지능화 캡스톤 프로젝트

프로젝트 #1 결과 발표

2022. 4. 13

충북대학교 산업인공지능학과

INDUSTRIAL AI RESEARCH CENTER

수행방법 및 기여도

수행방법

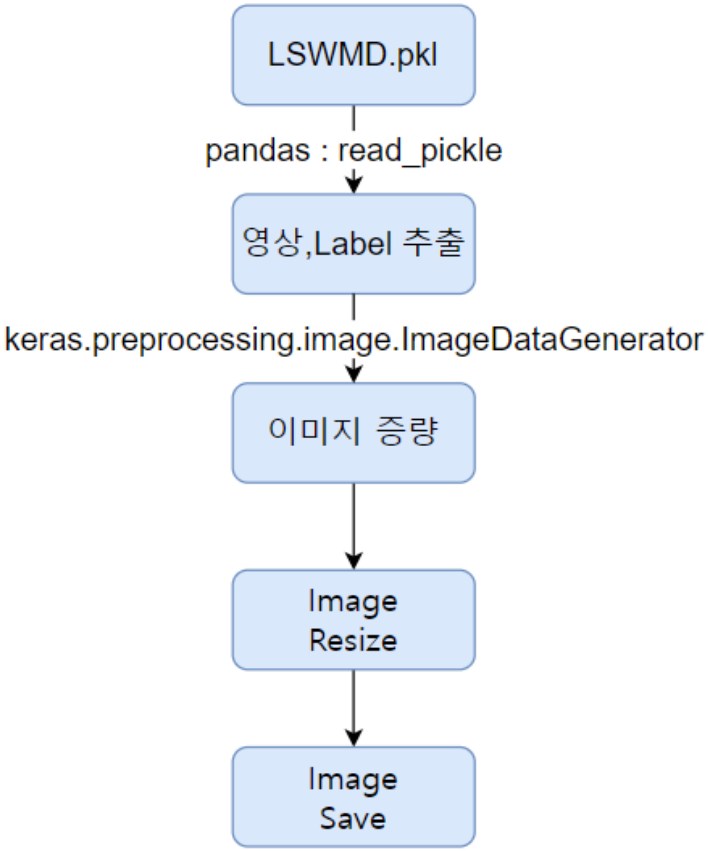
- Kaggle 에 업로드 되어있는 LSWMD.pkl 파일 분석.
- Dataset 을 Image 추출 및 증량 Part 와 학습 코드 Part 로 분장.
- 증량된 Image 파일을 학습 코드에서 Loading 하여 결과 도출.

