지능화 캡스톤 프로젝트

프로젝트 #1 결과 발표

2024. 04. 21

충북대학교 산업인공지능학과

[4조] 최현동, 이찬희, 사수진

프로젝트 수행체계

수행방법

- 웨이퍼맵 데이터 분석하여, 부족한 부분 데이터 증강 및 전처리를 통해 충분한 학습 데이터를 확보하고, 논문의 CNN-WDI를 구현하고, 학습시켜 결과를 보고자 합니다.
- 업무 분장을 통해 프로젝트를 진행하고 있습니다.
- 소통은 카카오톡으로 진행하며, 수요일마다 만남을 가져 진행을 하고 있습니다.

업무분장

이름	수행내용	비고
최현동	• 코딩 • 심층 컨볼루션 신경망(CNN-WDI) 설계	
이찬희	• 데이터 학습 및 발표자료 작성 • 결과 발표	
사수진	• 데이터 증강 • 데이터 전처리	

데이터셋(진행흐름도)

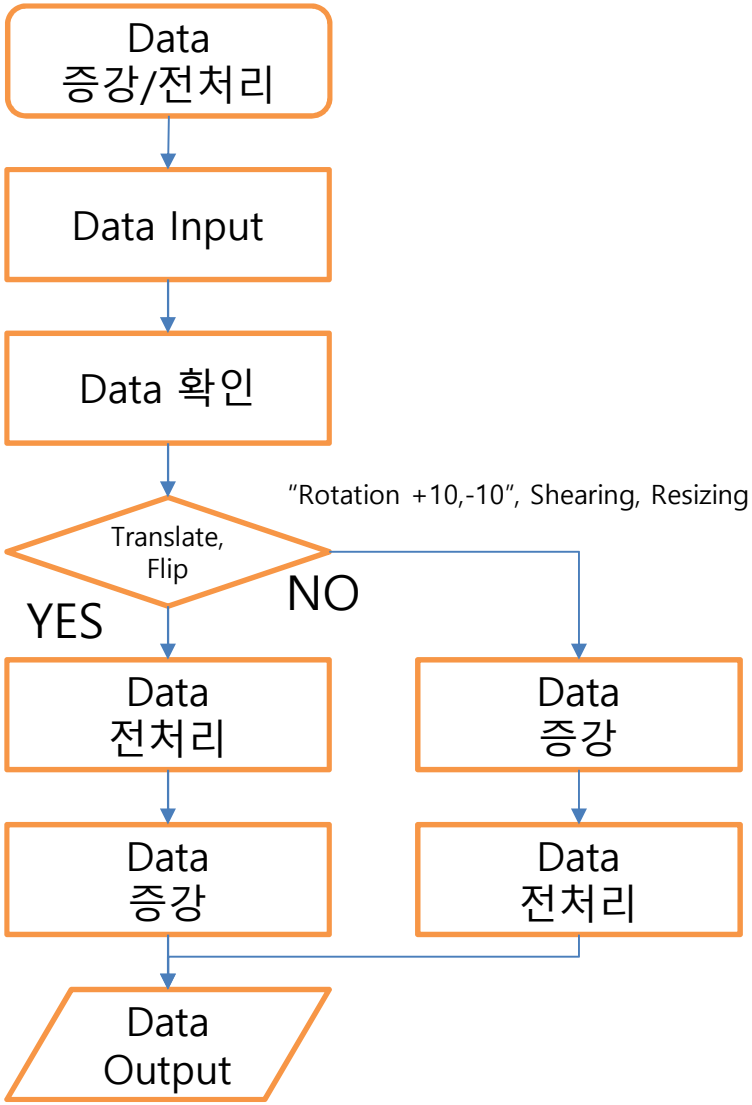
1. 데이터 셋
- Kaggle (wm811k-wafer-map)

2. 데이터 확인
- Total = 811,457, Label(o) = 172,950, Label(x) = 638,507
- Test data = 118,595, Training data = 54,355

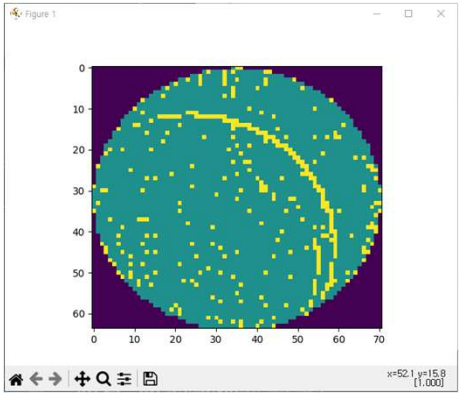
3. 데이터 증강 및 데이터 전처리
- Rotation +10 (증강 후 전처리)
- Rotation -10 (증강 후 전처리)
- Shearing (증강 후 전처리)
- Resizing (증강 후 전처리)
- Translate (전처리 후 증강)
- Flip (전처리 후 증강)

※ 전처리 = (224,224) Zero-padding

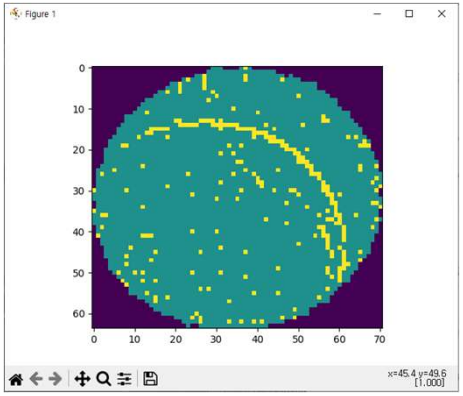
Type	With Label Data	Train Data	Total/Train Data(%)
0(center)	4,294	3,462	80%
1(Donut)	555	409	73%
2 (Edge-loc)	5,189	2,417	46%
3 (Edge-ring)	9,680	8,554	88%
4(loc)	3,593	1,620	45%
5(random)	866	609	70%
6(scratch)	1,193	500	41%
7(near-full)	149	54	36%
8(none)	147,431	36,730	24%



