지능화 캡스톤 프로젝트

프로젝트 #1 결과 발표

2022. 4. 13

충북대학교 산업인공지능학과
21-3조, 최준혁, 이지연

INDUSTRIAL AI
RESEARCH CENTER

수행방법 및 기여도

수행방법

- 같은 회사에 재직하여 업무분담과 진행상황 파악 용이.
- 작업환경은 Colab을 사용하고, 링크를 이용한 공유로 작업 진행.
- 하나의 CNN 예제를 기준으로 하여 각자 업무 진행중 작성된 코드 반영.

업무분장 및 기여도

이름	비중	수행내용	비고
최준혁	50%	• 데이터 전처리 및 증량 작업 • 데이터셋 발표	
이지연	50%	• CNN 작업 및 결과 분석 • CNN 구조 및 학습 발표	

데이터셋

데이터 구성

전체
811,457 (100%)

Without Label
638,507 (78.7%)

With Label
172,950 (21.3%)

None
147,431
(18.17%)

Center
4,294
(0.53%)

Donut
555
(0.07%)

Edge-
Loc
5189
(0.64%)

Edge-
Ring
9680
(1.19%)

Local
3593
(0.44%)

Random
866
(0.11%)

Scratch
1193
(0.15%)

Near-
full
149
(0.02%)

- 회전
- 좌우 대칭
- 수평/수직 이동
- 확대/축소

None
10,000

Center
10,000

Donut
10,000

Edge-
Loc
10,000

Edge-
Ring
10,000

Local
10,000

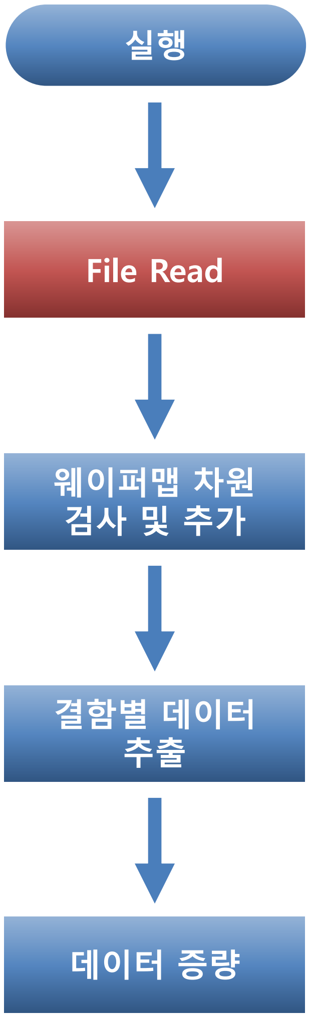
Random
10,000

Scratch
10,000

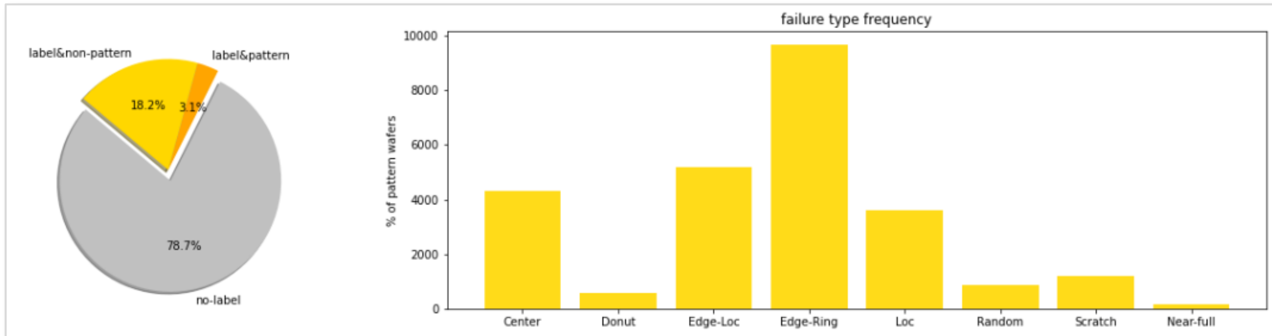
Near-
full
10,000

데이터셋

Data augmentation/전처리




```
df = pd.read_pickle('/content/drive/MyDrive/충북대/2-1 캡스톤/CNN을 이용한 불량 검출/LSWMD.pkl')
```