업무분장 및 기여도

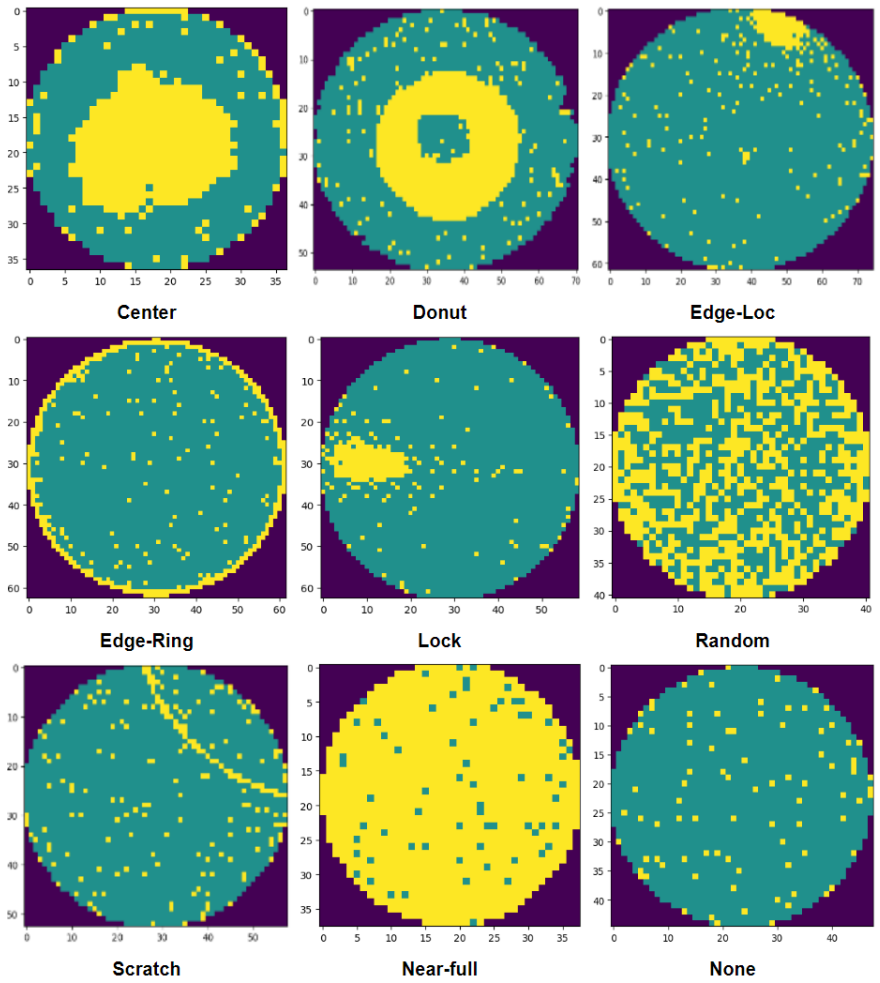
이름	비중	수행내용	비고
이용규	50%	• 데이터 증량 • 주제발표	
유대건	50%	• 코딩/학습 • 결과발표	

데이터셋

데이터셋
- WM-811K wafer map



Processing 순서



WM-811K wafer map 불량 패턴

※ 데이터 셋 소스 - https://www.kaggle.com/qingyi/wm811k-wafer-map/code

데이터셋

LSWMD.pkl -> 영상 추출

```
import pandas as pd
import gc
import matplotlib.pyplot as plt

df=pd.read_pickle("../input/LSWMD.pkl")
s = df.waferMap.size
print(df)

for i in range(0,s):
    if df.trianTestLabel[i].size > 0:
        img = df.waferMap[i]
        trainTest = df.trianTestLabel[i][0][0]
        failure = df.failureType[i][0][0]
        if trainTest == "Training":
            savefolder = parent_loc+
                        '/waferimages/training/'+str(failure)
            loc = parent_loc+'/waferimages/training/'
                  +str(failure)+'/'+str(i)+'.png'
        else:
            savefolder = parent_loc +
                        '/waferimages/testing/' + str(failure)
            loc = parent_loc+'/waferimages/testing/'
                        +str(failure)+'/'+str(i)+'.png'

        MakeFolders(savefolder)
        plt.imsave(loc,img)
        i= i+1
```

