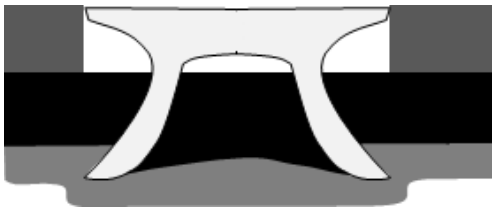
Portland State
UNIVERSITY

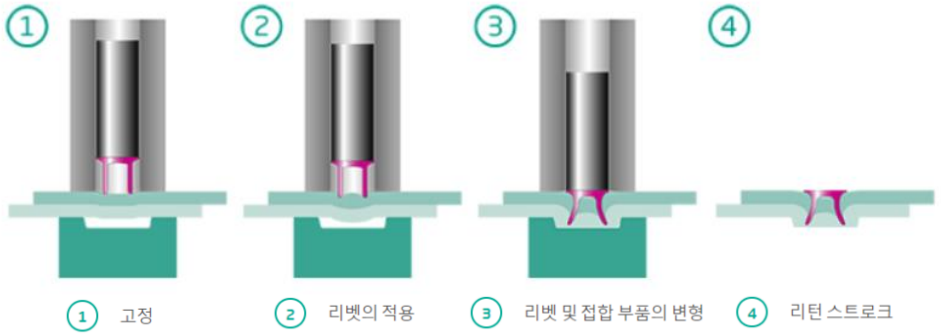
멀티센서 input을 이용한 SPR 접합특성 분류

0. 풀어야 할 문제

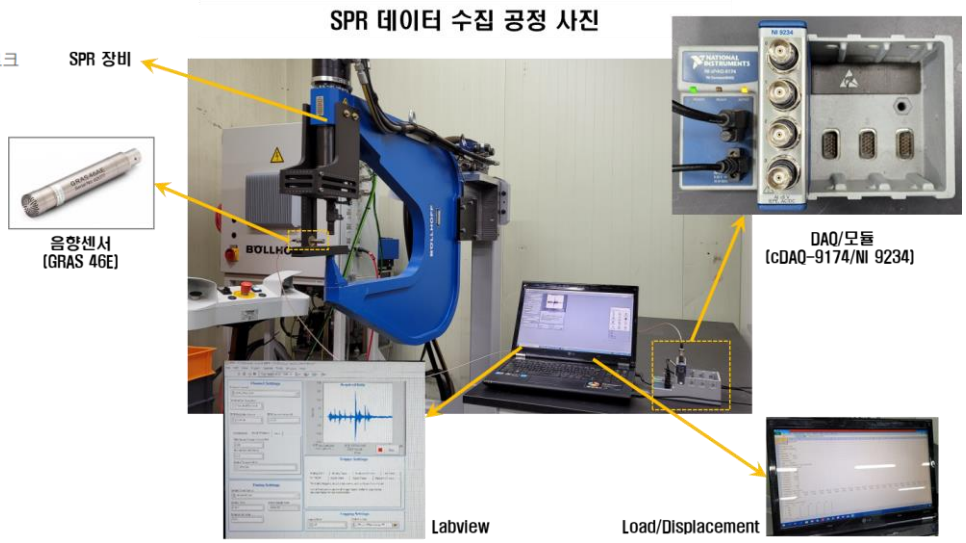
- 소재: 상판 - 1.2GPa 급 steel (1.4 mm)
  중판 - 열경화성 CFRP (1.8 mm)
  하판 – 590 MPa 급 steel (1.0 mm)



- 용접방법: 셀프 피어싱 리벳(Self-Piercing Rivet)

