Signal Classification

목차

1. Train Dataset으로 특징 추출 CSV 만들기 (with Python)

1.1 Signal의 Train 데이터

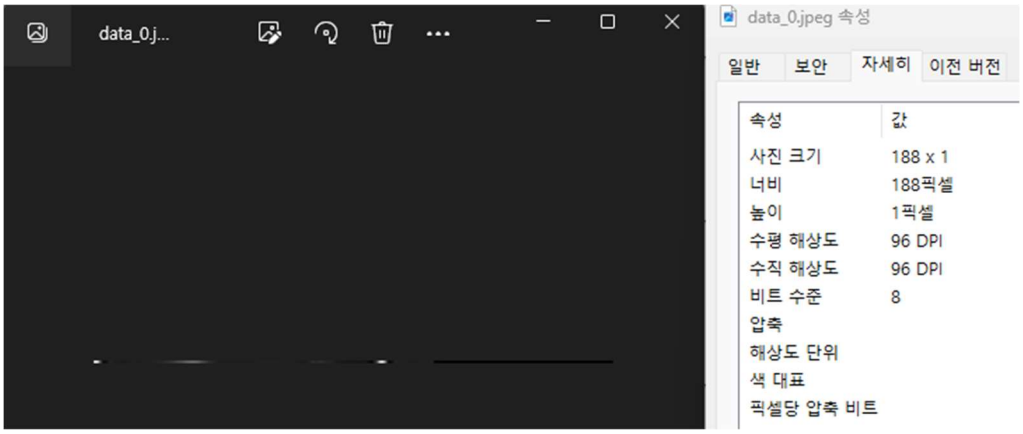

그림 1. Signal의 Train 데이터 [0으로 분류]

그림 1은 Signal의 Train 데이터이다. 이 이미지는 188 pixel로 188 x 1의 크기를 가지고 있다. 이 보고서에서는 0 또는 1로 분류되는 이미지를 갖고 각 이미지의 특징을 추출하고 이를 통해 모델을 만드는 것을 목표로 한다.

1.2 Signal 이미지의 특징 추출

이 보고서에서는 Python을 이용하여 이미지 픽셀의 특징을 추출할 예정이다. 각 이미지는 흑백의 픽셀을 갖고 있기 때문에 cv2.IMREAD_GRAYSCALE을 이용하여 이미지를 받아온다. 이후 이 이미지의 각 픽셀에 대한 값을 0~255까지의 값들로 구분한다.

1.3 Signal 이미지 특징을 이용한 CSV파일

픽셀 값이 분류가 되면 한 행에 이미지 정보를 저장한다. 한 행에 1번~188번 픽셀까지의 값(0~255)들을 저장하면 하나의 이미지가 저장된 것이다. Signal Train Dataset은 0, 1로 분류되는데 10185개의 레코드가 생성된다.