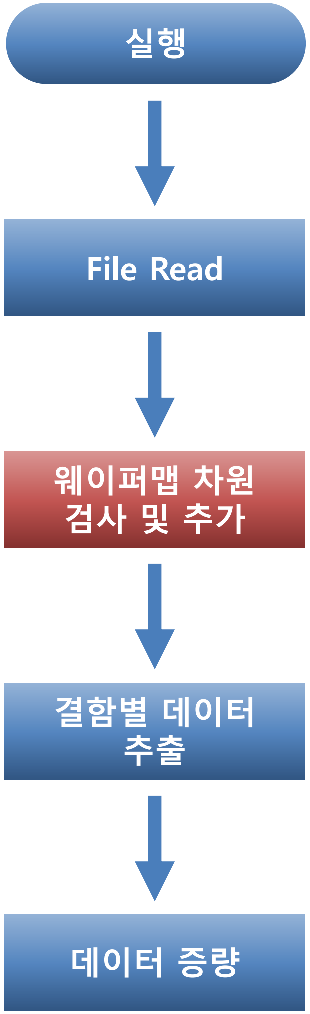
	waferMap	dieSize	lotName	waferIndex	trianTestLabel	failureType
0	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1683.0	lot1	1.0	[[Training]]	[[none]]
1	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1683.0	lot1	2.0	[[Training]]	[[none]]
2	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1683.0	lot1	3.0	[[Training]]	[[none]]
3	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1683.0	lot1	4.0	[[Training]]	[[none]]
4	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1683.0	lot1	5.0	[[Training]]	[[none]]

```
#라벨 있는 데이터, None 아닌 패턴라벨 있는 데이터, None 패턴 데이터
data_with_label.shape[0], data_with_pattern.shape[0], data_non_pattern.shape[0]

(172950, 25519, 147431)
```

LSWMD.pkl 데이터 구조

데이터셋

Data augmentation/전처리




```
#전체 데이터 갯수
print(df.shape[0])

#웨이퍼 맵의 서로 다른 차원 갯수 확인.
uni_waferDim=np.unique(df.waferMapDim, return_counts=True)
print(uni_waferDim[0].shape[0])
```

```
811457
632
```

```
mapping_type={'Center':0,'Donut':1,'Edge-Loc':2,'Edge-Ring':3,'Loc':4,'Random':5,'Scratch':6,'Near-full':7,'none':8}
```

	waferMap	dieSize	lotName	waferIndex	trianTestLabel	failureType	waferMapDim	failureNum	trainTestNum
137040	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1801.0	lot8918	15.0	[]	[]	(56, 41)	[]	[]
168047	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	5133.0	lot10709	5.0	[]	[]	(87, 75)	[]	[]
470701	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1513.0	lot28951	24.0	[]	[]	(49, 39)	[]	[]
775651	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1442.0	lot46087	8.0	[[Test]]	[[none]]	(41, 45)	8	1
79221	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	776.0	lot5768	16.0	[]	[]	(30, 34)	[]	[]
136429	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1801.0	lot8892	20.0	[]	[]	(56, 41)	[]	[]
425622	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	2412.0	lot25566	4.0	[]	[]	(51, 61)	[]	[]
94765	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 1,...	600.0	lot6684	2.0	[]	[]	(26, 30)	[]	[]
490021	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,...	710.0	lot30306	2.0	[]	[]	(32, 29)	[]	[]
789969	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,...	1187.0	lot46668	6.0	[[Test]]	[[none]]	(51, 30)	8	1

데이터셋

Data augmentation/전처리

- 실행
- File Read
- 웨이퍼맵 차원 검사 및 추가
- 결함별 데이터 추출
- 데이터 증량

결함번호

```python
mapping_type={'Center':0,'Donut':1,'Edge-Loc':2,'Edge-Ring':3,'Loc':4,'Random':5,'Scratch':6,'Near-full':7,'none':8}
```

```python
#case 0 ~ 8
for i in range(0, 9):
  _x, _y = preProcess(i)
```

결함번호

해당 이미지, 라벨

```python
#전처리
def preProcess(failureNum):
  data_with_label = df[(df['failureNum']==failureNum) & (df['waferMapDim'] == (26, 26))]
  data_with_label = data_with_label.reset_index()

  sw = np.ones((1, 26, 26))
  label = list()

  for i in range(len(data_with_label)):
      # skip null label
      if len(data_with_label.iloc[i,:]['failureType']) == 0:
          continue
      sw = np.concatenate((sw, data_with_label.iloc[i,:]['waferMap'].reshape(1, 26, 26)))
      label.append(failureNum)

  x = sw[1:]
  x = x.reshape((x.shape[0], x.shape[1], x.shape[2], 1))
  y = np.array(label).reshape((-1,1))

  return x, y
```

