지능화 캡스톤 프로젝트

프로젝트 #1 결과 발표

2022. 4. 13

충북대학교 산업인공지능학과

[21-5조] 방창현, 정원용

INDUSTRIAL AI RESEARCH CENTER

수행방법 및 기여도

수행방법

- 같은 회사에 재직중이어서 수시로 얘기하여 업무 분장 및 수행 진행
- 방창현 : 현업에서 개발 보다는 프로젝트 관리 업무 수행중이며 관련 경험에 맞추어 코딩보다는 데이터 셋 관련 업무 중심으로 진행함.
- 정원용 : 현업에서 개발에 관한 설계 및 개발 업무를 진행하고 있어서 코딩/학습 관련 업무 중심으로 진행함.

업무분장 및 기여도

이름	비중	수행내용	비고
방창현	50%	• 데이터 증량 • 주제발표	
정원용	50%	• 코딩/학습 • 결과발표	

- 비중은 총합이 100%일 것

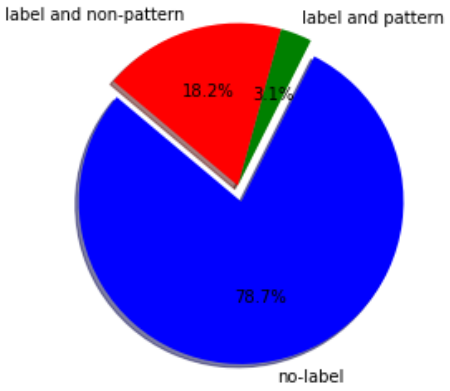
데이터셋

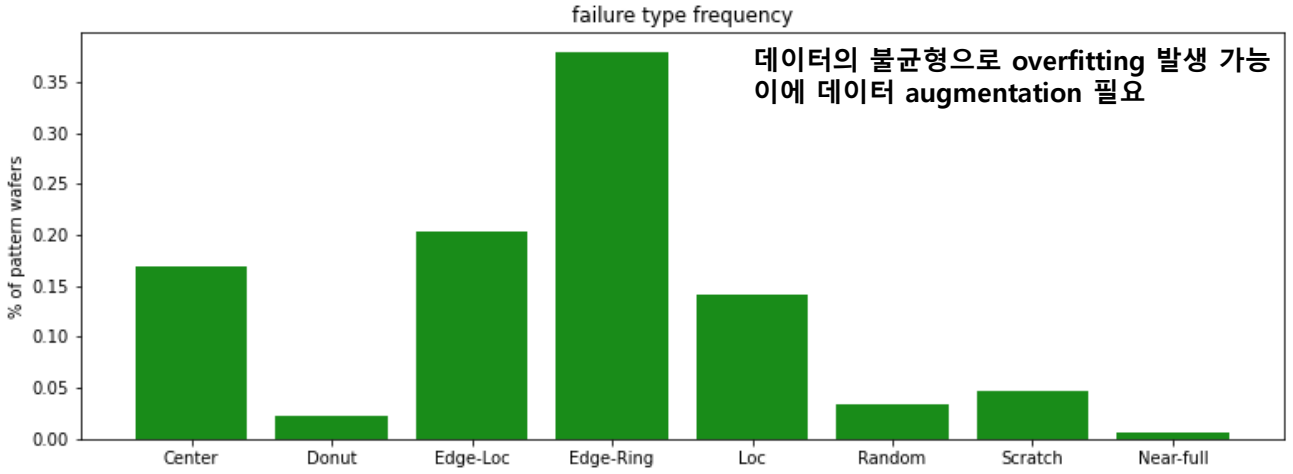
Data augmentation/전처리

데이터 : 'LSWMD.pkl ' (kaggle에서 다운로드)

실제 제조에서 46,393개 로트에서 수집된 811,457개의 웨이퍼 맵에 대한 데이터

이 데이터셋의 모든 결함 유형에 대한 종류 :
Center, Donut, Edge-Loc, Edge-Ring, Loc, Random, Scratch, Near-full, none.



