Deep Learning

Wafer Map Classification (Anomaly Detection)

U Kang
Seoul National University

In This Project

- Imbalanced multiclass classification
- Anomaly detection in wafer maps
- Implement a deep learning model using TensorFlow

Outline

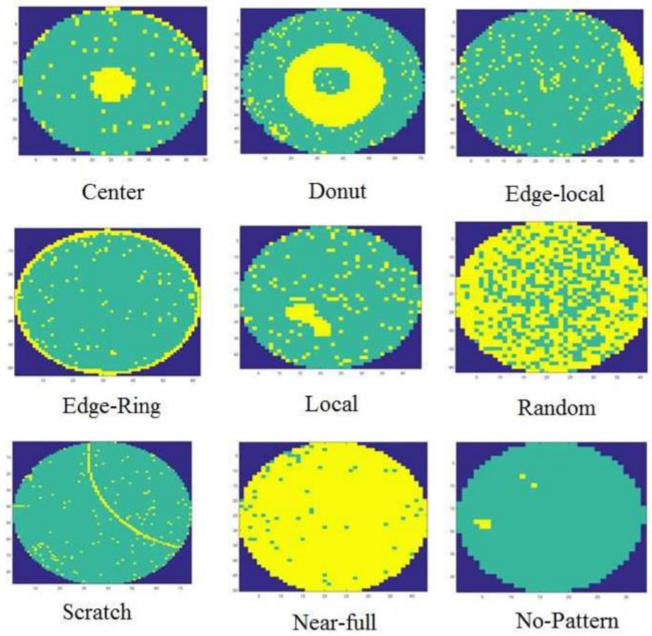
- ☐ **Introduction**
- ☐ Data
- ☐ Preprocessing Codes

Motivation

- Wafer map analysis is critical in semiconductor manufacturing operations
  - It provides visual details that are crucial for identifying failures
  - However, it is time-consuming to identify them manually
- How can we train a model that detects **anomalies** and their **abnormality types** in wafer maps?

Goals

- Classify the wafer maps into one of nine categories
  - "No-pattern” indicates a normal wafer map
  - The others indicate the types of abnormality



Problem Definition

- ***Given***
  - Various types of **wafer maps**
- ***Goal***
  - **Classify** the wafer maps into correct categories
- ***Requirement***
  - Precision and recall of each category are important criteria for measurement since the data are **severely imbalanced**

Evaluation

- F1-score
  - It considers both of **precision** and **recall** of classification
  - An evaluation measure for an **imbalanced** dataset

$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}} = 2 \times \frac{p \times r}{(p + r)}$$

$$p = \frac{true\ positive}{true\ positive + false\ positive}$$

$$r = \frac{true\ positive}{true\ positive + false\ negative}$$

  - It is not only for binary classification but also for **multiclass classification**.
  - A model will be evaluated by the **macro F1-score**

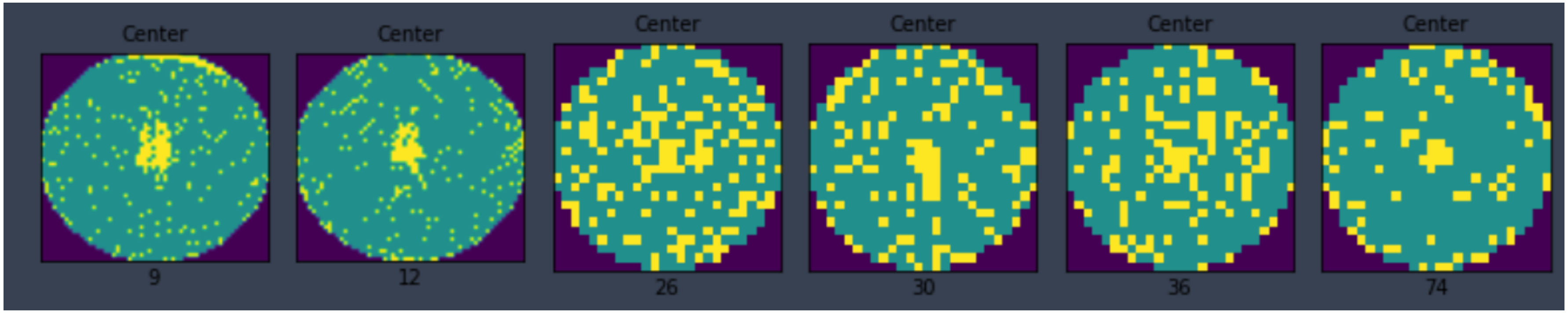
Evaluation

- Macro F1 score
  - Average of per-class F1 score
  - Example: macro F1 = (42.1 + 30.8 + 66.7)/3 = 46.5 %

Class	Precision	Recall	F1-score
Cat	30.8%	66.7%	42.1%
Fish	66.7%	20.0%	30.8%
Hen	66.7%	66.7%	66.7%