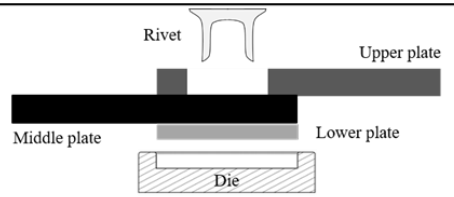
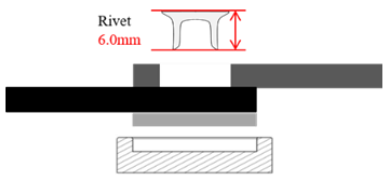
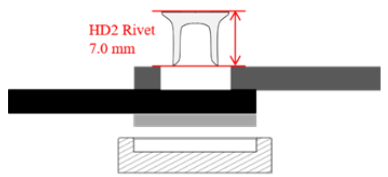
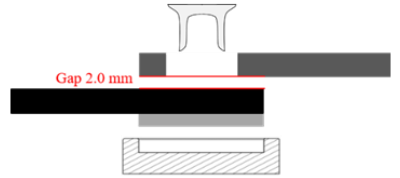
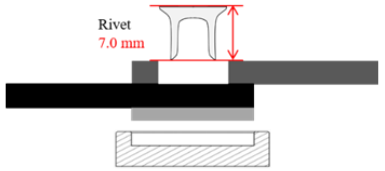
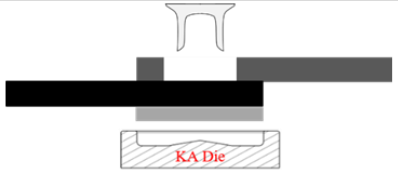
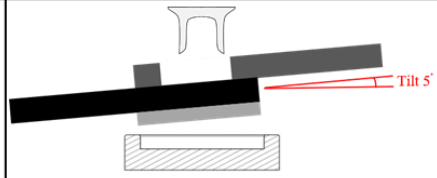
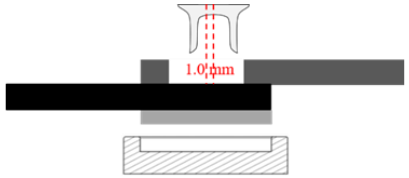
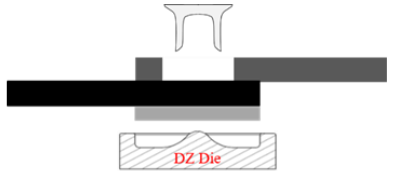
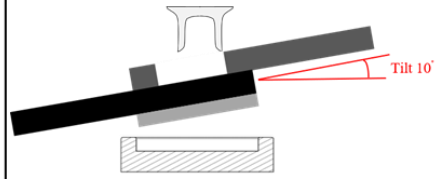
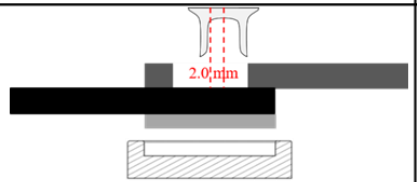
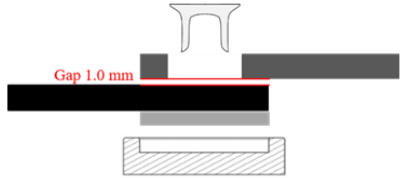
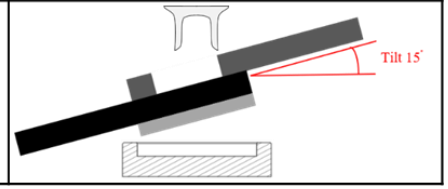