해당 데이터 병합

데이터셋

Data augmentation/전처리

실행

File Read

웨이퍼맵 차원 검사 및 추가

결함별 데이터 추출

데이터 증량

결함번호

```
mapping_type={'Center':0,'Donut':1,'Edge-Loc':2,'Edge-Ring':3,'Loc':4,'Random':5,'Scratch':6,'Near-full':7,'none':8}
```

```
#case 0 ~ 8
for i in range(0, 9):
  _x, _y = preProcess(i)
  data_x, data_y = imageGen(_x, i)
```

결함번호

해당 이미지, 라벨

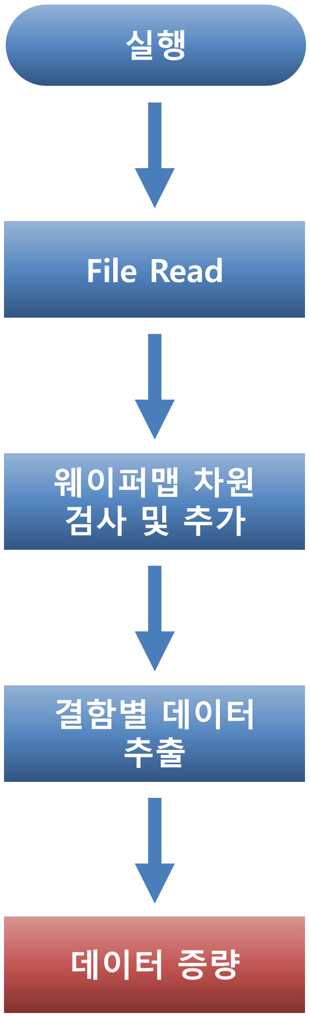
```
#증량
def imageGen(x, failureNum):

  gen_x = np.zeros((1, 26, 26, 1))

  X_data = x

  datagen = ImageDataGenerator(
    rescale = 1./255, # 모든 이미지 원소값들을 255로 나누기
    rotation_range=50, # 0~25도 사이에서 임의의 각도로 원본이미지를 회전
    zoom_range=0.2, # (1-0.2)~(1+0.2) => 0.8~1.2 사이에서 임의의 수치만큼 확대/축소
    horizontal_flip=True, # 좌우로 뒤집기
    vertical_flip=True,
    fill_mode='nearest',
  )

  datagen.fit(X_data)

  label = list()
  for i in range(0, 10000):
    flow=datagen.flow(_x,_y,batch_size=1, shuffle=True)
    x, y = flow.next()
    gen_x = np.concatenate((gen_x, x), axis=0)
    label.append(failureNum)

  gen_x = gen_x[1:]
  gen_y = np.array(label).reshape((-1,1))
  return gen_x, gen_y
```