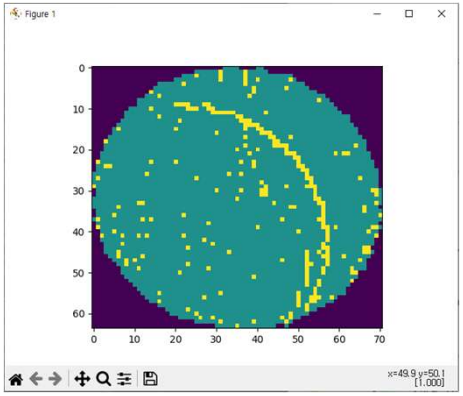
데이터셋(증강 작업)



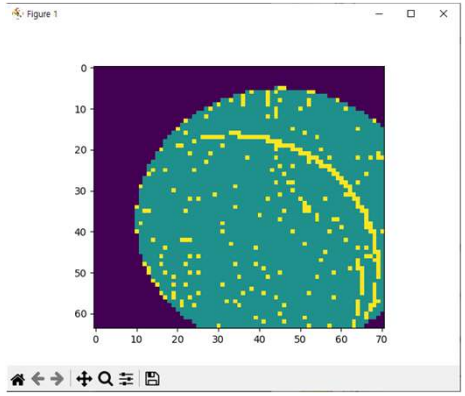
원본 데이터



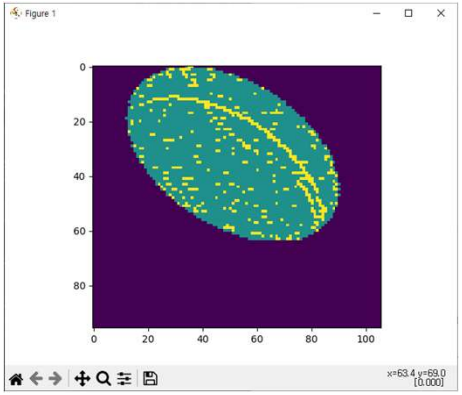
Rotation 10도



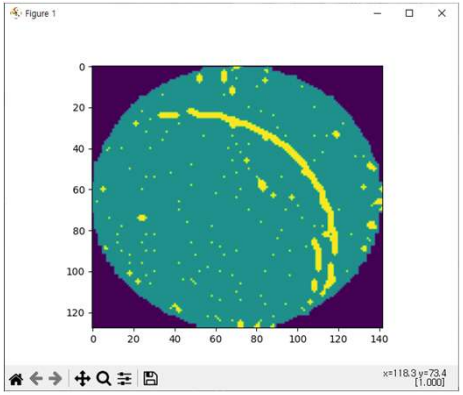
Rotation -10도



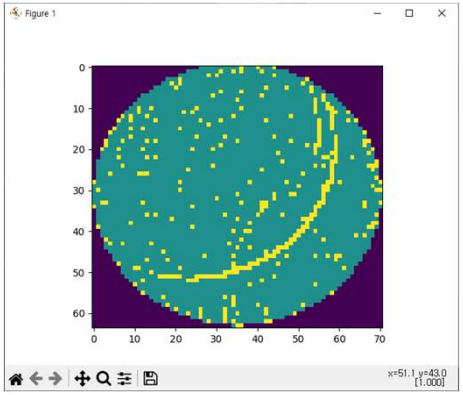
Translate



shearing



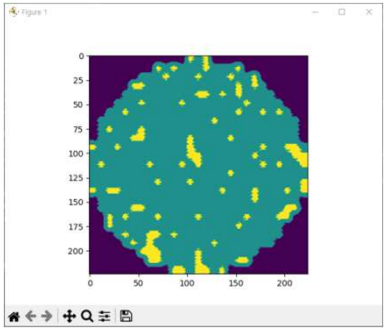
Zoom



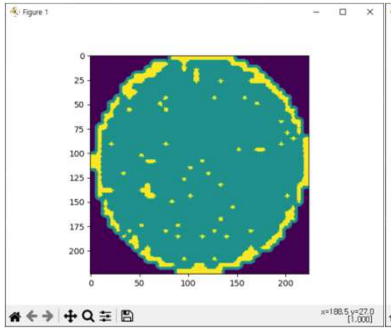
filp

* 예시를 위한 이미지로 padding작업 없이 진행하였습니다.

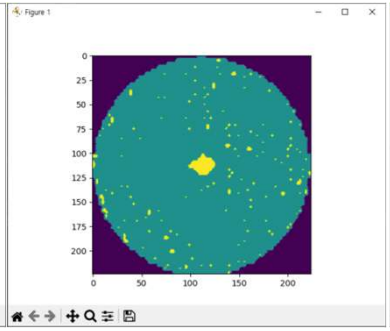
데이터셋(학습 전 이미지)



