 - accuracy: 0.9704 - 28ms/epoch - 6ms/step
0.9703703522682190
```

컨퓨전 메트릭스

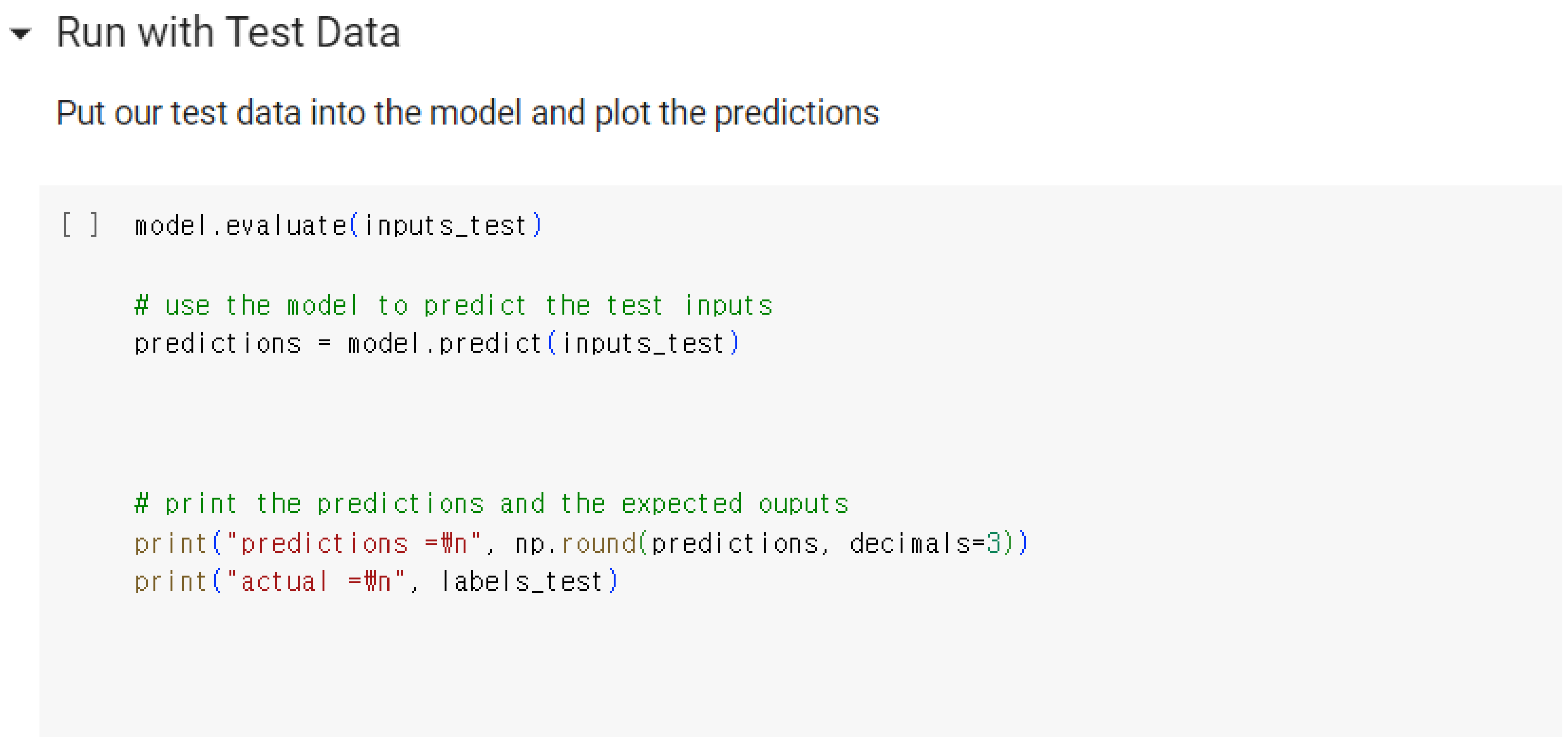


```
y_true = labels_test

predict_x = model.predict(inputs_test)
y_pred = np.argmax(predict_x, axis=1)


#classes=[0,1,2,3,4,5,6,7]
classes = GESTURES

con_mat = tf.math.confusion_matrix(labels=y_true, predictions=y_pred).numpy()
con_mat_norm = np.around(con_mat.astype('float') / con_mat.sum(axis=1)[:, np.newaxis], decimals=2)

con_mat_df = pd.DataFrame(con_mat_norm,
                          index = classes,
                          columns = classes)

figure = plt.figure(figsize=(8, 8))
sns.heatmap(con_mat_df, annot=True,cmap=plt.cm.Blues)
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
```

결과


True label
Predicted label
driving
air
seated
standing
uphill
downhill
upstair
downstair
treadmill_normal
driving 1 0 0 0 0 0 0 0 0
air 0 1 0 0 0 0 0 0 0
seated 0 0 1 0 0 0 0 0 0
standing 0 0 0 1 0 0 0 0 0
uphill 0 0 0 0 1 0 0 0 0
downhill 0 0 0 0 0.22 0.78 0 0 0
upstair 0 0 0 0 0 0 0.75 0.25 0
downstair 0 0 0 0 0 0 0 1 0
treadmill_normal 0 0 0 0 0 0 0 0 1
1.0
0.8
0.6
0.4
0.2
0.0

실제 테스트



Run with Test Data

Put our test data into the model and plot the predictions

```
[ ]  model.evaluate(inputs_test)

     # use the model to predict the test inputs
     predictions = model.predict(inputs_test)



     # print the predictions and the expected ouputs
     print("predictions =₩n", np.round(predictions, decimals=3))
     print("actual =₩n", labels_test)
```

코드와 함께 실습 해보기