10163	1	229	91	71	3	0	15	17	21	20	23	27	29	30	33	40	45	40	47	55	58	59	62	68	74
10164	1	254	222	179	158	167	135	134	126	126	120	121	120	116	125	131	124	120	115	116	122	121	112	109	113
10165	1	254	205	122	82	64	38	34	35	31	28	26	26	27	29	29	28	28	30	33	35	37	39	42	44
10166	1	229	186	129	88	72	69	64	58	60	57	57	60	60	57	57	60	58	58	58	57	55	52	49	47
10167	1	249	127	39	28	24	13	12	11	14	12	10	10	12	13	13	12	14	13	14	18	18	16	19	25
10168	1	250	169	93	24	76	79	62	40	39	37	37	38	41	43	43	52	45	57	57	61	71	69	83	
10169	1	221	143	35	0	6	0	2	1	1	3	5	6	6	8	11	14	13	16	20	24	28	33	39	43
10170	1	238	81	28	15	25	26	25	25	25	25	25	24	24	23	23	23	23	23	23	24	26	28	30	31
10171	1	255	126	67	12	0	49	54	53	48	48	56	49	61	42	50	62	49	59	57	51	66	45	56	51
10172	1	255	158	42	29	19	11	2	5	1	1	1	2	2	3	3	3	4	5	6	7	7	7	7	6
10173	1	255	179	92	44	29	25	25	29	29	34	24	33	27	42	34	33	35	40	43	44	48	53	54	51
10174	1	230	253	148	82	59	41	21	21	20	17	15	14	15	16	16	15	22	18	19	24	29	30	31	34
10175	1	231	253	223	145	88	68	59	54	59	46	33	32	34	33	30	29	33	34	31	40	33	28	39	27
10176	1	255	186	112	76	61	44	30	27	29	27	26	25	24	22	17	14	13	16	20	23	25	27	29	32
10177	1	254	152	42	61	53	30	29	29	28	33	26	35	23	33	28	34	36	27	36	34	34	37	29	38
10178	1	255	151	108	60	45	30	19	20	18	19	20	20	18	18	19	20	19	19	19	18	18	17	17	17
10179	1	255	173	107	41	61	70	45	39	33	34	32	27	27	33	36	35	35	40	45	47	47	51	59	67
10180	1	255	136	59	0	20	46	49	51	48	49	45	56	41	58	54	49	56	53	54	60	63	60	59	60
10181	1	255	163	0	20	58	86	91	89	95	93	94	98	101	101	103	108	110	113	105	131	115	118	128	132
10182	1	235	143	55	28	32	25	17	17	15	15	15	16	16	17	17	17	16	18	20	20	20	22	25	27
10183	1	255	176	110	42	27	8	3	9	6	6	6	7	8	11	13	15	18	24	32	37	39	42	46	50
10184	1	255	201	111	84	62	64	65	64	59	65	67	65	64	67	68	66	67	69	65	73	65	60	74	63
10185	1	255	226	148	64	67	53	40	33	33	25	30	34	28	28	35	35	32	27	30	32	24	24	27	20

2. Train Dataset을 이용한 모델 학습과 생성

2.1 학습 파라미터 설정

이미지 데이터들을 Keras API를 사용하여 순차적으로 모델을 생성한다. 이 모델에서는 3개의 레이어를 가지고 있다. 먼저 입력 레이어로 이미지당 188개의 픽셀로 이루어진 1차원 배열을 사용한다. 입력 레이어에서 128개의 유닛을 사용하고 활성화 함수로 ReLU 함수를 사용하여 비선형성을 추가하고 Neural Network가 비선형적 패턴을 학습할 수 있도록 지원한다. 은닉 레이어는 입력 레이어보다 적은 64개의 유닛을 사용하고 활성화 함수로 ReLU 함수를 사용하여 비선형적 패턴을 학습한다. 마지막으로 출력 레이어에서는 0 또는 1로 분류하기 위해 1개의 유닛을 갖고 있으며 활성화 함수로 시그모이드 함수를 사용하여 확률 값을 출력한다.

$$ReLU\ function = \begin{cases}(x < 0),\ f(x) = 0\\(x \geq 0),\ f(x) = x\end{cases} \qquad Sigmoid\ function = \frac{1}{1+e^{-x}}$$

epochs = 10

batch_size = 32

2.2 CSV 파일을 이용한 모델 학습 및 생성

이 모델을 생성하기 위해 1에서 만들어 둔 CSV파일을 열어 학습 데이터를 받아온다. 이 데이터를 pandas를 통해 데이터프레임으로 저장하고 특성으로 사용하는 픽셀 데이터(0~255)와 레이블을 분할한다. 학습을 위해 픽셀 데이터를 255.0으로 나누어 줌으로써 0~1의 값을 갖도록 하여 정규화 한다. 학습을 진행하기 위해 이 데이터들을 훈련 데이터와 테스트 데이터로 나누고 순차적으로 입력, 은닉, 출력 레이어를 거쳐 학습을 진행한다. 학습을 수행하는 최적화 알고리즘으로 Adam을 사용하고 손실 함수로 binary_crossentropy를 사용한다.