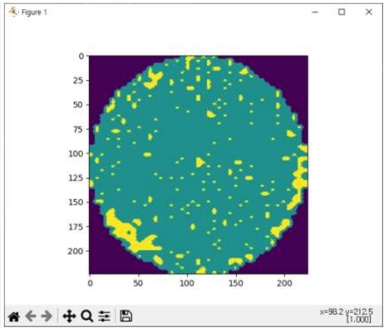
None



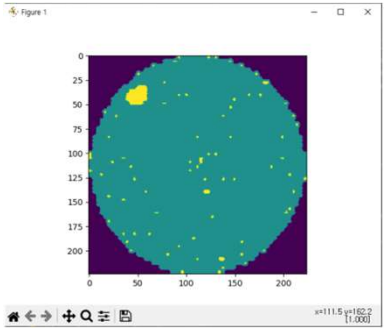
Edge-ring



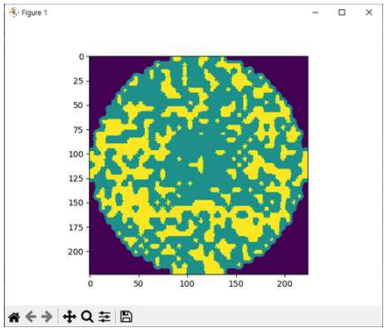
Center



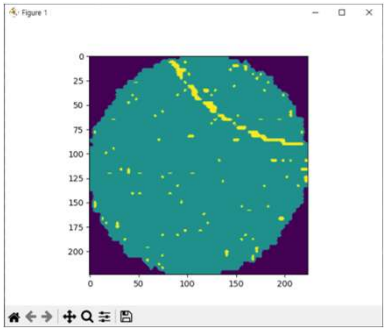
Edge_loc



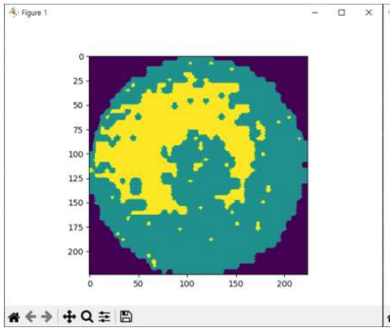
Loc



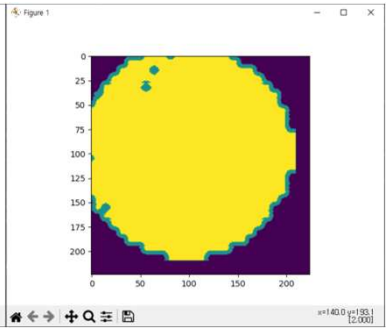
Random



Scratch



Donut

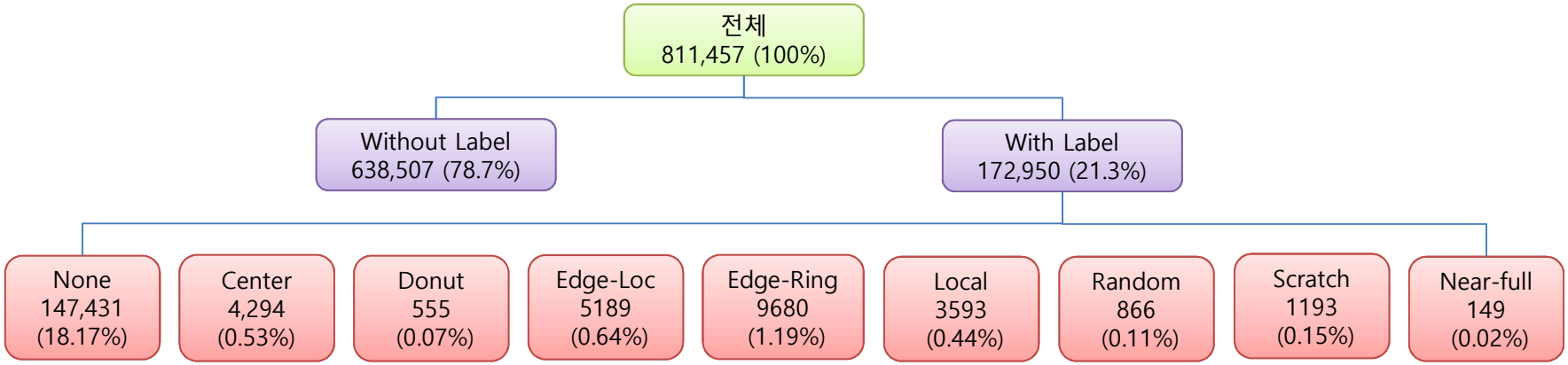


Near-full

데이터셋

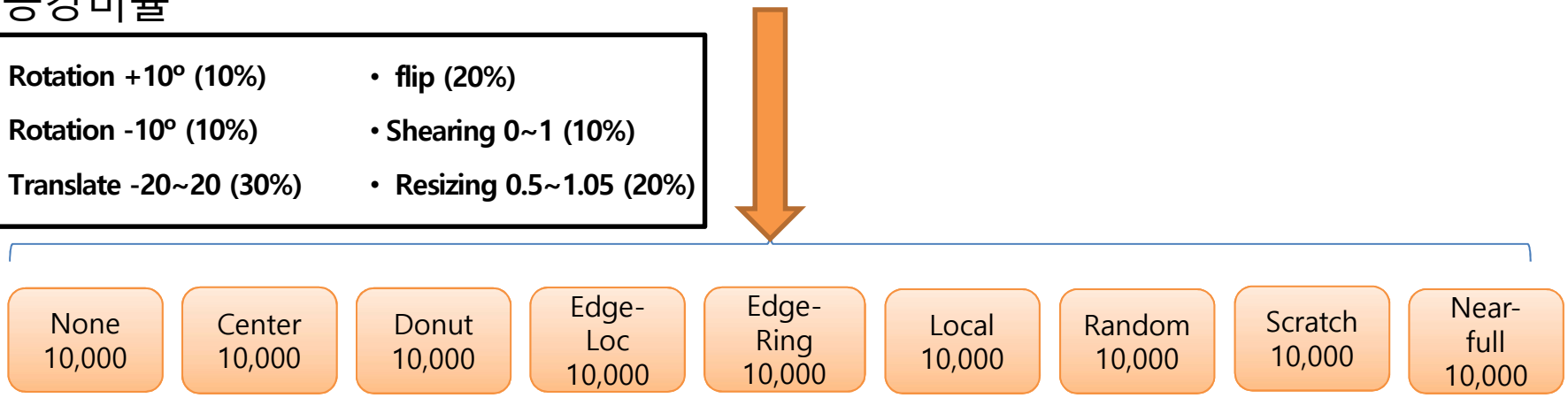
데이터 구성

wm811k-wafer-map Dataset



*증강비율

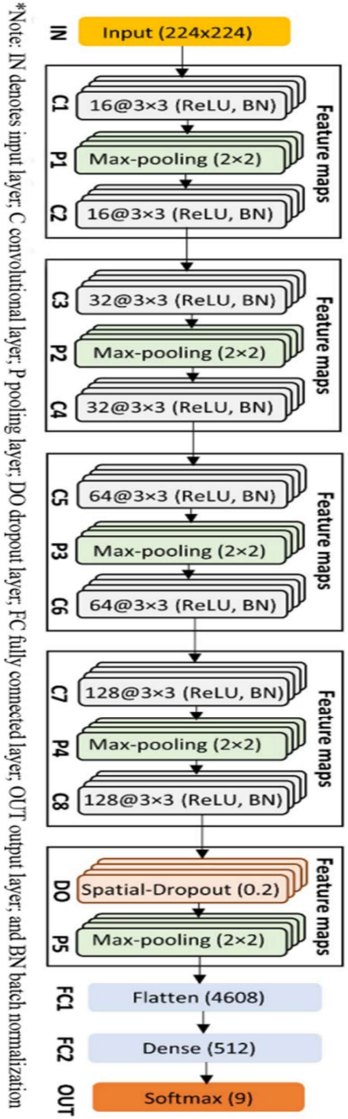
- **Rotation +10° (10%)**
- **Rotation -10° (10%)**
- **Translate -20~20 (30%)**
- **flip (20%)**
- **Shearing 0~1 (10%)**
- **Resizing 0.5~1.05 (20%)**



Train : Validation : Test = 65 : 20 : 15

CNN 구조

CNN 구조



◆ 구현 WDI구조

```
input_shape = (224,224,1)
class_num = 9
KERNEL_SIZE = 3

input_layer = layers.Input(shape=input_shape) # Input 224x224
x = layers.Conv2D(16, kernel_size=KERNEL_SIZE, padding='valid')(input_layer) # 16, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization
x = layers.MaxPool2D(pool_size=(2,2))(x) # Max-pooling
x = layers.Conv2D(16, kernel_size=KERNEL_SIZE, padding='same')(x) # 16, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization

x = layers.Conv2D(32, kernel_size=KERNEL_SIZE, padding='same')(x) # 32, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization
x = layers.MaxPool2D(pool_size=(2,2))(x) # Max-pooling
x = layers.Conv2D(32, kernel_size=KERNEL_SIZE, padding='same')(x) # 32, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization

x = layers.Conv2D(64, kernel_size=KERNEL_SIZE, padding='same')(x) # 64, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization
x = layers.MaxPool2D(pool_size=(2,2))(x) # Max-pooling
x = layers.Conv2D(64, kernel_size=KERNEL_SIZE, padding='same')(x) # 64, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization
x = layers.Conv2D(128, kernel_size=KERNEL_SIZE, padding='same')(x) # 128, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization
x = layers.MaxPool2D(pool_size=(2,2))(x) # Max-pooling
x = layers.Conv2D(128, kernel_size=KERNEL_SIZE, padding='same')(x) # 128, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization

x = layers.SpatialDropout2D(0.2)(x) # Spartial Dropout 0.2
x = layers.MaxPool2D(pool_size=(2,2))(x) # Max-pooling
x = layers.Flatten()(x) # 4608 확인
x = layers.Dense(512, activation='relu')(x)
output_layer = layers.Dense(class_num, activation='softmax')(x)

CNN_WDI = Model(input_layer, output_layer)

CNN_WDI.summary()
```

CNN 구조

주요 코드

```python
import numpy as np
import pandas as pd