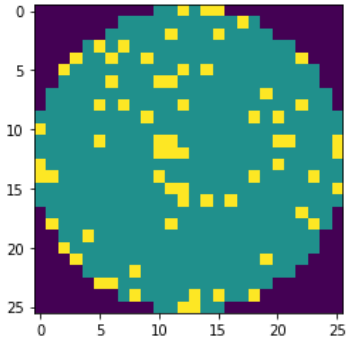


Total wafers = 811457
With label = 172950
Non label = 638507

With pattern = 25519
Non pattern = 147431

데이터셋

Data augmentation/전처리



Faulty case : ['none']

26 X 26

Center : 90
Donut : 1
Edge-Loc : 296
Edge-Ring : 31
Loc : 297
Near-full : 16
Random : 74
Scratch : 72
none : 13489
new_x.shape : (14366, 26, 26, 3)

Encoder

epoch= 30
batch_size= 1024
padding= 'same'
activation= 'sigmoid'
optimizer = 'Adam'
loss = 'mse'

Decoder

epoch= 30
batch_size= 1024
padding= 'same'
activation= 'relu', 'sigmoid'
optimizer = 'Adam'
loss = 'mse'

학습절차

1. 원래 결함 웨이퍼 인코딩
2. 인코딩된 잠재적 결함 웨이퍼 벡터에 노이즈를 추가.
3. 모든 결함 케이스에 대한 보강
4. 데이터 증량
   - 웨이퍼의 총 개수가 2000개가 될 때까지 웨이퍼를 만듬
   - 동일한 길이의 레이블 벡터를 만듬.

데이터셋

Data augmentation/전처리

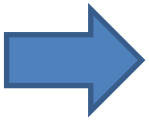
학습결과

After Generate new_x shape : (30707, 26, 26, 3),
new_y shape : (30707, 1)

Center : 2160
Donut : 2002
Edge-Loc : 2368
Edge-Ring : 2046
Loc : 2376
Near-full : 2032
Random : 2146
Scratch : 2088
none : 13489

After Delete "none" class new_x shape : (19707, 26, 26, 3),
new_y shape : (19707, 1)

Center : 2160
Donut : 2002
Edge-Loc : 2368
Edge-Ring : 2046
Loc : 2376
Near-full : 2032
Random : 2146
Scratch : 2088
none : 2489



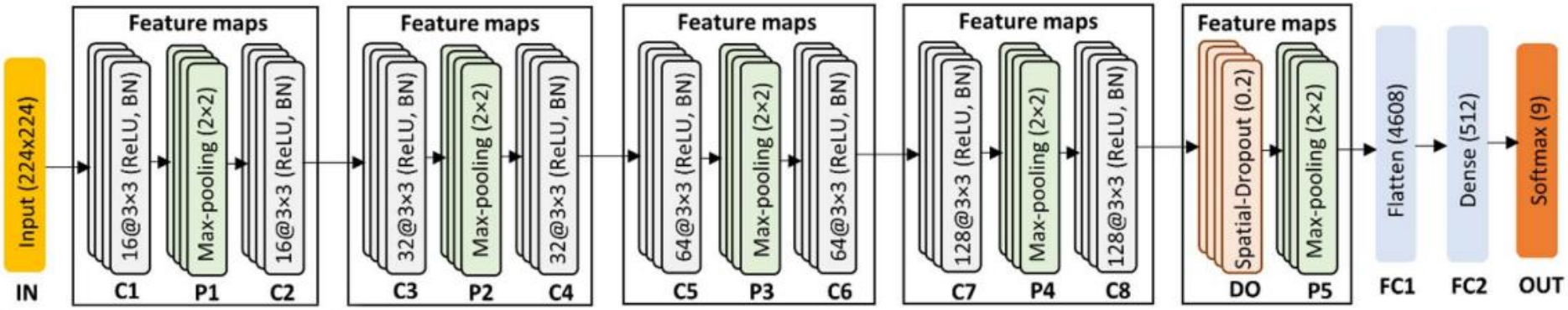
Train
x : (**12730**, 26, 26, 3),
y : (12730, 9)

Test
x: (**6270**, 26, 26, 3),
y : (6270, 9)