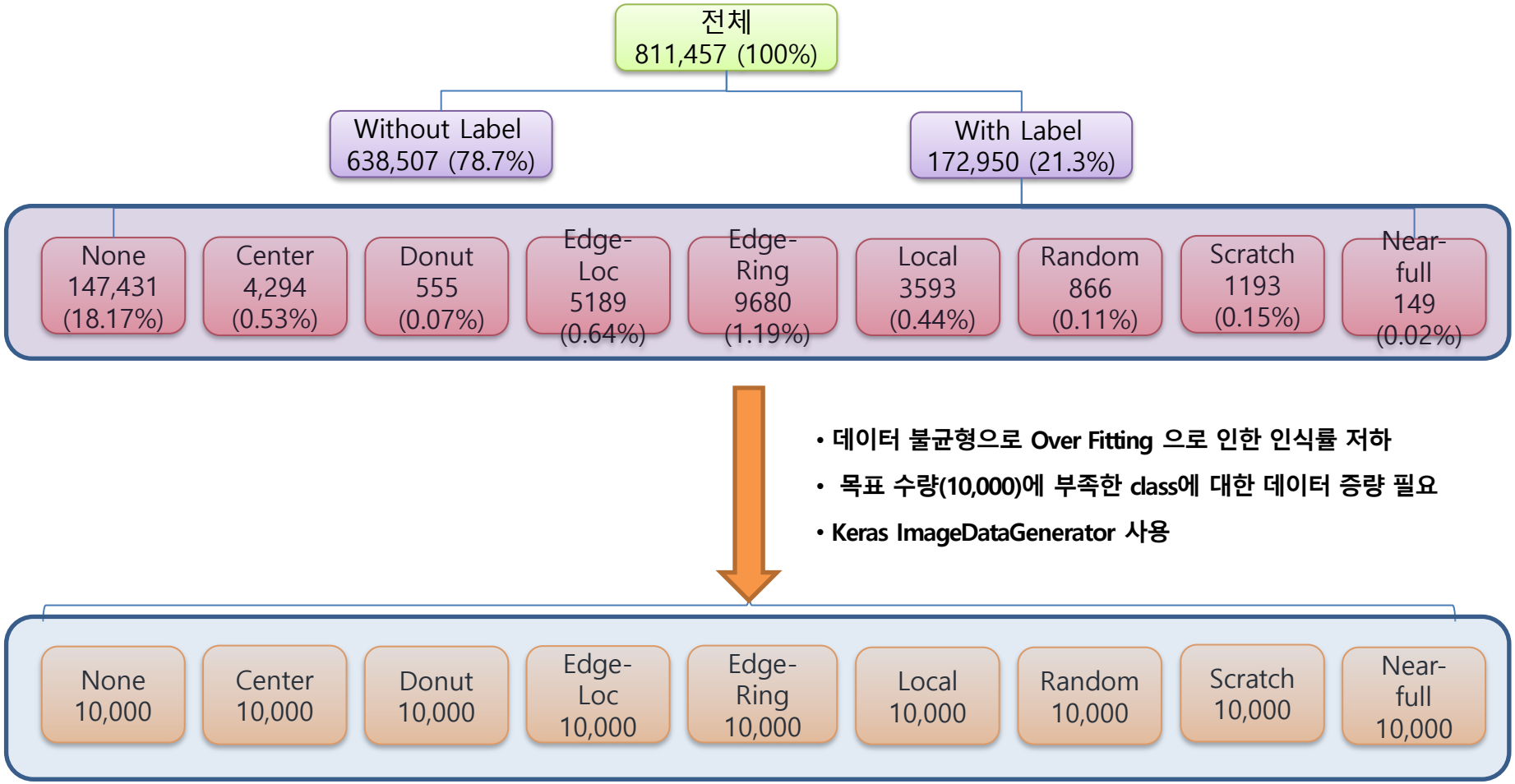
총 데이터 수량 : 811,457

	training	testing	sum
Center	3462	832	4294
Donut	409	146	555
Edge-Loc	2417	2772	5189
Edge-Ring	8554	1126	9680
Loc	1620	1973	3593
Near-full	54	95	149
none	36730	110701	147431
Random	609	257	866
Scratch	500	693	1193
total	54355	118595	172950