import cv2
import matplotlib.pyplot as plt
import random
import pickle
import datetime
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers
from tensorflow.keras import optimizers
from tensorflow.keras.models import Model
from tensorflow.keras.callbacks import ModelCheckpoint
```

라이브러리, 모듈

```python
def rotation_10_degree(data_img):
    height, width = data_img.shape
    rotation_10_degree_img = cv2.getRotationMatrix2D( center: (width/2, height/2), angle: 10, scale: 1)
    dst = cv2.warpAffine(data_img, rotation_10_degree_img, dsize: (width,height))
    padded_dst = zero_padding(dst, set_size: 224)
    return padded_dst

def rotation_minus_10_degree(data_img):
    height, width = data_img.shape
    rotation_10_degree_img = cv2.getRotationMatrix2D( center: (width/2, height/2), -10, scale: 1)
    dst = cv2.warpAffine(data_img, rotation_10_degree_img, dsize: (width,height))
    padded_dst = zero_padding(dst, set_size: 224)
    return padded_dst

def translate(data_img):
    padded_img = zero_padding(data_img, set_size: 224)
    x_translate = random.randrange(-20, stop: 21)
    y_translate = random.randrange(-20, stop: 21)
    translate_matrix = np.float32([[1, 0, y_translate],
                                   [0, 1, x_translate]])
    dst = cv2.warpAffine(padded_img, translate_matrix, dsize: (224,224))
    return dst

def filping(data_img):
    padded_img = zero_padding(data_img, set_size: 224)
    dst = cv2.flip(padded_img, random.choice([0, 1]))
    return dst

def shearing(data_img):
    x_shearing = random.random()
    y_shearing = random.random()
    shearing_matrix = np.float32([[1,          x_shearing, 0],
                                  [y_shearing, 1,          0],
                                  [0,          0  ,        1]])
    dst = cv2.warpPerspective(data_img, shearing_matrix, dsize: (224, 224))
    return dst

def shearing(data_img):
    x_shearing = random.random()
    y_shearing = random.random()
    shearing_matrix = np.float32([[1,          x_shearing, 0],
                                  [y_shearing, 1,          0],
                                  [0,          0  ,        1]])
    dst = cv2.warpPerspective(data_img, shearing_matrix, dsize: (224, 224))
    return dst

def resizing(data_img):
    scale_list = [0.5, 0.6, 0.7, 0.8, 1.05]
    dst = cv2.resize(data_img, dsize=(0,0), fx=random.choice(scale_list), fy=random.choice(scale_list),
                     interpolation=cv2.INTER_LINEAR)
    padded_img = zero_padding(dst, set_size: 224)
    return padded_img
```