2.3 모델 평가

모델을 평가하는 기준으로 정확도를 선택하여 테스트 정확도를 출력하여 확인한다. 모델 학습과 생성을 동시에 진행하도록 코드를 작성하였다. 1st epoch에 0.9583의 accuracy와 0.1112의 loss를 가지고 있었다. 이는 epoch를 거칠수록 증가와 감소함을 보이고 있다.

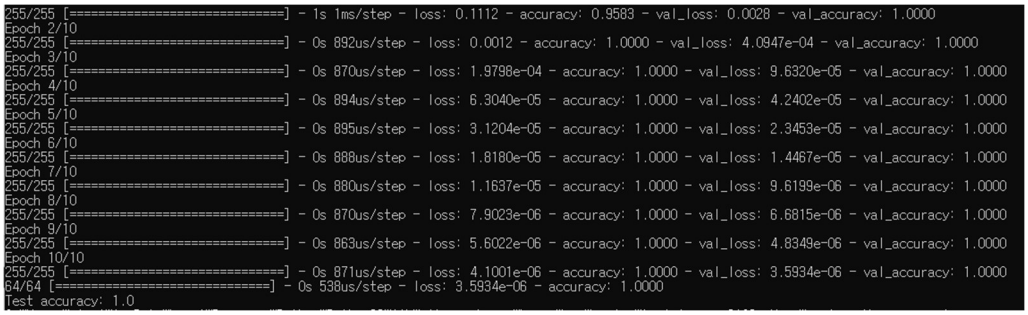

그림 2. 모델 학습 및 평가

3. Test Dataset으로 성능 측정 및 모델 분석

3.1 Test Dataset을 이용한 성능 측정

2에서 생성된 모델을 통해 Test Dataset의 이미지들을 분류한다. 모델을 학습하면서 이미 train data와 test data를 나누어 학습을 시켰지만 새로운 Test Dataset의 이미지를 제대로 인식하고 분류할 수 있는 지 확인하고 이 또한 정확도를 출력해본다.