- 공정 중에 발생하는 load, displacement, acoustic data 수집



0. 풀어야 할 문제

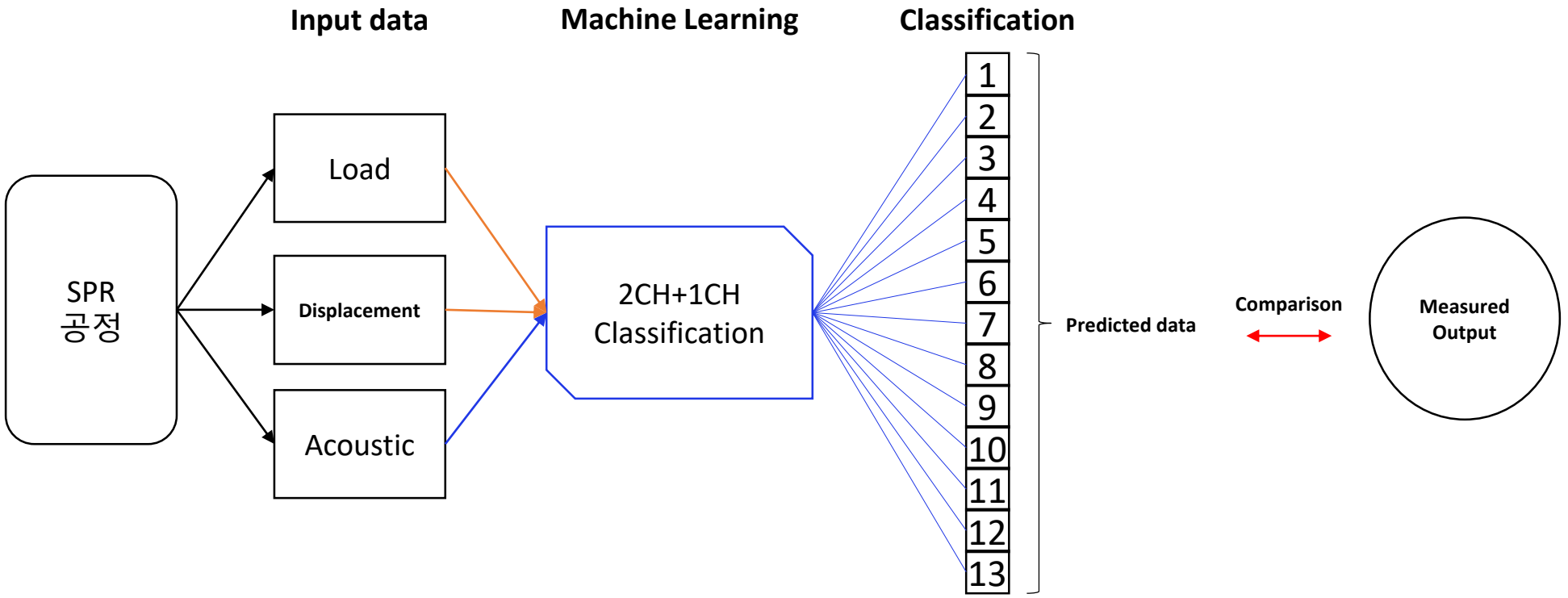
- SPR 단면 형상에 따라서 13가지의 case로 나누었으며, 기준에 맞춰서 데이터 분류

Reference schematic (Case 1)					
					
Case	**schematic**	**Case**	**schematic**	**Case**	**schematic**
2		6		10	
3		7		11	
4		8		12	
5		9		13	