CNN 구조

CNN 구조

- *그림/표/모델정보(실행화면)로 표현 (논문과 동일? 다르면 어떤 부분이 다른지 등)*



*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

C2	Convolution2	16	111×111	3×3	Yes	ReLU
C3	Convolution3	32	111×111	3×3	Yes	ReLU
P2	Max Pooling2	32	55×55	2×2	No	-
C4	Convolution4	32	55×55	3×3	Yes	ReLU
C5	Convolution5	64	55×55	3×3	Yes	ReLU
P3	Max Pooling3	64	27×27	2×2	No	-
C6	Convolution6	64	27×27	3×3	Yes	ReLU
C7	Convolution7	128	27×27	3×3	Yes	ReLU
P4	Max Pooling4	128	13×13	2×2	No	-
C8	Convolution8	128	13×13	3×3	Yes	ReLU
P5	Max Pooling5	128	6×6	2×2	No	-
FC1	Fully-Connected1	1	4608	-	-	ReLU
FC2	Fully Connected2	1	512	-	-	ReLU
OUT	Output	1	9	-	-	Softmax

과적합을 방지하기 위한 규제화(regulation)

- *Batch Normalization(정규화)*
- *Spatial Dropout = 0.2*

CNN 구조

주요 코드 및 실행 결과

- *딥러닝 프레임워크(tensorflow, keras)*

```
# Encoder
input_shape = (26, 26, 3)
input_tensor = Input(input_shape)
encode = layers.Conv2D(64, (3,3), padding='same', activation='relu')(input_tensor)

latent_vector = layers.MaxPool2D()(encode)

# Decoder
decode_layer_1 = layers.Conv2DTranspose(64, (3,3), padding='same', activation='relu')
decode_layer_2 = layers.UpSampling2D()
output_tensor = layers.Conv2DTranspose(3, (3,3), padding='same', activation='sigmoid')

# connect decoder layers
decode = decode_layer_1(latent_vector)
decode = decode_layer_2(decode)

ae = models.Model(input_tensor, output_tensor(decode))
ae.compile(optimizer = 'Adam',
              loss = 'mse',
             )
```

```
epoch=30
batch_size=1024
```

```
# start train
ae.fit(new_x, new_x,
       batch_size=batch_size,
       epochs=epoch,
       verbose=2)
```

CNN 구조

주요 코드 및 실행 결과

- 활성화 함수 : sigmoid, optimizer : Adam, loss함수 : categorical_crossentropy

```
def create_model():
    input_shape = (26, 26, 3)
    input_tensor = Input(input_shape)

    conv_1 = layers.Conv2D(16, (3,3), activation='sigmoid', padding='same')(input_tensor)
    conv_2 = layers.Conv2D(64, (3,3), activation='sigmoid', padding='same')(conv_1)
    conv_3 = layers.Conv2D(128, (3,3), activation='sigmoid', padding='same')(conv_2)

    flat = layers.Flatten()(conv_3)

    dense_1 = layers.Dense(512, activation='sigmoid')(flat)
    dense_2 = layers.Dense(128, activation='sigmoid')(dense_1)
    output_tensor = layers.Dense(9, activation='sigmoid')(dense_2)

    model = models.Model(input_tensor, output_tensor)
    model.compile(optimizer='Adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    return model
```

학습 방법

딥러닝 학습 조건

- *(HW) PC 사양, 학습시간*

  CPU : Intel CPU i5-1135G7 @ 2.40GHz

  RAM : 16GB

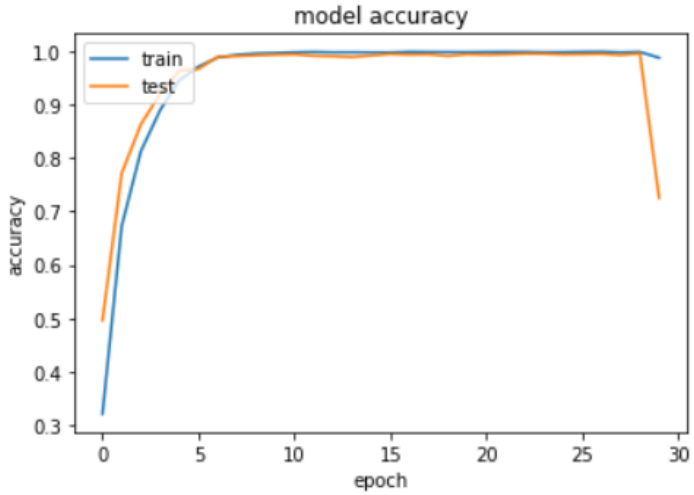
  GPU : 내장 그래픽카드(Iris Xe)