Label이 존재하는 데이터 수량

데이터셋

데이터 구성



데이터 목표량

데이터셋

데이터 증량

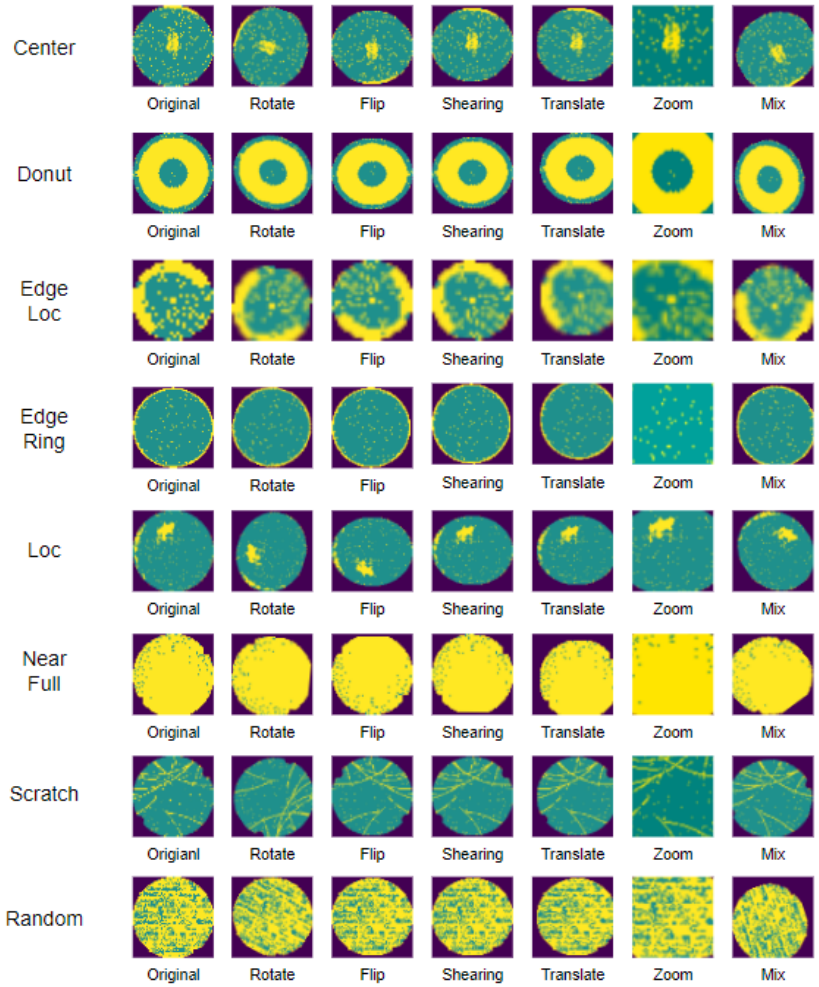
- 목적 : 데이터의 보강을 위한 이미지 변형
- 적용 방법 : ImageDataGenerator 사용
- 목표 수량 : 각 Class 별 10,000 장

```python
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np

image_generator = ImageDataGenerator(rotation_range=10,
                zoom_range=0.10, #지정된 범위 내 이미지 확대
                shear_range=0.5,# 지정된 범위 내 시계 반대 방향으로 이미지 기울기
                width_shift_range=0.10,# 지정된 범위 내 좌우 이동
                height_shift_range=0.10,# 지정된 범위 내 상하 이동
                horizontal_flip=True,# 이미지 가로 뒤집기
                vertical_flip=False  # 이미지 새로 뒤집기
                )
i = 0
for batch in image_generator.flow(
                x,# 원본 이미지
                save_to_dir='./augumentation',# 저장 위치
                save_prefix='gen',# 파일명 접두어
                 save_format='png'# 이미지 저장 포맷
            ):
        i += 1
        if i > 3:
            break
```