1. 머신 러닝 모델 구축에 사용된 Input, Output parameter

Machine learning을 이용한 학습 순서

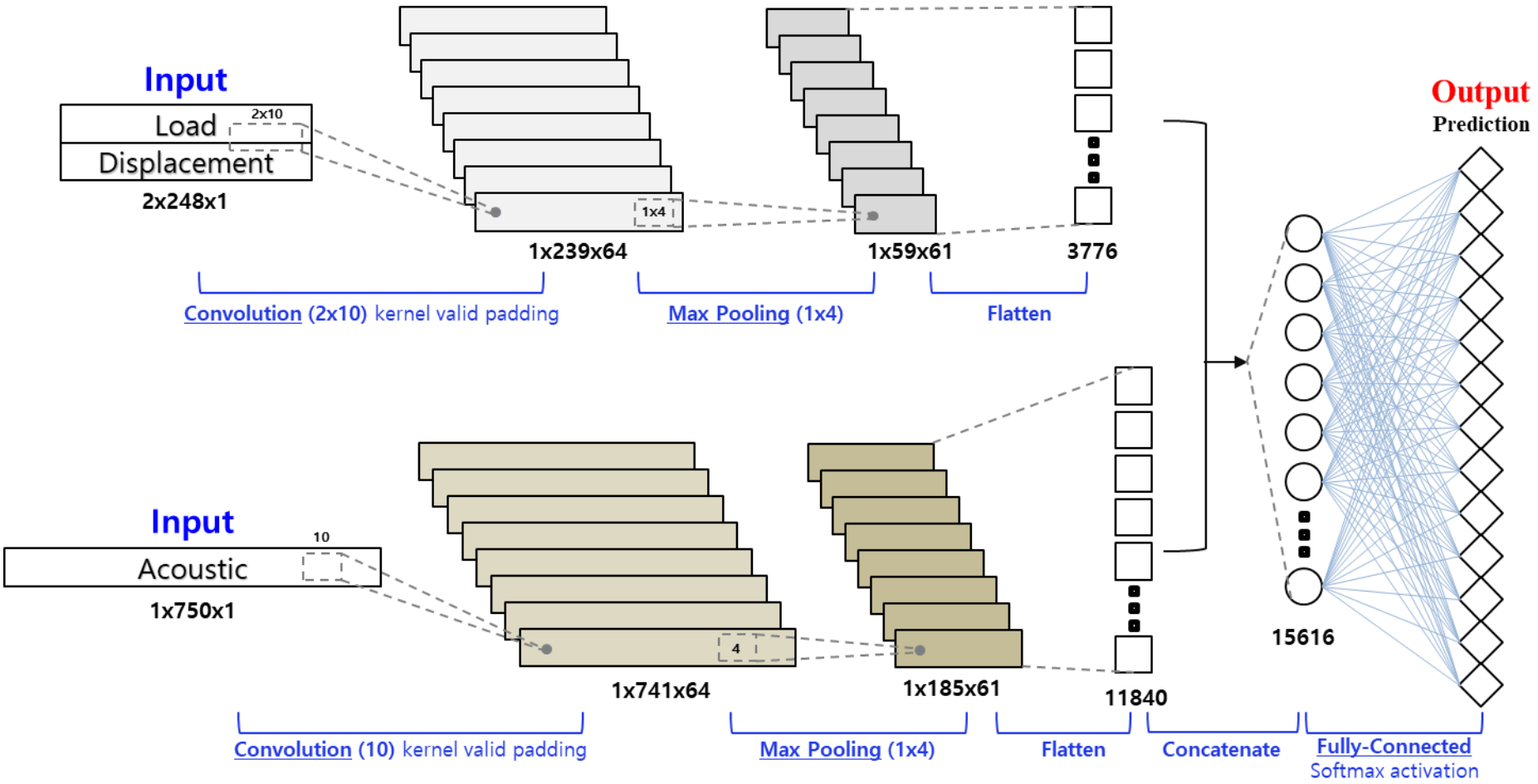
- 2CH + 1CH Classification : Load, Displacement + Acoustic data를 각각 CNN 처리 후, 합친 다음 분류작업



2. SPR 2CH+1CH Classification

2CH + 1CH CNN structure

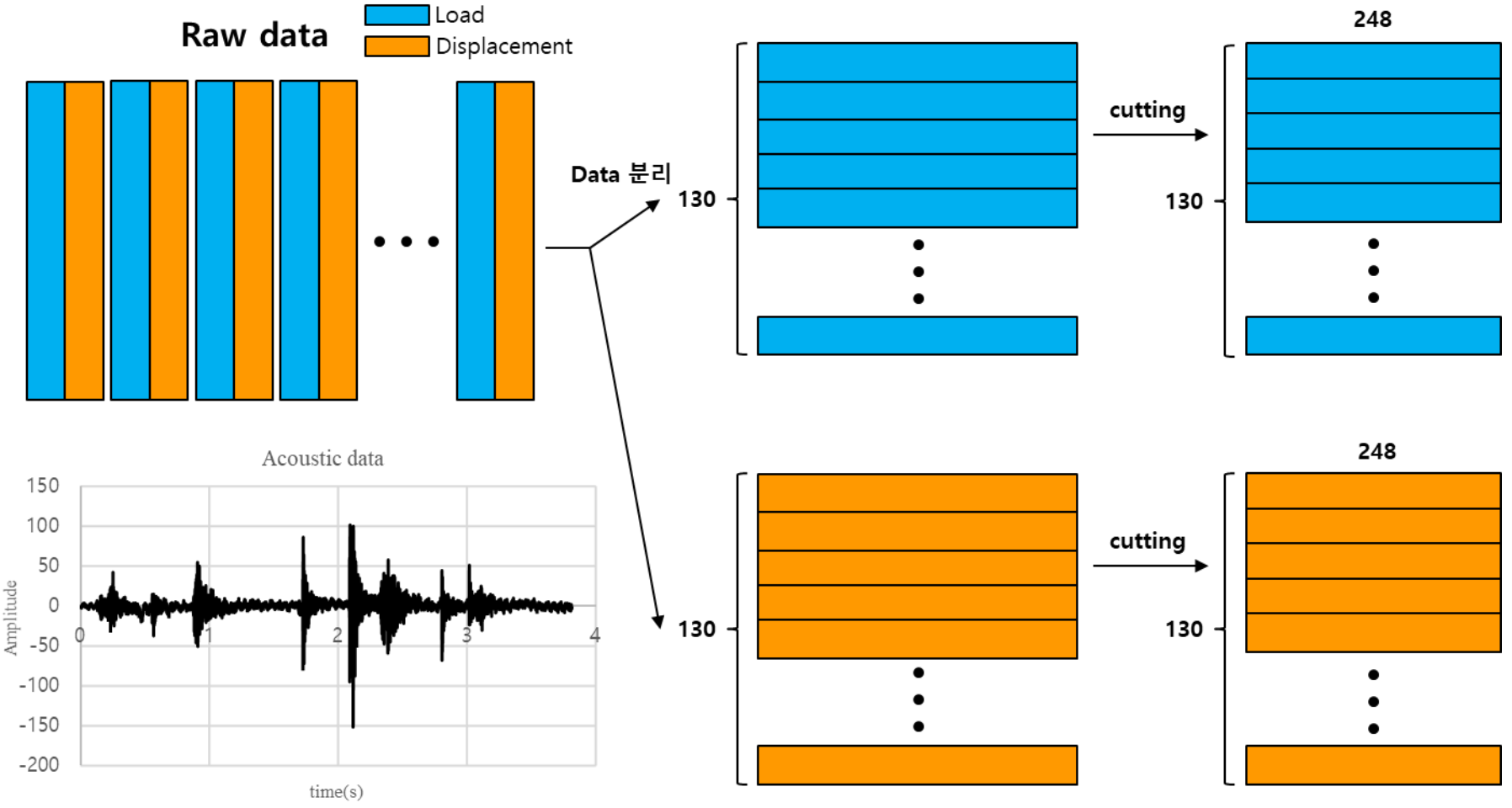
- SPR 데이터 중 Load, Displacement와 Acoustic를 흑백 이미지로 생각하여 CNN을 통해 classification을 통한 결과 예측을 위한 CNN structure 구성