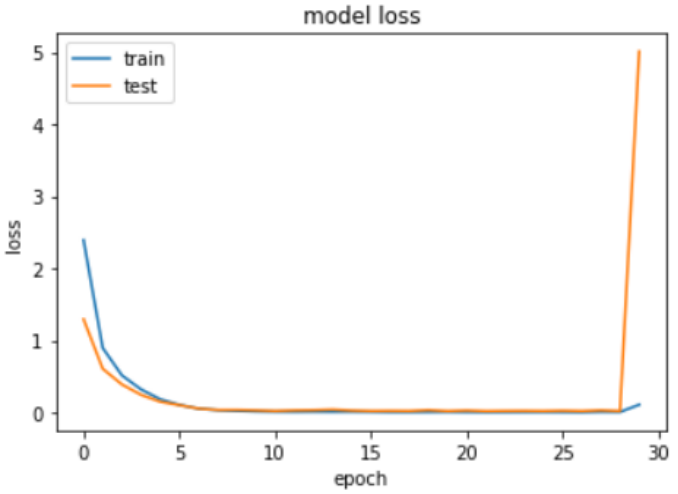
- *하이퍼파라미터 : epoch=30, batch size=1024,*

  optimizer = 'Adam', loss 함수 = 'categorical_crossentropy'