Sample Code

데이터셋

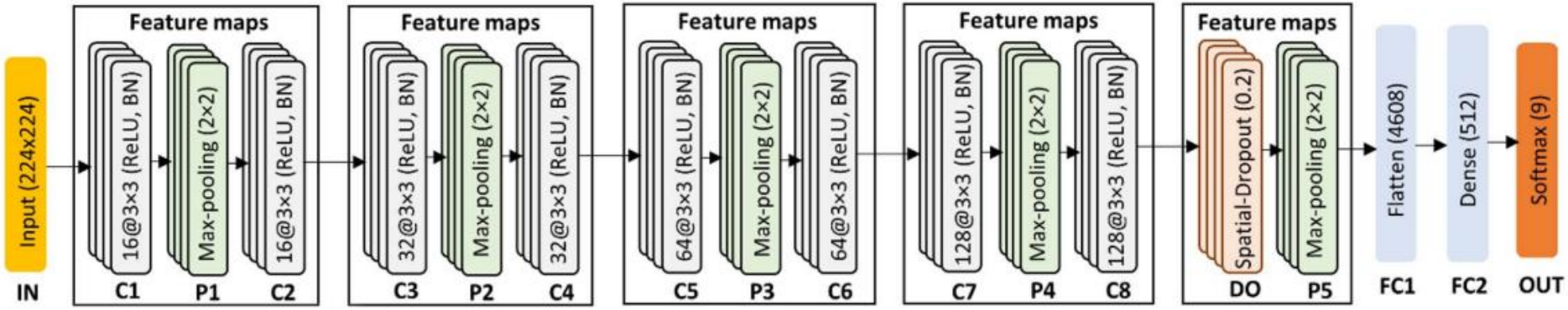
데이터 증량 결과



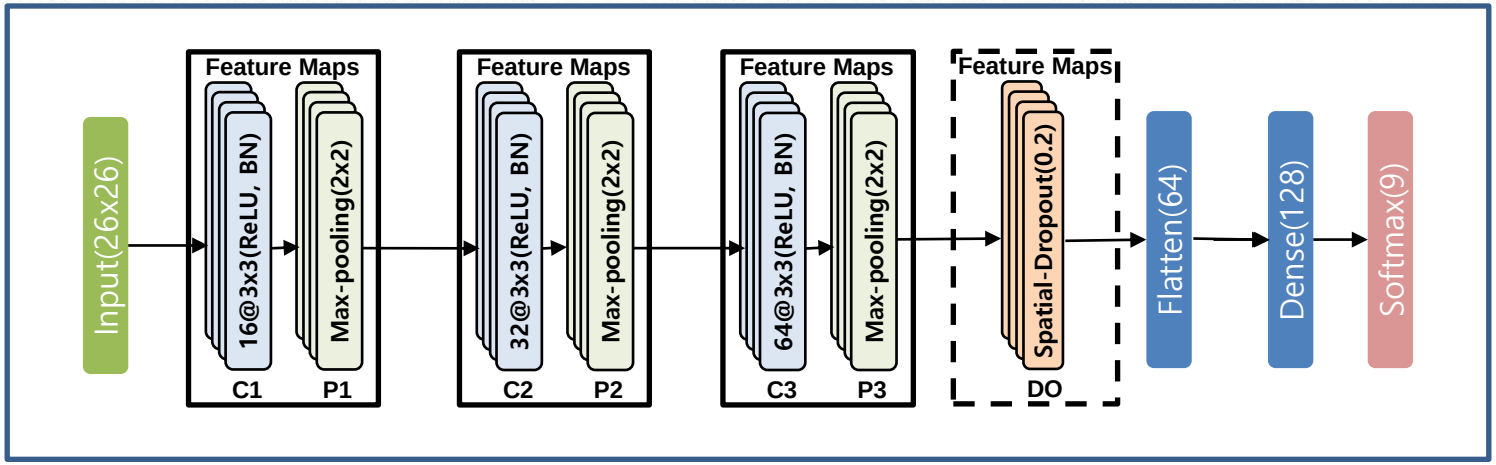
데이터 증량 샘플

CNN 구조

CNN 구조



*Note: IN denotes input layer; C convolutional layer; P pooling layer; DO dropout layer; FC fully connected layer; OUT output layer; and BN batch normalization



- 하나의 입력층, 각 배치 정규화, Zero 패딩 및 ReLU 활성화가 있는 3개의 Conv 계층, 3개의 풀 계층, 2개의 완전연결(FC) 계층, 1개의 출력층 적용.
- 첫 번째, 두 번째, 세 번째 Conv-Pool 그룹에 16, 32, 64 개의 Feature Map 을 적용.

CNN 구조

CNN 구조

```
In [10]: model0.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 26, 26, 3)         0

 conv2d (Conv2D)             (None, 24, 24, 16)        448

 max_pooling2d (MaxPooling2D  (None, 12, 12, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 10, 10, 32)        4640

 max_pooling2d_1 (MaxPooling  (None, 5, 5, 32)         0
 2D)

 conv2d_2 (Conv2D)           (None, 3, 3, 64)          18496

 max_pooling2d_2 (MaxPooling  (None, 1, 1, 64)         0
 2D)

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 128)               8320

 dense_1 (Dense)             (None, 9)                 1161

=================================================================
Total params: 33,065
Trainable params: 33,065
Non-trainable params: 0
_________________________________________________________________
```