2. SPR 2CH+1CH Classification

Preprocessing

- SPR data 중 Load / Displacement는 한 장비에서 동시에 나오기 때문에 코드에서 사용할 수 있게 preprocessing 작업 필요
- Load와 Displacement를 각각 나누어서 한 파일로 정리 후 data의 size를 맞추기 위하여 자름 (248x130)



2. SPR 2CH+1CH Classification

Coding

- SPR 데이터 중 Load, Displacement와 Acoustic 신호를 이용하여 모델 별 분류예측을 하기 위한 Code 작성
- Acoustic 데이터의 양이 Load, Displacement의 데이터 양에 비해 과다하여 Down sampling 필요

```
#Displacement data loading
_Displacement_path = "./Data/displacement13.csv"
Displacement_original_Data = np.loadtxt(_Displacement_path, delimiter=',', dtype=np.float32, encoding='utf-8')

#Load data loading
_Load_path = "./Data/load13.csv"
Load_original_Data = np.loadtxt(_Load_path, delimiter=',', dtype=np.float32, encoding='utf-8')

#Acoustic data loading - 15000개 데이터 중 7500개만 사용
_Acoustic_path = "./Data/Acoustic.csv"
Acoustic_original_Data = np.loadtxt(_Acoustic_path, delimiter=',', dtype=np.float32, encoding='utf-8')

Acoustic_original_Data = Acoustic_original_Data[:,:7500]
```

Input 데이터 불러오기

```
#Output data loading
_Class_result_path = "./Data/Class_result.csv"
Class_result_original_Data = np.loadtxt(_Class_result_path, delimiter=',', dtype=np.int32, encoding='utf-8')
```

Output 데이터 불러오기

```
#데이터 표준화
Displacement_original_Data -= Displacement_original_Data.mean(axis=0)
Displacement_original_Data /= Displacement_original_Data.std(axis=0)

Load_original_Data -= Load_original_Data.mean(axis=0)
Load_original_Data /= Load_original_Data.std(axis=0)

Acoustic_original_Data -= Acoustic_original_Data.mean(axis=0)
Acoustic_original_Data /= Acoustic_original_Data.std(axis=0)

Displacement_norm_Data = Displacement_original_Data
Load_norm_Data = Load_original_Data
Acoustic_norm_Data = Acoustic_original_Data
```