소요시간 : 약 2시간

추이 그래프

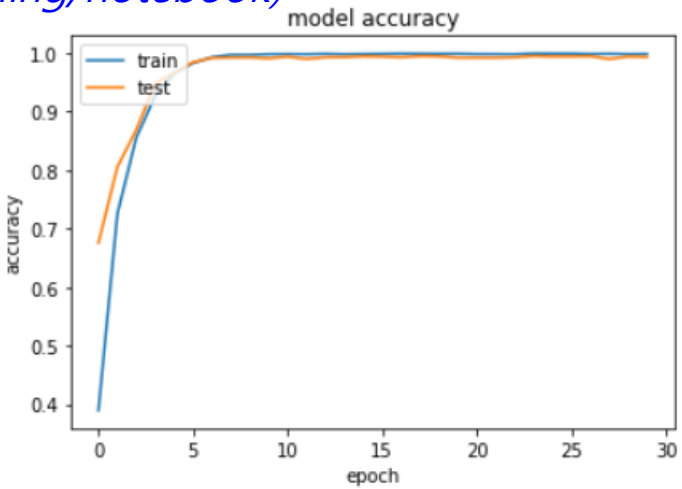


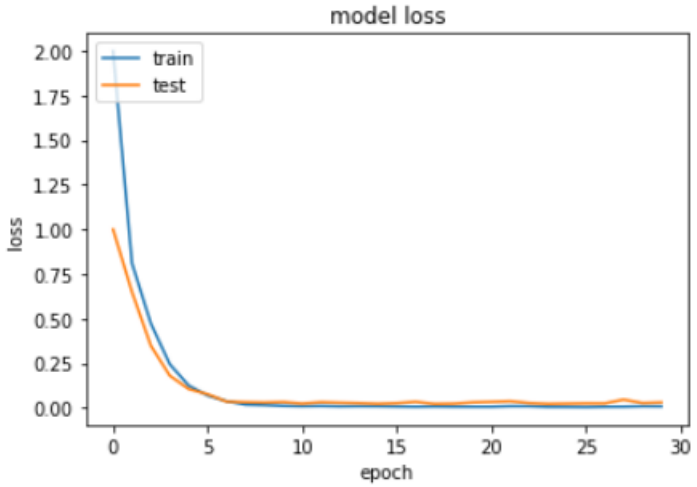


CNN 구조

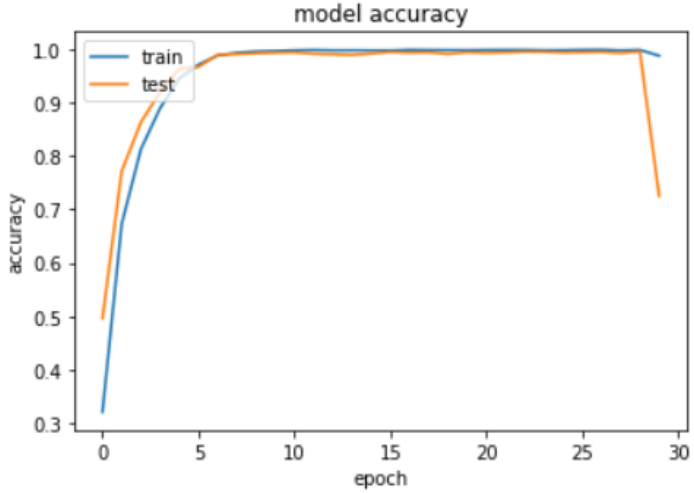
- *Kaggle 참고 사이트 결과*

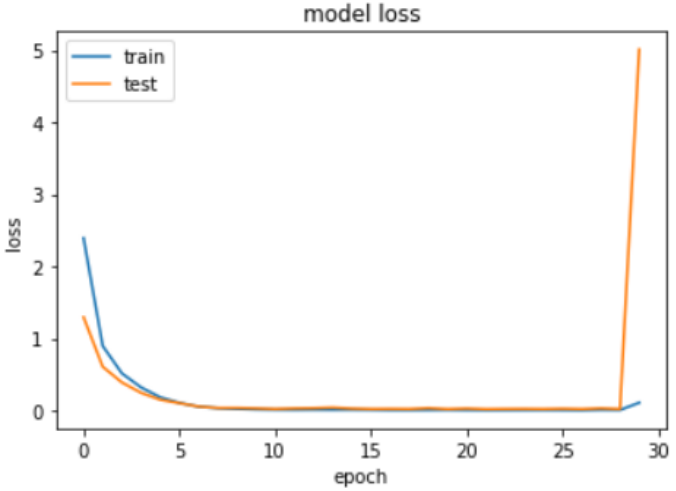
(https://www.kaggle.com/code/shawon10/wafer-defect-classification-by-deep-learning/notebook)





- *구동 결과*

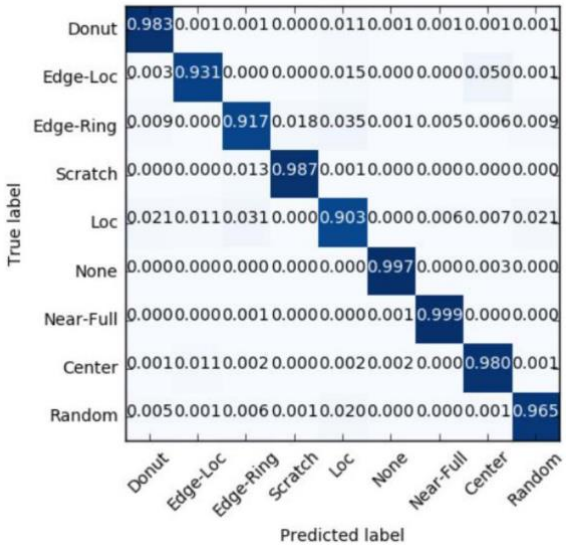




결과 및 토의

분류 성능

- *Confusion matrix 및 평가지표 (논문 결과와 비교 : 우수/동일/미달... 그 이유는?)*
- *구체적인 분석*



논문의 성능과 비교

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

- kaggle의 예제를 이해하기도 벅찬 상태

- 현업의 업무가 우선이고 야근이 많다 보니 주말에 시간내서 하고는 있지만 따라가기 쉽지 않은 상태임.

감사합니다