과적합을 방지하기 위한 규제화(regulation)

- 출력층만 softmax 함수 적용.
- SpatilalDropout(SD) = 0.2 : SD는 Conv층에서 피처맵을 삭제

CNN 구조

주요 코드 및 실행 결과

- 딥러닝 프레임워크(tensorflow, keras)

In [8]:
```
num_classes = 9

model0 = tf.keras.Sequential([
  layers.experimental.preprocessing.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
  tf.keras.layers.Conv2D(16, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(32, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Conv2D(64, 3, activation='relu'),
  tf.keras.layers.MaxPooling2D(),
  tf.keras.layers.Flatten(),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dense(num_classes, activation='softmax')
])
```

In [9]:
```
model0.compile(
  optimizer='adam',
  loss=tf.losses.SparseCategoricalCrossentropy(from_logits=True),
  metrics=['accuracy'])
```

CNN 구조

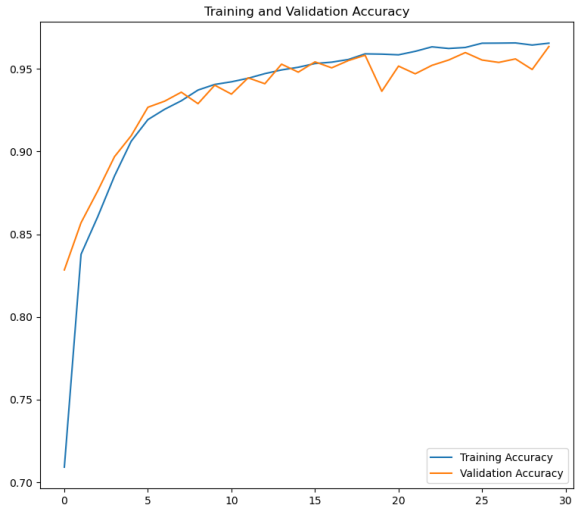
주요 코드 및 실행 결과

In [13]:
```
epoch = 30

history = model0.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epoch
)
```

```
720/720 [==============================] - 7s 10ms/step - loss: 1.4311 - accuracy: 0.9422 - val_loss: 1.4386 - val_accuracy: 0.9347
Epoch 12/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4284 - accuracy: 0.9443 - val_loss: 1.4284 - val_accuracy: 0.9446
Epoch 13/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4257 - accuracy: 0.9471 - val_loss: 1.4315 - val_accuracy: 0.9410
Epoch 14/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4234 - accuracy: 0.9493 - val_loss: 1.4199 - val_accuracy: 0.9528
Epoch 15/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4215 - accuracy: 0.9510 - val_loss: 1.4246 - val_accuracy: 0.9480
Epoch 16/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4195 - accuracy: 0.9532 - val_loss: 1.4183 - val_accuracy: 0.9542
Epoch 17/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4185 - accuracy: 0.9540 - val_loss: 1.4217 - val_accuracy: 0.9506
Epoch 18/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4169 - accuracy: 0.9557 - val_loss: 1.4172 - val_accuracy: 0.9550
Epoch 19/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4135 - accuracy: 0.9591 - val_loss: 1.4145 - val_accuracy: 0.9582
Epoch 20/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4137 - accuracy: 0.9589 - val_loss: 1.4360 - val_accuracy: 0.9364
Epoch 21/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4137 - accuracy: 0.9585 - val_loss: 1.4216 - val_accuracy: 0.9517
Epoch 22/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4119 - accuracy: 0.9606 - val_loss: 1.4254 - val_accuracy: 0.9470
Epoch 23/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4093 - accuracy: 0.9633 - val_loss: 1.4207 - val_accuracy: 0.9521
Epoch 24/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4101 - accuracy: 0.9623 - val_loss: 1.4168 - val_accuracy: 0.9553
Epoch 25/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4095 - accuracy: 0.9629 - val_loss: 1.4124 - val_accuracy: 0.9598
Epoch 26/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4068 - accuracy: 0.9655 - val_loss: 1.4167 - val_accuracy: 0.9553
Epoch 27/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4066 - accuracy: 0.9656 - val_loss: 1.4187 - val_accuracy: 0.9539
Epoch 28/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4067 - accuracy: 0.9657 - val_loss: 1.4162 - val_accuracy: 0.9560
Epoch 29/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4080 - accuracy: 0.9644 - val_loss: 1.4223 - val_accuracy: 0.9496
Epoch 30/30
720/720 [==============================] - 7s 10ms/step - loss: 1.4067 - accuracy: 0.9655 - val_loss: 1.4088 - val_accuracy: 0.9635
```