`from keras_preprocessing.image import ImageDataGenerator` 활용

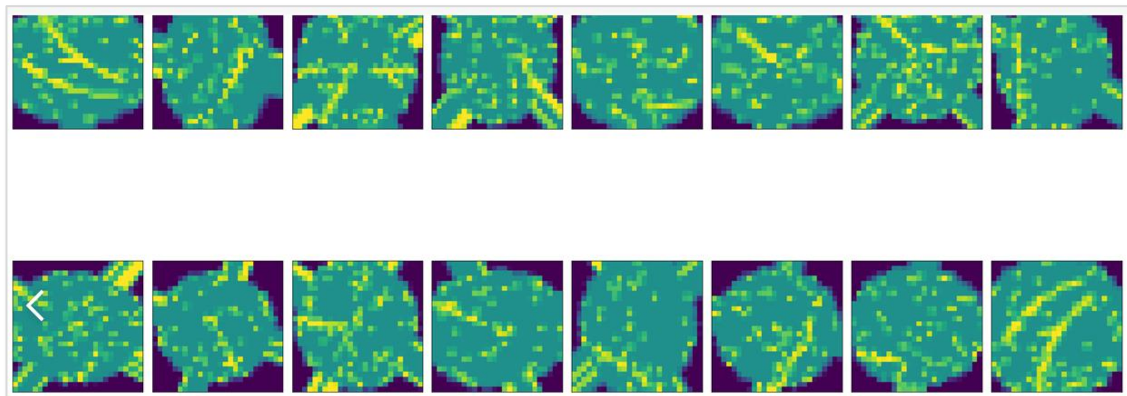
데이터 생성 후 변수에 병합

데이터셋

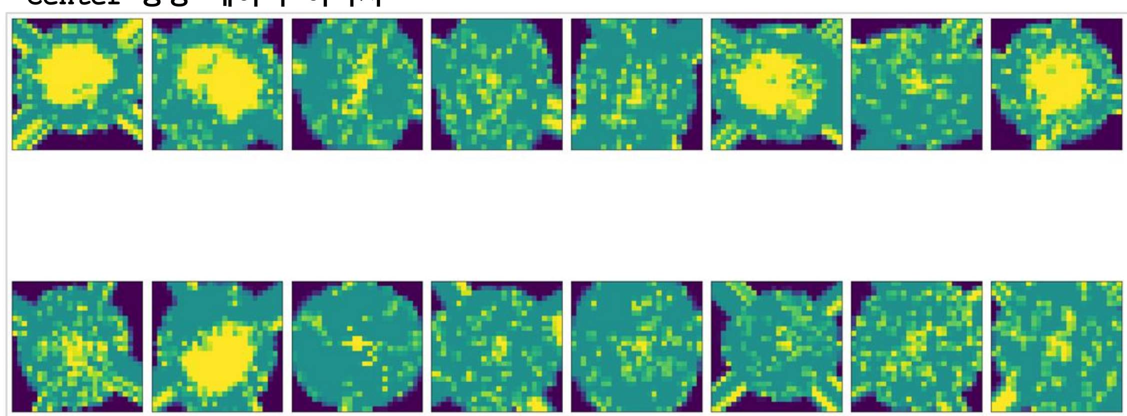
Data augmentation/전처리



Scratch 증량 데이터 이미지



Center 증량 데이터 이미지



데이터셋

주요 코드 및 실행 결과

```python
#df = pd.read_pickle('/content/drive/MyDrive/LSWMD.pkl')
df = pd.read_pickle('/content/drive/MyDrive/충북대/2-1 캡스톤/CNN을 이용한 불량 검출/LSWMD.pkl')
```

```python
# 차원 검사용 변수 waferMapDIM 생성
def find_dim(x):
    dim0=np.size(x,axis=0)
    dim1=np.size(x,axis=1)
    return dim0,dim1
df['waferMapDim']=df.waferMap.apply(find_dim)
df.sample(10)
```

	waferMap	dieSize	lotName	waferIndex	trianTestLabel	failureType	waferMapDim
418995	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1,...	899.0	lot25130	20.0	[]	[]	(37, 31)
374375	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2,...	811.0	lot22362	15.0	[]	[]	(34, 31)
764609	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2,...	904.0	lot45635	13.0	[[Test]]	[[Loc]]	(34, 34)
209355	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	3532.0	lot13355	4.0	[]	[]	(64, 71)
240662	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1791.0	lot14997	2.0	[]	[]	(46, 50)
223445	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1507.0	lot14069	8.0	[]	[]	(44, 44)
499159	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,...	710.0	lot31068	12.0	[]	[]	(32, 29)
192460	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 1,...	1187.0	lot12175	17.0	[]	[]	(51, 30)
282210	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	2459.0	lot17275	6.0	[]	[]	(60, 54)
480450	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 1,...	846.0	lot29768	24.0	[]	[]	(33, 33)

```python
#Label -> Num 치환
df['failureNum']=df.failureType
df['trainTestNum']=df.trianTestLabel
mapping_type={'Center':0,'Donut':1,'Edge-Loc':2,'Edge-Ring':3,'Loc':4,'Random':5,'Scratch':6,'Near-full':7,'none':8}
mapping_traintest={'Training':0,'Test':1}
df=df.replace({'failureNum':mapping_type, 'trainTestNum':mapping_traintest})
df.sample(10)
```