데이터 증강 함수

CNN 구조 (증강구현)

주요 코드 및 실행 결과

```
# None(8): 36,730 중 10,000 개 Sampling
# 참고 : https://rfriend.tistory.com/602
wm811k_new_train_class_None = wm811k_train.query("failureNum == 8").sample(n=10000, random_state=2022)
wm811k_new_train_class_None["waferMap_augmentation"] = wm811k_new_train_class_None["waferMap"].apply(lambda x: zero_padding(x, set_size: 224))
wm811k_new_train_class_None['waferMap_augmentation_Dim']=wm811k_new_train_class_None['waferMap_augmentation'].apply(find_dim)
wm811k_new_train_class_None.info()

wm811k_new_train_class_None['waferMap_augmentation_Dim'].apply(lambda x: str(x)).value_counts()

# Edge-Ring(3) 현 보유 8,554개 / 1,446개 augmentation 필요
wm811k_new_train_class_Edge_Ring = wm811k_train.query("failureNum == 3")
wm811k_new_train_class_Edge_Ring["waferMap_augmentation"] = wm811k_new_train_class_Edge_Ring['waferMap'].apply(lambda x: zero_padding(x, set_size: 224))
wm811k_new_train_class_Edge_Ring['waferMap_augmentation_Dim']=wm811k_new_train_class_Edge_Ring['waferMap_augmentation'].apply(find_dim)
wm811k_new_train_class_Edge_Ring.info()

# +10도 회전 : 10% - 144개
wm811k_new_train_class_Edge_Ring_10 = wm811k_new_train_class_Edge_Ring.sample(n=144, random_state=2022)
wm811k_new_train_class_Edge_Ring_10["waferMap_augmentation"] = wm811k_new_train_class_Edge_Ring_10["waferMap"].apply(lambda x: rotation_10_degree(x))
wm811k_new_train_class_Edge_Ring_10['waferMap_augmentation_Dim']=wm811k_new_train_class_Edge_Ring_10['waferMap_augmentation'].apply(find_dim)

# -10도 회전 : 10% - 144개
wm811k_new_train_class_Edge_Ring_minus_10 = wm811k_new_train_class_Edge_Ring.sample(n=144, random_state=2022)
wm811k_new_train_class_Edge_Ring_minus_10["waferMap_augmentation"] = wm811k_new_train_class_Edge_Ring_minus_10["waferMap"].apply(lambda x: rotation_minus_10_degree(x))
wm811k_new_train_class_Edge_Ring_minus_10['waferMap_augmentation_Dim']=wm811k_new_train_class_Edge_Ring_minus_10['waferMap_augmentation'].apply(find_dim)

# 좌우 대칭 : 20% - 288개
wm811k_new_train_class_Edge_Ring_flip = wm811k_new_train_class_Edge_Ring.sample(n=288, random_state=2022)
wm811k_new_train_class_Edge_Ring_flip["waferMap_augmentation"] = wm811k_new_train_class_Edge_Ring_flip["waferMap"].apply(lambda x: filping(x))
wm811k_new_train_class_Edge_Ring_flip['waferMap_augmentation_Dim']=wm811k_new_train_class_Edge_Ring_flip['waferMap_augmentation'].apply(find_dim)
# 평행 이동 : 30% - 432개
wm811k_new_train_class_Edge_Ring_translate = wm811k_new_train_class_Edge_Ring.sample(n=432, random_state=2022)
wm811k_new_train_class_Edge_Ring_translate["waferMap_augmentation"] = wm811k_new_train_class_Edge_Ring_translate["waferMap"].apply(lambda x: translate(x))
wm811k_new_train_class_Edge_Ring_translate['waferMap_augmentation_Dim']=wm811k_new_train_class_Edge_Ring_translate['waferMap_augmentation'].apply(find_dim)

# 전단 범위(shearing range) : 10% - 144개
wm811k_new_train_class_Edge_Ring_shearing = wm811k_new_train_class_Edge_Ring.sample(n=144, random_state=2022)
wm811k_new_train_class_Edge_Ring_shearing["waferMap_augmentation"] = wm811k_new_train_class_Edge_Ring_shearing["waferMap"].apply(lambda x: shearing(x))
wm811k_new_train_class_Edge_Ring_shearing['waferMap_augmentation_Dim']=wm811k_new_train_class_Edge_Ring_shearing['waferMap_augmentation'].apply(find_dim)

# 확대 : 20% - 294개
wm811k_new_train_class_Edge_Ring_resize = wm811k_new_train_class_Edge_Ring.sample(n=294, random_state=2022)
wm811k_new_train_class_Edge_Ring_resize["waferMap_augmentation"] = wm811k_new_train_class_Edge_Ring_resize["waferMap"].apply(lambda x: resizing(x))
wm811k_new_train_class_Edge_Ring_resize['waferMap_augmentation_Dim']=wm811k_new_train_class_Edge_Ring_resize['waferMap_augmentation'].apply(find_dim)

# concat
wm811k_new_train_class_Edge_Ring_augmentation = pd.concat([wm811k_new_train_class_Edge_Ring,
                                                           wm811k_new_train_class_Edge_Ring_10,
                                                           wm811k_new_train_class_Edge_Ring_minus_10,
                                                           wm811k_new_train_class_Edge_Ring_flip,
                                                           wm811k_new_train_class_Edge_Ring_translate,
                                                           wm811k_new_train_class_Edge_Ring_shearing,
                                                           wm811k_new_train_class_Edge_Ring_resize
                                                          ])
wm811k_new_train_class_Edge_Ring_augmentation.info()
```