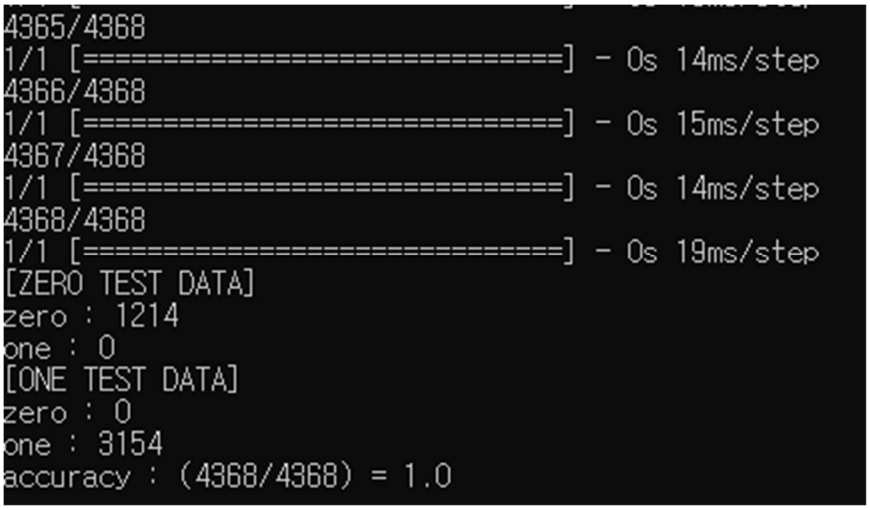

그림 3. Test Dataset 분류

3.2 모델 분석

CSV 파일을 불러와 픽셀 데이터와 레이블로 분할하고 픽셀 데이터를 0~255 사이의 값으로 정규화한다. 이 데이터를 학습과 테스트 데이터로 분할하여 훈련에 사용한다. Tensorflow의 Sequential 모델을 이용하여 순차적으로 신경망 모델을 생성한다. 입력, 은닉, 출력 레이어가 있는데 각각 128, 64, 1개의 유닛이 있다. 출력 레이어의 1개의 유닛은 시그모이드 함수에 의해 이진 분류에 사용한다. 이 모델의 손실 함수로는 binary_crossentropy, 옵티마이저로는 Adam을 활용해 모델을 컴파일한다. 10번의 epoch 동안 32의 batch_size를 갖고 학습을 진행한다. 이렇게 생성된 모델은 모델 평과와 함께 "signal_classifier.h5"의 이름을 갖고 저장된다.

4. 코드 첨부

4.1 Signal image to CSV file (Train Dataset)

```python
import cv2
import numpy as np
import csv
import os

folder1_path = 'D:/Signal/train\\0'
folder2_path = 'D:/Signal/train\\1'
csv_file = 'image_data_with_labels.csv'

def process_images_to_csv(image_folder, label):
    with open(csv_file, 'a', newline='') as csvfile:
        writer = csv.writer(csvfile)
        for filename in os.listdir(image_folder):
            img_path = os.path.join(image_folder, filename)
            image = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
            resized_image = cv2.resize(image, (188, 1))
            row = np.concatenate(([label], resized_image.flatten()))
            writer.writerow(row)

with open(csv_file, 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['label'] + [f"pixel_{i}" for i in range(188)])

process_images_to_csv(folder1_path, 0)
process_images_to_csv(folder2_path, 1)
```

4.2 Model Train (Train Dataset)

```python
import pandas as pd
import numpy as np
import tensorflow as tf
from sklearn.model_selection import train_test_split

file_path = 'image_data_with_labels.csv'
data = pd.read_csv(file_path)

X = data.iloc[:, 1:].values
y = data.iloc[:, 0].values

X = X / 255.0

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, activation='relu', input_shape=(188,)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))

loss, accuracy = model.evaluate(X_test, y_test)
print(f'Test accuracy: {accuracy}')

model.save('signal_classifier.h5')
```

4.3 Model Evaluation (Test Dataset)

```python
import cv2
import numpy as np
import tensorflow as tf

def preprocess_image(image_path):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    resized_image = cv2.resize(image, (188, 1))
    return resized_image.flatten() / 255.0

i = 2832
zero, one = 0, 0
model = tf.keras.models.load_model('signal_classifier.h5')

zero_test_size = 1214
one_test_size = 3154
correct = 0
incorrect = 0

while i < 4046:
    print(str(i-2831) + "/" + str(one_test_size + zero_test_size))
    test_image_path = "D:/Signal/test/\\0\data_" + str(i) + ".jpeg"
    preprocessed_image = preprocess_image(test_image_path)
    prediction = model.predict(np.array([preprocessed_image]))

    if prediction[0] >= 0.5:
        one = one + 1
    else:
        zero = zero + 1

    i = i + 1

str_res0 = "[ZERO TEST DATA]\nzero : " + str(zero) + "\none : " + str(one)
correct = zero
incorrect = one

i = 7352
zero, one = 0, 0

while i < 10506:
    print(str(i-7351+1214) + "/" + str(one_test_size + zero_test_size))
    test_image_path = "D:/Signal/test/\\1\data_" + str(i) + ".jpeg"
    preprocessed_image = preprocess_image(test_image_path)
    prediction = model.predict(np.array([preprocessed_image]))

    if prediction[0] >= 0.5:
        one = one + 1
    else:
        zero = zero + 1

    i = i + 1

str_res1 = "[ONE TEST DATA]\nzero : " + str(zero) + "\none : " + str(one)
correct = correct + one
incorrect = incorrect + zero
print(str_res0)
print(str_res1)
print("accuracy : (" + str(correct) + "/" + str(one_test_size + zero_test_size) + ") = "
      + str(correct/(one_test_size + zero_test_size)))
```