	waferMap	dieSize	lotName	waferIndex	trianTestLabel	failureType	waferMapDim	failureNum	trainTestNum
137040	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1801.0	lot8918	15.0	[]	[]	(56, 41)	[]	[]
168047	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	5133.0	lot10709	5.0	[]	[]	(87, 75)	[]	[]
470701	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1513.0	lot28951	24.0	[]	[]	(49, 39)	[]	[]
775651	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1442.0	lot46087	8.0	[[Test]]	[[none]]	(41, 45)	8	1
79221	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	776.0	lot5768	16.0	[]	[]	(30, 34)	[]	[]
136429	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	1801.0	lot8892	20.0	[]	[]	(56, 41)	[]	[]
425622	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	2412.0	lot25566	4.0	[]	[]	(51, 61)	[]	[]
94765	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 1,...	600.0	lot6684	2.0	[]	[]	(26, 30)	[]	[]
490021	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,...	710.0	lot30306	2.0	[]	[]	(32, 29)	[]	[]
789969	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1,...	1187.0	lot46668	6.0	[[Test]]	[[none]]	(51, 30)	8	1

```python
#전처리
def preProcess(failureNum):
  data_with_label = df[(df['failureNum']==failureNum) & (df['waferMapDim'] == (26, 26))]
  data_with_label = data_with_label.reset_index()

  sw = np.ones((1, 26, 26))
  label = list()

  for i in range(len(data_with_label)):
      # skip null label
      if len(data_with_label.iloc[i,:]['failureType']) == 0:
          continue
      sw = np.concatenate((sw, data_with_label.iloc[i,:]['waferMap'].reshape(1, 26, 26)))
      label.append(failureNum)

  x = sw[1:]
  x = x.reshape((x.shape[0], x.shape[1], x.shape[2], 1))
  y = np.array(label).reshape((-1,1))

  return x, y
```

```python
#증량
def imageGen(x, failureNum):

  gen_x = np.zeros((1, 26, 26, 1))

  X_data = x

  datagen = ImageDataGenerator(
    rescale = 1./255, # 모든 이미지 원소값들을 255로 나누기
    rotation_range=50, # 0~25도 사이에서 임의의 각도로 원본이미지를 회전
    zoom_range=0.2, # (1-0.2)~(1+0.2) => 0.8~1.2 사이에서 임의의 수치만큼 확대/축소
    horizontal_flip=True, # 좌우로 뒤집기
    vertical_flip=True,
    fill_mode='nearest',
  )

  datagen.fit(X_data)

  label = list()
  for i in range(0, 10000):
    flow=datagen.flow(_x,_y,batch_size=1, shuffle=True)
    x, y = flow.next()
    gen_x = np.concatenate((gen_x, x), axis=0)
    label.append(failureNum)

  gen_x = gen_x[1:]
  gen_y = np.array(label).reshape((-1,1))
  return gen_x, gen_y
```

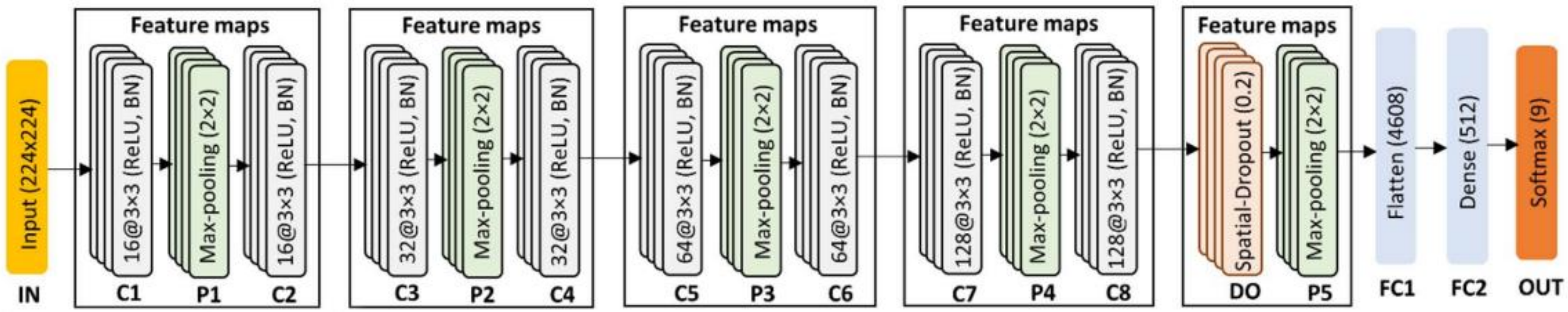
```python
x = np.ones((1, 26, 26, 1))
y = np.ones((1, 1))

#case 0 ~ 8
for i in range(0, 9):
  _x, _y = preProcess(i)
  data_x, data_y = imageGen(_x, i)

  x = np.concatenate((x, data_x), axis=0)
  y = np.concatenate((y, data_y), axis=0)
#flow1=imageGenerator.flow(_x,_y,batch_size=100, shuffle=True)

x = x[1:]
y = y[1:]
```

CNN 구조

CNN 구조

TABLE I
PROPOSED DEEP CNN MODEL PARAMETERS

Layer	Type	Feature Maps	Output Size	Filter size	Padding	Activation
IN	Input	3 (RGB)	224×224	–	–	–
C1	Convolution1	16	222×222	3×3	No	ReLU
P1	Max Pooling1	16	111×111	2×2	No	–
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



*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

과적합을 방지하기 위한 규제화(regulation)

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

학습 방법

딥러닝 학습 조건

학습사양

- Google Colab 사용

CNN 모델 하이퍼 파라미퍼

- optimizer(최적화알고리즘) : Adam Stochastic
- Batch size : 100
- Epoch : 20
- Learning rate : 0.001
- Loss function(손실함수) : categorical crossentropy