증강 코드

```
<class 'pandas.core.frame.DataFrame'>
Index: 8554 entries, 100 to 786313
Data columns (total 10 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   waferMap                 8554 non-null   object
 1   dieSize                  8554 non-null   float64
 2   lotName                  8554 non-null   object
 3   trianTestLabel           8554 non-null   object
 4   failureType              8554 non-null   object
 5   waferMapDim              8554 non-null   object
 6   failureNum               8554 non-null   object
 7   trainTestNum             8554 non-null   object
 8   waferMap_augmentation    8554 non-null   object
 9   waferMap_augmentation_Dim  8554 non-null   object
dtypes: float64(1), object(9)
memory usage: 735.1+ KB
<class 'pandas.core.frame.DataFrame'>
Index: 10000 entries, 100 to 199660
Data columns (total 10 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   waferMap                 10000 non-null  object
 1   dieSize                  10000 non-null  float64
 2   lotName                  10000 non-null  object
 3   trianTestLabel           10000 non-null  object
 4   failureType              10000 non-null  object
 5   waferMapDim              10000 non-null  object
 6   failureNum               10000 non-null  object
 7   trainTestNum             10000 non-null  object
 8   waferMap_augmentation    10000 non-null  object
 9   waferMap_augmentation_Dim  10000 non-null  object
dtypes: float64(1), object(9)
```

실행 결과

CNN 구조 (신경망 모델)

주요 코드 및 실행 결과

```
input_shape = (224,224,1)
class_num = 9
KERNEL_SIZE = 3

input_layer = layers.Input(shape=input_shape) # Input 224x224
x = layers.Conv2D(16, kernel_size=KERNEL_SIZE, padding='valid')(input_layer) # 16, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization
x = layers.MaxPool2D(pool_size=(2,2))(x) # Max-pooling
x = layers.Conv2D(16, kernel_size=KERNEL_SIZE, padding='same')(x) # 16, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization

x = layers.Conv2D(32, kernel_size=KERNEL_SIZE, padding='same')(x) # 32, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization
x = layers.MaxPool2D(pool_size=(2,2))(x) # Max-pooling
x = layers.Conv2D(32, kernel_size=KERNEL_SIZE, padding='same')(x) # 32, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization

x = layers.Conv2D(64, kernel_size=KERNEL_SIZE, padding='same')(x) # 64, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization
x = layers.MaxPool2D(pool_size=(2,2))(x) # Max-pooling
x = layers.Conv2D(64, kernel_size=KERNEL_SIZE, padding='same')(x) # 64, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization
x = layers.Conv2D(128, kernel_size=KERNEL_SIZE, padding='same')(x) # 128, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization
x = layers.MaxPool2D(pool_size=(2,2))(x) # Max-pooling
x = layers.Conv2D(128, kernel_size=KERNEL_SIZE, padding='same')(x) # 128, 3x3
x = layers.Activation(activation='relu')(x) # ReLU
x = layers.BatchNormalization()(x) # BatchNormalization

x = layers.SpatialDropout2D(0.2)(x) # Spartial Dropout 0.2
x = layers.MaxPool2D(pool_size=(2,2))(x) # Max-pooling
x = layers.Flatten()(x) # 4608 확인
x = layers.Dense(512, activation='relu')(x)
output_layer = layers.Dense(class_num, activation='softmax')(x)

CNN_WDI = Model(input_layer, output_layer)

CNN_WDI.summary()
```