CNN 구조

주요 코드 및 실행 결과



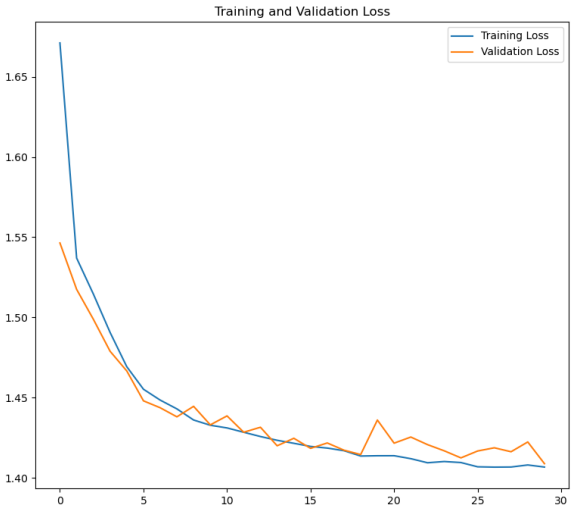
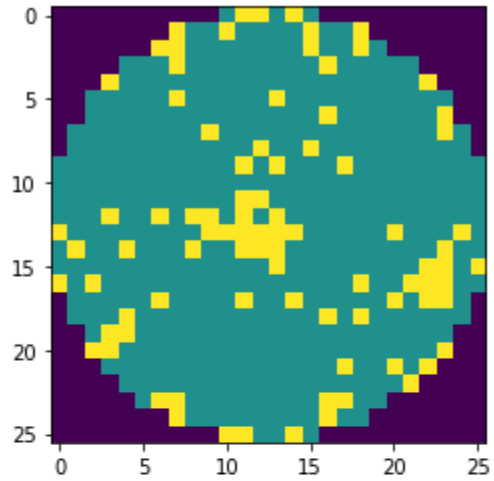


Image 증량 전	Image 증량 후
Training Accuracy : 0.9059	Training Accuracy : 0.9655
Validation Accuracy : 0.9043	Validation Accuracy : 0.9635
Training Loss : 1.4663	Training Loss : 1.4067
Validation Loss : 1.4674	Validation Loss : 1.4088

CNN 구조

주요 코드 및 실행 결과

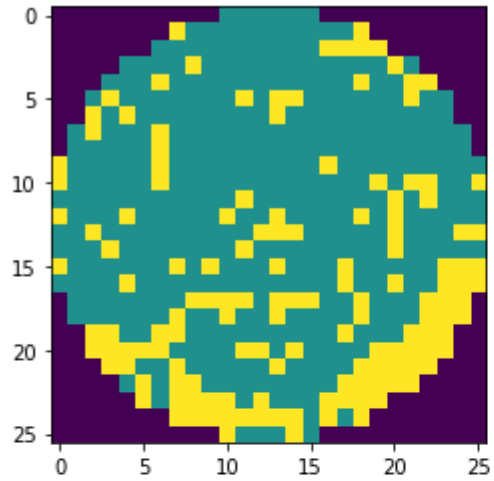


In [23]:
```
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model0.predict(img_array)

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(predictions)], 100 * np.max(predictions))
)
```

```
This image most likely belongs to Center with a 99.28 percent confidence.
```

CNN 구조

주요 코드 및 실행 결과



In [25]:
```
img_array = keras.preprocessing.image.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model0.predict(img_array)

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(predictions)], 100 * np.max(predictions))
)
```

```
This image most likely belongs to Edge-Loc with a 52.86 percent confidence.
```