CNN 구조

주요 코드 및 실행 결과

- tensorflow, keras 사용

```
from keras.models import Sequential
from keras import optimizers
from keras.layers import Dense, Activation, Flatten, Conv2D, MaxPooling2D
import tensorflow as tf
from keras.layers import Dropout, BatchNormalization
from keras.preprocessing.image import ImageDataGenerator
```

```
#순차적 레이어 층을 더해줌
model = Sequential()

#16@3*3 ReLu, BN (ConV계층) : C1
model.add(Conv2D(filters=16, kernel_size = (3,3), padding ='valid', activation='relu', input_shape = (X_data.shape[1], X_data.shape[2], X_data.shape[3])))
model.add(BatchNormalization())
#MaxPooling 2*2 : P1
model.add(MaxPooling2D(pool_size=(2, 2), padding='valid'))
#16@3*3 ReLu, BN (ConV계층) : C2
model.add(Conv2D(filters=16, kernel_size = (3,3), padding ='same', activation='relu'))
model.add(BatchNormalization())
#32@3*3 ReLu, BN (ConV계층) : C3
model.add(Conv2D(filters=32, kernel_size = (3,3), padding ='same', activation='relu'))
model.add(BatchNormalization())
#MaxPooling 2*2 : P2
model.add(MaxPooling2D(pool_size=(2, 2), padding='valid'))
#32@3*3 ReLu, BN (ConV계층) : C4
model.add(Conv2D(filters=32, kernel_size = (3,3), padding ='same', activation='relu'))
model.add(BatchNormalization())
#64@3*3 ReLu, BN (ConV계층) : C5
model.add(Conv2D(filters=64, kernel_size = (3,3), padding ='same', activation='relu'))
model.add(BatchNormalization())
```

CNN 구조

주요 코드 및 실행 결과

```
#MaxPooling 2*2 : P3
model.add(MaxPooling2D(pool_size=(2, 2), padding='valid'))
#64@3*3 ReLu, BN (ConV계층) : C6
model.add(Conv2D(filters=64, kernel_size = (3,3), padding ='same', activation='relu'))
model.add(BatchNormalization())
#128@3*3 ReLu, BN (ConV계층) : C7
model.add(Conv2D(filters=128, kernel_size = (3,3), padding ='same', activation='relu'))
model.add(BatchNormalization())
#MaxPooling 2*2 : P4
model.add(MaxPooling2D(pool_size=(2, 2), padding='valid'))
#128@3*3 ReLu, BN (ConV계층) :C8
model.add(Conv2D(filters=128, kernel_size = (3,3), padding ='same', activation='relu'))
model.add(BatchNormalization())
#Dropout 0.2
model.add(Dropout(0.2))
#Flatten
model.add(Flatten())
#Dense 512 : FC1
model.add(Dense(512, activation='relu'))
#SoftMax 9 : FC2
model.add(Dense(9, activation='softmax'))
```

```
#최적화 알고리즘
adam = tf.optimizers.Adam(lr = 0.001)
#손실 함수
model.compile(loss = 'categorical_crossentropy', optimizer = adam, metrics = ['accuracy'])

# 모델 훈련
history = model.fit(X_train, y_train,
                    batch_size = 100,
                    epochs = 20,
                    verbose = 2,
                    validation_data = (X_test, y_test))
```