CNN-WDI 구현 코드

학습 방법

딥러닝 학습 조건

Hardware : 1) CPU : 11th Gen Intel® Core™ i7-11800H @ 2.3GHz (16 CPUs), ~2.3GHz

2) GPU : NVIDIA GeForce RTX 3060 Laptop GPU

3) Memory : 16GB RAM

4) TEST Size = 0.2

LEARNING_RATE = 0.001

BATCH_SIZE = 10

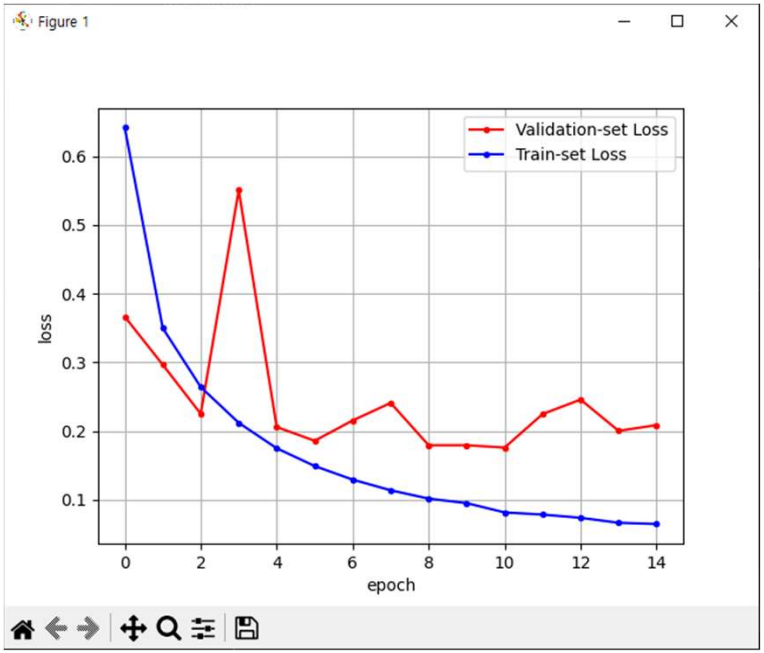
EPOCH = 15

5) 학습시간 : 약 2시간 55분

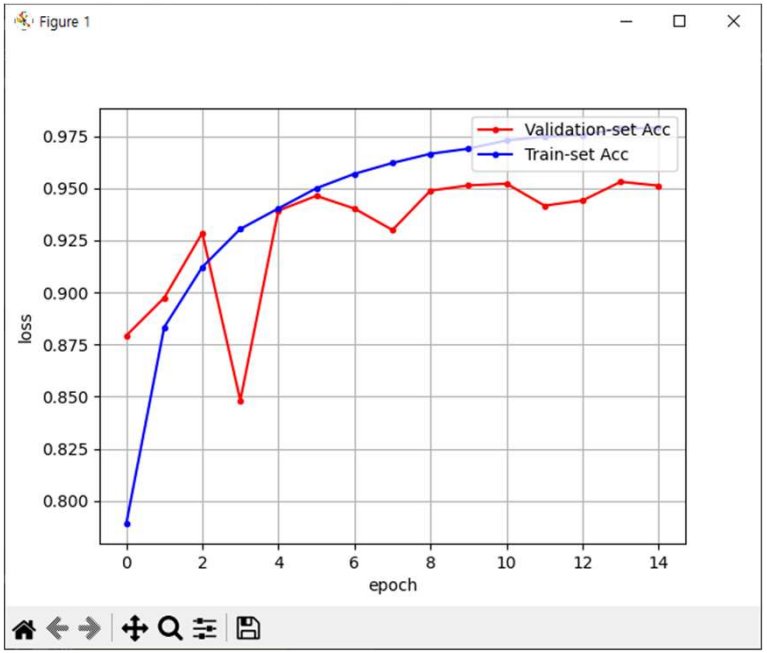
```
model_id = 'cnn_wdi'

LEARNING_RATE = 0.001
BATCH_SIZE = 10
EPOCH = 15

CNN_WDI.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                optimizer=optimizers.Adam(learning_rate=LEARNING_RATE),
                metrics=['acc'])

checkpointer = ModelCheckpoint(filepath="{0}.keras".format(model_id), verbose=1, save_best_only=True)

hist = CNN_WDI.fit(X_train, y_train, epochs=EPOCH, batch_size=BATCH_SIZE, shuffle=True, validation_data=(X_val, y_val), callbacks=[checkpointer])
```

모델의 성능

딥러닝 학습 곡선



Loss graph



Accuracy graph

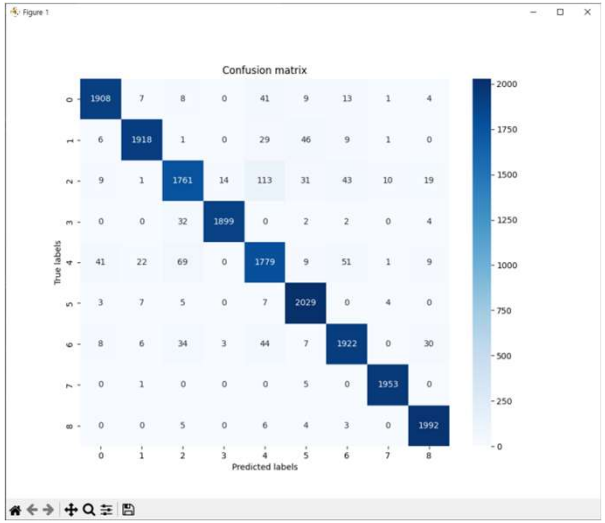
모델의 성능

딥러닝 학습 결과

PERFORMANCE EVALUATION OF CNN-WDI FOR BALANCED AND IMBLANCED DATASETS (%)

Defect Class	Balanced dataset			Imbalanced dataset		
	Precision	Recall	F1-score	Precision	Recall	F1-score
Center	93.6	**98.0**	95.7	**94.7**	97.7	**96.2**
Donut	**96.0**	**98.3**	**97.2**	85.2	90.4	87.7
Edge-Loc	**97.4**	**93.1**	**95.2**	89.2	87.4	88.3
Edge-Ring	94.4	91.7	93.0	**97.7**	**98.3**	**98.0**
Local	**91.6**	**90.3**	**90.9**	82.8	81.4	82.1
Near-Full	98.9	**99.9**	**99.4**	**100.0**	59.1	74.2
Random	**96.7**	**96.5**	**96.6**	83.0	90.0	86.4
Scratch	**98.1**	**98.7**	**98.4**	80.4	73.2	76.6
None	99.5	99.7	99.6	**99.9**	**100.0**	**100.0**
Average	**96.24**	**96.24**	**96.22**	90.32	86.39	87.72

논문 학습 결과



구현 Confusion matrix

Type
0(center)
1(Donut)
2(Edge-loc)
3(Edge-ring)
4(Local)
5(random)
6(scratch)
7(near-full)
8(none)