학습 방법

딥러닝 학습 조건

- (HW) PC 사양, 학습시간

CPU : 11th Gen Intel(R) Core(TM) i5-1135G7 @ 2.40GHz

RAM : 16GB

GPU : Intel(R) Iris(R) Xe Graphics(On Board)

- 하이퍼파라미터

batch size - 100

epoch수 – 30

학습률 – 적용안함.

optimizer - Adam

loss 함수 - SparseCategoricalCrossentropy

- 학습 중 알게 된 내용 등

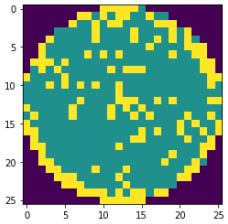
학습률/optimzer에 따른 학습추이 비교

Learning Rate	Adam	SGD	Adagrad	RMSprop
0	96.57	91.73	77.34	96.07

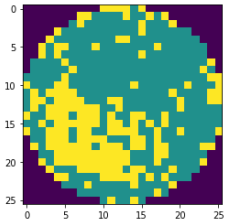
결과 및 토의

분류 성능

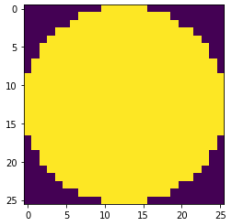
- *Confusion matrix 및 평가지표 (논문 결과와 비교 : 미달)*
- *Confusion matrix 미구현으로 단일 Image 에 대해서만 예측 결과 확인*



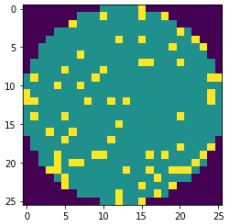
Edge-Ring : 97.80



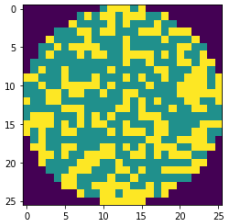
Loc ->
Edge-Loc : 70.02



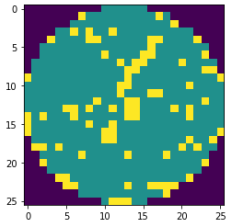
Near-full : 99.99



none : 99.25



Random : 94.18



Scratch : 93.70

OVERALL PERFORMANCE COMPARISON OF VARIOUS CLASSIFIERS (%)

Classifier	Training Acc	Validation Acc	Testing Acc	Precision	Recall	F1-Score
CNN-WDI	98.9	**96.4**	**96.2**	**96.2**	**96.2**	**96.2**
CNN-D	97.6	95.5	95.2	95.2	95.2	95.2
CNN-BN	**99.4**	95.6	95.6	95.6	95.6	95.6
CNN-SD	98.6	94.7	94.8	94.8	94.8	94.8
VGG-16	82.3	80.0	80.1	80.3	80.1	79.9
ANN	95.9	95.9	72.0	95.2	95.9	95.4
SVM	91.3	91.0	32.6	87.5	91.0	88.0

Note: Boldface numbers denote the highest values of different performance measures and Acc accuracy

결과 및 토의

토의 및 개선점

- 학습된 모델에 단일 이미지를 입력으로 주어 예측한 결과 클래스를 예측하지 못하는 경우 발생.
- 학습률, 옵티마이저 등 여러 조건을 적용하여 학습결과 비교 필요.
- 학습 후 Accuracy 는 높게 평가되었으나 예측결과가 다소 불안정.
- 이미지 증량하는 방법을 다양화할 필요성 있음.

결과 및 토의

참고사항

1. 발표방법 안내
   - 심사위원 : 박태형 학과장님, 기석철 교수님, 가디언(4인)
   - 발표시간 : 10분 이내 (시간준수) + Q&A
   - 발표자료 제출 : 4/13(수) eCampus / 담당교수 이메일

2. 평가기준

항목	내용	점수
우수성	결과 정확도, 속도	25
창의성	접근방법의 차별성, 아이디어의 독창성	25
발표력	발료자료, 설득력, 전달력	25
난이도	적용기술의 난이도	25

감사합니다