데이터 정규화

2. SPR 2CH+1CH Classification

Coding

- 데이터 preprocessing 작업 (행렬 변환, 흑백 이미지화, down sampling)
- Acoustic 데이터 down sampling시 10개의 평균 값만 취해서 1/10의 크기로 줄임

```
#데이터 합치기 1x248 2개의 데이터를 이미지 처럼 2x248 데이터로
#Initializing
Input_data = []

#Merging each line
for i in range(Data_amount):

    Input_data.append(np.stack([Displacement_norm_Data[i,:], Load_norm_Data[i,:]]))

Input_data = np.array(Input_data)
print(Input_data.shape)
```

흑백 이미지화

```
#Acoustic data 다운 샘플링 10개 평균 7500개 -> 750개 데이터

Input_data2 = np.zeros((130,750))

for i in range(130):
    for j in range(750):
        Input_data2[i][j] = np.mean(Acoustic_norm_Data[i][j*10:j*10+10])
```

Acoustic 데이터 downsampling

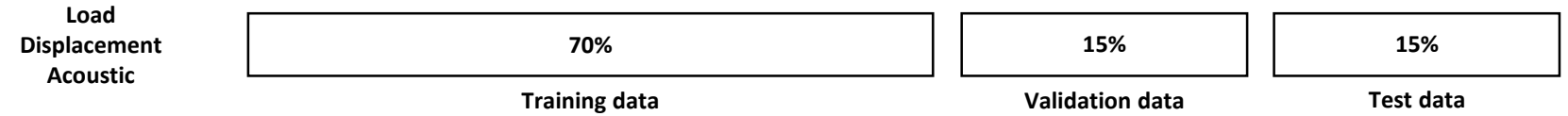
```
#CNN을 위한 입력 데이터 리사이징
Input_data = Input_data.reshape((130,2,248,1))
Input_data2 = Input_data2.reshape((130,750,1))
```

데이터 리사이징

2. SPR 2CH+1CH Classification

Coding

- 검증을 하기위해 각 Input 데이터를 70:15:15로 나눔(training, validation, test)




```
#입력 데이터 70:15:15로 나누기(Training, validation, test)
#Displacement and Load

# split data: Training data를 70%
x_train, x_test, y_train, y_test = train_test_split(Input_data, Output_data,
train_size=0.70, random_state=42)

# split data: Validation과 Test data를 50:50 으로
x_val, x_test, y_val, y_test = train_test_split(x_test, y_test, train_size=0.50, random_state=42)
```

Load, Displacement 데이터 나눔

```
#Acoustic

# split data: Training data를 70%
x2_train, x2_test, y_train, y_test = train_test_split(Input_data2, Output_data,
train_size=0.70, random_state=42)

# split data: Validation과 Test data를 50:50 으로
x2_val, x2_test, y_val, y_test = train_test_split(x2_test, y_test, train_size=0.50, random_state=42)
```

Acoustic 데이터 나눔

2. SPR 2CH+1CH Classification

Coding

- 도식화한 CNN structure을 토대로 코드 작성
- 적절한 활성화 함수 설정 (ReLU, Softmax)

```
#모델 구조 설계 및 컴파일러 설정

#CNN structure

X1 = Input(shape=(2, 248, 1), dtype='float32')
X2 = Input(shape=(750, 1), dtype='float32')

Model_1_1 = layers.Conv2D(64, (2, 10), activation='relu', input_shape=(2, 248, 1))(X1)
Model_1_2 = layers.MaxPooling2D((1, 4))(Model_1_1)
Model_1_3 = layers.Flatten()(Model_1_2)

Model_2_1 = layers.Conv1D(64, 10, activation='relu', input_shape=(750, 1))(X2)
Model_2_2 = layers.MaxPooling1D((4))(Model_2_1)
Model_2_3 = layers.Flatten()(Model_2_2)

concatenated = layers.concatenate([Model_1_3, Model_2_3], axis=-1)

#FCN structure & 모델 컴파일
answer = layers.Dense(13, activation='softmax')(concatenated)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model = Model([X1, X2], answer)

model.summary()

#학습 시작history = model.fit([x_train, x2_train], y_train, epochs=100, batch_size=5, validation_data=([x_val, x2_val],
y_val))
```