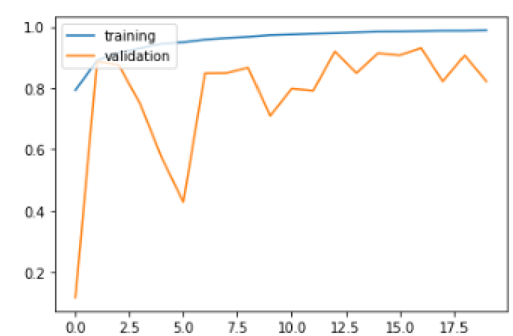
CNN 구조

주요 코드 및 실행 결과

- 총 1시간 46분 소요 epochs 20

```
Epoch 1/20
630/630 - 330s - loss: 0.5471 - accuracy: 0.7930 - val_loss: 5.7678 - val_accuracy: 0.1177 - 330s/epoch - 525ms/step
Epoch 2/20
630/630 - 330s - loss: 0.2956 - accuracy: 0.8896 - val_loss: 0.3045 - val_accuracy: 0.8866 - 330s/epoch - 523ms/step
Epoch 3/20
630/630 - 324s - loss: 0.2291 - accuracy: 0.9155 - val_loss: 0.3562 - val_accuracy: 0.8756 - 324s/epoch - 514ms/step
Epoch 4/20
630/630 - 327s - loss: 0.1936 - accuracy: 0.9297 - val_loss: 0.9597 - val_accuracy: 0.7480 - 327s/epoch - 519ms/step
Epoch 5/20
630/630 - 325s - loss: 0.1534 - accuracy: 0.9442 - val_loss: 1.9315 - val_accuracy: 0.5720 - 325s/epoch - 516ms/step
Epoch 6/20
630/630 - 326s - loss: 0.1386 - accuracy: 0.9495 - val_loss: 7.6670 - val_accuracy: 0.4276 - 326s/epoch - 517ms/step
Epoch 7/20
630/630 - 324s - loss: 0.1173 - accuracy: 0.9579 - val_loss: 0.6298 - val_accuracy: 0.8477 - 324s/epoch - 515ms/step
Epoch 8/20
630/630 - 323s - loss: 0.1023 - accuracy: 0.9632 - val_loss: 0.5988 - val_accuracy: 0.8486 - 323s/epoch - 513ms/step
Epoch 9/20
630/630 - 322s - loss: 0.0935 - accuracy: 0.9670 - val_loss: 0.7790 - val_accuracy: 0.8663 - 322s/epoch - 512ms/step
Epoch 10/20
630/630 - 316s - loss: 0.0772 - accuracy: 0.9726 - val_loss: 1.5495 - val_accuracy: 0.7088 - 316s/epoch
Epoch 11/20
630/630 - 310s - loss: 0.0726 - accuracy: 0.9748 - val_loss: 0.8647 - val_accuracy: 0.7979 - 310s/epoch
Epoch 12/20
630/630 - 313s - loss: 0.0641 - accuracy: 0.9772 - val_loss: 1.1183 - val_accuracy: 0.7914 - 313s/epoch
Epoch 13/20
630/630 - 312s - loss: 0.0600 - accuracy: 0.9790 - val_loss: 0.3366 - val_accuracy: 0.9194 - 312s/epoch
Epoch 14/20
630/630 - 316s - loss: 0.0534 - accuracy: 0.9816 - val_loss: 0.7763 - val_accuracy: 0.8487 - 316s/epoch
Epoch 15/20
630/630 - 311s - loss: 0.0459 - accuracy: 0.9838 - val_loss: 0.3896 - val_accuracy: 0.9136 - 311s/epoch
Epoch 16/20
630/630 - 311s - loss: 0.0447 - accuracy: 0.9843 - val_loss: 0.4138 - val_accuracy: 0.9070 - 311s/epoch
Epoch 17/20
630/630 - 313s - loss: 0.0415 - accuracy: 0.9855 - val_loss: 0.2909 - val_accuracy: 0.9306 - 313s/epoch
Epoch 18/20
630/630 - 311s - loss: 0.0390 - accuracy: 0.9866 - val_loss: 0.9001 - val_accuracy: 0.8213 - 311s/epoch
Epoch 19/20
630/630 - 310s - loss: 0.0380 - accuracy: 0.9867 - val_loss: 0.3653 - val_accuracy: 0.9060 - 310s/epoch
Epoch 20/20
630/630 - 312s - loss: 0.0350 - accuracy: 0.9880 - val_loss: 1.1293 - val_accuracy: 0.8216 - 312s/epoch
```

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training', 'validation'], loc = 'upper left')
plt.show()
```




```
results = model.evaluate(X_test, y_test)
print('Test accuracy: ', results[1])
```

```
844/844 [==============================] - 33s 39ms/step - loss: 1.1293 - accuracy: 0.8216
Test accuracy:  0.8216296434402466
```

CNN 구조

주요 코드 및 실행 결과

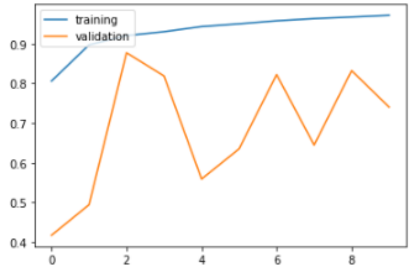
- 총 1시간 소요 epochs 10

[11]