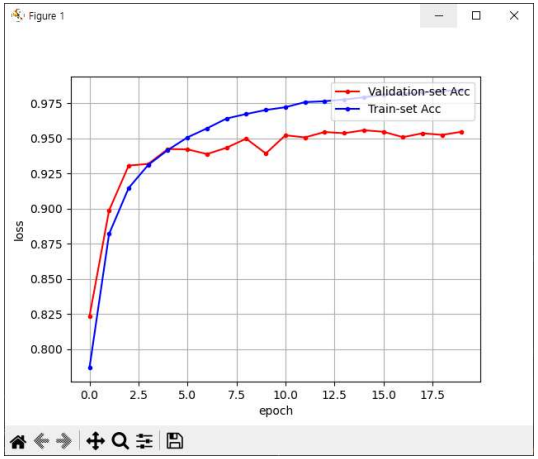
	precision	recall	f1-score	support
0	0.97	0.96	0.96	1991
1	0.98	0.95	0.97	2010
2	0.92	0.88	0.90	2001
3	0.99	0.98	0.99	1939
4	0.88	0.90	0.89	1981
5	0.95	0.99	0.97	2055
6	0.94	0.94	0.94	2054
7	0.99	1.00	0.99	1959
8	0.97	0.99	0.98	2010
accuracy			0.95	18000
macro avg	0.95	0.95	0.95	18000
weighted avg	0.95	0.95	0.95	18000

각 타입별 정밀도, 재현율, F1-score

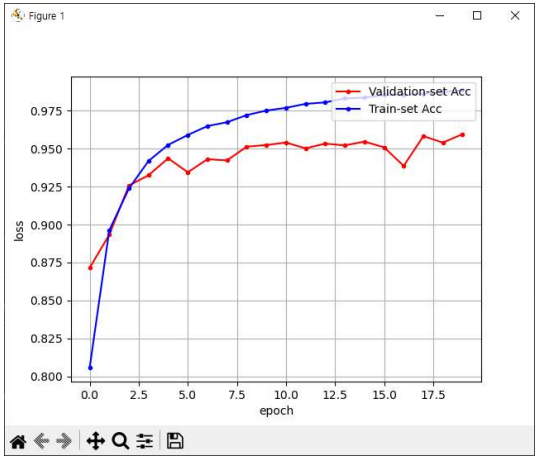
하이퍼파라미터 튜닝-Accuracy Graph

하이퍼 파라미터 튜닝 값.

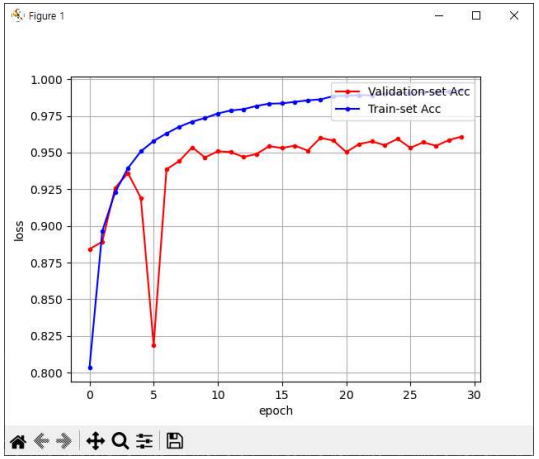
	LEARNING RATE	BATCH_SIZE	EPOCH
Setting 값1	0.01	10	20
Setting 값2	0.005	10	20
Setting 값3	0.005	10	30



Setting 값1



Setting 값2



Setting 값3

- Setting 값 1, 2번을 비교 해보면 Traing Acc값이 0.005일때 좀 더 학습이 되어 보이며, 20EPOCH에서 Setting 2번 값이 더 높게 나오는 것을 확인할 수 있었습니다.
- Setting 값 2, 3번을 비교 해보면 Epoch20, 30으로 차이를 뒀으며, 두 셋팅이 다 돌아간 상태에서의 검증 값은 크게 차이가 없어 보입니다.
- Setting 값 3번에서 Epoch 초반에 V자로 Validation Acc값이 떨어지는 데, 이유를 유추해 보자면, 학습 초기에 노이즈 데이터를 학습하여 정확도가 순간적으로 떨어진 것이 아닌가 생각됩니다.

결론

결과 요약 및 의미

- 논문 학습 결과는 Precision, Recall, F1-score 수치가 약 96%였으며, 논문 코드 구현 결과는 약 95%가량 나와 논문과 유사하게 구현했다고 볼 수 있을 것 같습니다.

개선점

1. 하이퍼 파라미터를 임의로 조정하며 실험한 경험은 있었지만, 이를 최적화 방법인 그리드 탐색, 랜덤 탐색, 베이지안 최적화 등 해당 작업을 코드로 구현하지 못한 점이 아쉽습니다.

2. 프로젝트 진행 중 데이터 증강 및 전처리 문제로 인해 학습 정확도가 낮게 나타난 적이 있습니다. 이를 통해 데이터의 품질이 학습 및 결과에 큰 영향을 미친다는 점을 깨달았고, 이후 프로젝트나 학습에 참여할 때는 데이터 증강과 전처리 과정에 더 많은 신경을 써야겠다는 생각이 들었습니다.

3. 프로젝트 과정에서 메모리 오류가 종종 발생했는데, 이 부분에 대한 확실한 해결을 하지 못하고 진행 되었습니다. 다음 기회가 된다면 메모리 오류가 발생하지 않도록, 메모리 사용량 최적화를 위해 코드를 변경하거나, 데이터를 일부분씩 처리하거나, 데이터 압축 등의 방법으로 이 부분을 해소하며 진행하고 싶습니다.

감사합니다