Load+ Displacement, Acoustic 각각 CNN 작업

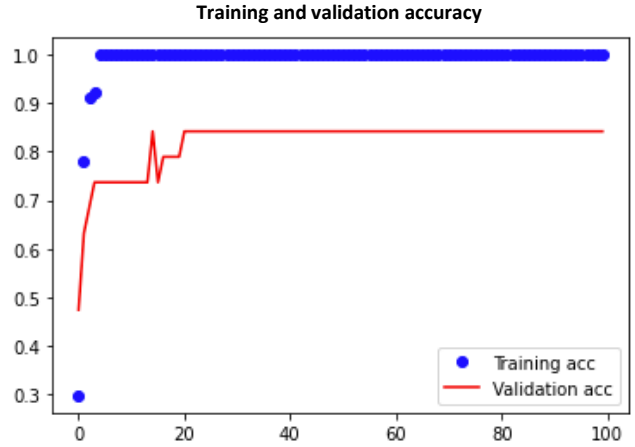
CNN 완료된 데이터 FCN 작업

2. SPR 2CH+1CH Classification

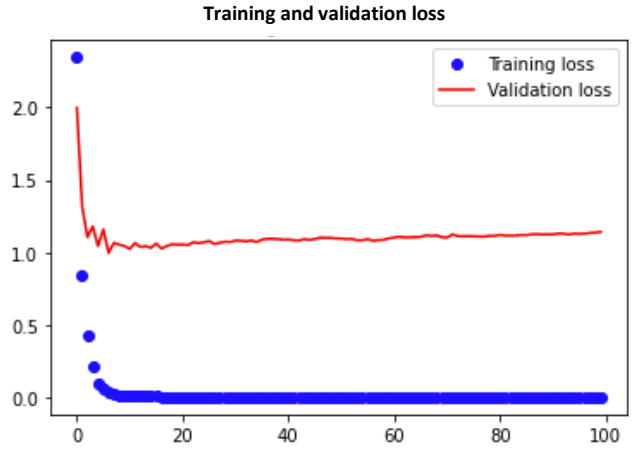
Coding

- 정확도와 손실을 파악하기 위한 그래프 plot
- 그래프를 통해 Epoch를 반복할 수록 정확도는 증가하고, loss는 감소하는 경향을 확인함

```
#train, validation accuracy graph ploting
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'r', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
```




```
#train and validation loss graph ploting
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()

plt.show()
```



2. SPR 2CH+1CH Classification

Model Prediction

- SPR 데이터 중 Load, Displacement 데이터 + Acoustic 데이터를 사용하여 case 별 분류예측을 실시
- Train 데이터로 예측했을 경우 모두 정확히 예측됨
- Validation 데이터로 예측했을 경우 3가지 case에서 예측이 맞지 않은 것을 확인함

◆ Train data Matrix

Prediction

Target

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	8	0	0	0	0	0	0	0	0	0	0	0	0
2	0	4	0	0	0	0	0	0	0	0	0	0	0
3	0	0	7	0	0	0	0	0	0	0	0	0	0
4	0	0	0	8	0	0	0	0	0	0	0	0	0
5	0	0	0	0	6	0	0	0	0	0	0	0	0
6	0	0	0	0	0	7	0	0	0	0	0	0	0
7	0	0	0	0	0	0	7	0	0	0	0	0	0
8	0	0	0	0	0	0	0	7	0	0	0	0	0
9	0	0	0	0	0	0	0	0	6	0	0	0	0
10	0	0	0	0	0	0	0	0	0	7	0	0	0
11	0	0	0	0	0	0	0	0	0	0	8	0	0
12	0	0	0	0	0	0	0	0	0	0	0	8	0
13	0	0	0	0	0	0	0	0	0	0	0	0	8

◆ Validation data Matrix

Prediction

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	4	0	0	0	0	0	0	1	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0	0
5	0	0	0	0	3	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	1	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	3	0	0	0
11	0	0	0	0	0	0	0	0	0	0	1	0	0
12	0	0	0	0	0	0	0	0	0	0	0	1	0
13	0	0	0	0	0	0	0	0	0	0	0	1	0

감 사 합 니 다