```
Epoch 1/10
720/720 - 367s - loss: 0.5183 - accuracy: 0.8061 - val_loss: 1.9119 - val_accuracy: 0.4170 - 367s/epoch - 509ms/step
Epoch 2/10
720/720 - 372s - loss: 0.2794 - accuracy: 0.8973 - val_loss: 4.0533 - val_accuracy: 0.4940 - 372s/epoch - 517ms/step
Epoch 3/10
720/720 - 368s - loss: 0.2211 - accuracy: 0.9207 - val_loss: 0.4237 - val_accuracy: 0.8774 - 368s/epoch - 511ms/step
Epoch 4/10
720/720 - 366s - loss: 0.1886 - accuracy: 0.9306 - val_loss: 0.6135 - val_accuracy: 0.8188 - 366s/epoch - 509ms/step
Epoch 5/10
720/720 - 371s - loss: 0.1540 - accuracy: 0.9436 - val_loss: 3.0143 - val_accuracy: 0.5589 - 371s/epoch - 515ms/step
Epoch 6/10
720/720 - 370s - loss: 0.1364 - accuracy: 0.9504 - val_loss: 2.0974 - val_accuracy: 0.6350 - 370s/epoch - 515ms/step
Epoch 7/10
720/720 - 370s - loss: 0.1167 - accuracy: 0.9581 - val_loss: 0.7827 - val_accuracy: 0.8224 - 370s/epoch - 514ms/step
Epoch 8/10
720/720 - 374s - loss: 0.0997 - accuracy: 0.9636 - val_loss: 2.8506 - val_accuracy: 0.6446 - 374s/epoch - 519ms/step
Epoch 9/10
720/720 - 371s - loss: 0.0909 - accuracy: 0.9677 - val_loss: 0.7290 - val_accuracy: 0.8324 - 371s/epoch - 515ms/step
Epoch 10/10
720/720 - 371s - loss: 0.0773 - accuracy: 0.9719 - val_loss: 1.7237 - val_accuracy: 0.7402 - 371s/epoch - 515ms/step
```

[12]

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['training', 'validation'], loc = 'upper left')
plt.show()
```




```
results = model.evaluate(X_test, y_test)
print('Test accuracy: ', results[1])
```

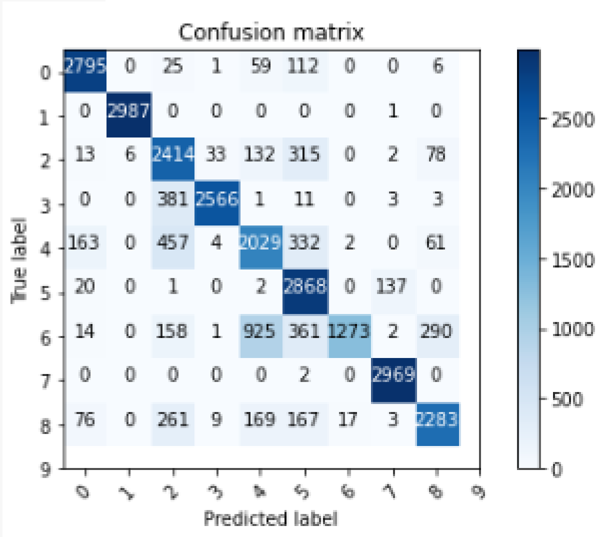
```
563/563 [==============================] - 25s 43ms/step - loss: 1.7237 - accuracy: 0.7402
Test accuracy:  0.7401666641235352
```

결과 및 토의

성능 평가

- Confusion matrix

```python
import numpy as np
import matplotlib.pyplot as plt

# Look at confusion matrix

def plot_confusion_matrix(cm, classes, normalize=False, title='Confusion matrix', cmap=plt.cm.Blues)
  plt.imshow(cm, interpolation='nearest', cmap=cmap)
  plt.title(title)
  plt.colorbar()
  tick_marks=np.arange(len(classes))
  plt.xticks(tick_marks, classes, rotation=45)
  plt.yticks(tick_marks, classes)

  if normalize:
    cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

  thresh = cm.max() / 2.
  for i in range (cm.shape[0]):
    for j in range (cm.shape[1]):
      plt.text(j, i, cm[i, j], horizontalalignment="center", color="white" if cm[i, j] > thresh else

  plt.tight_layout()
  plt.ylabel('True label')
  plt.xlabel('Predicted label')
  plt.figure(figsize=(20,20))

# Predict the values from the validation datase
Y_pred = model.predict(X_test)
# Convert predictions classes to one hot vectors
Y_pred_classes = np.argmax(Y_pred, axis = 1)
# Convert validation observations to one hot vectors
Y_true = np.argmax(y_test, axis = 1)
# compute the confusion matrix
confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
# plot the confusion matrix
plot_confusion_matrix(confusion_mtx, classes = range(10))
```



'Center':0,'Donut':1,'Edge-Loc':2,'Edge-Ring':3,
'Loc':4,'Random':5,'Scratch':6,'Near-full':7,'none':8

감사합니다