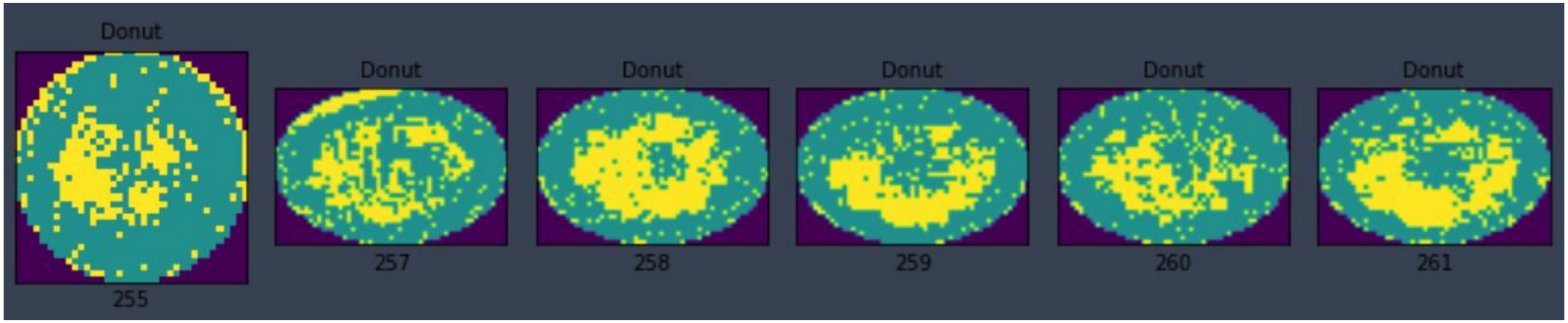
Category description (1)

- Center (abnormal)



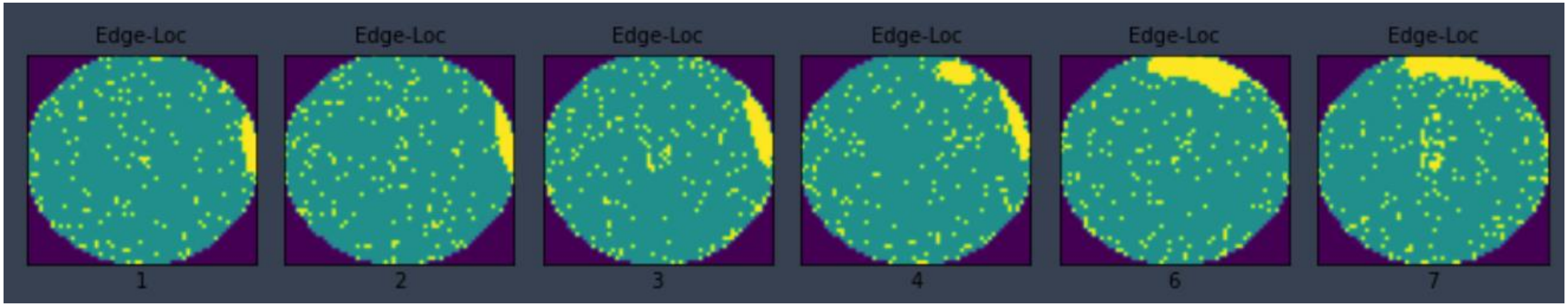
Category description (2)

- Donut (abnormal)



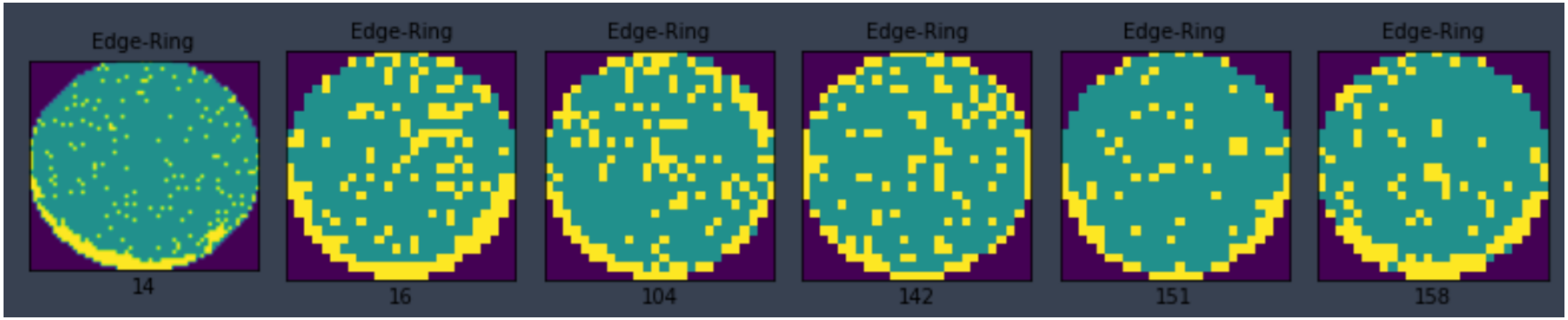
Category description (3)

- Edge-Loc (abnormal)



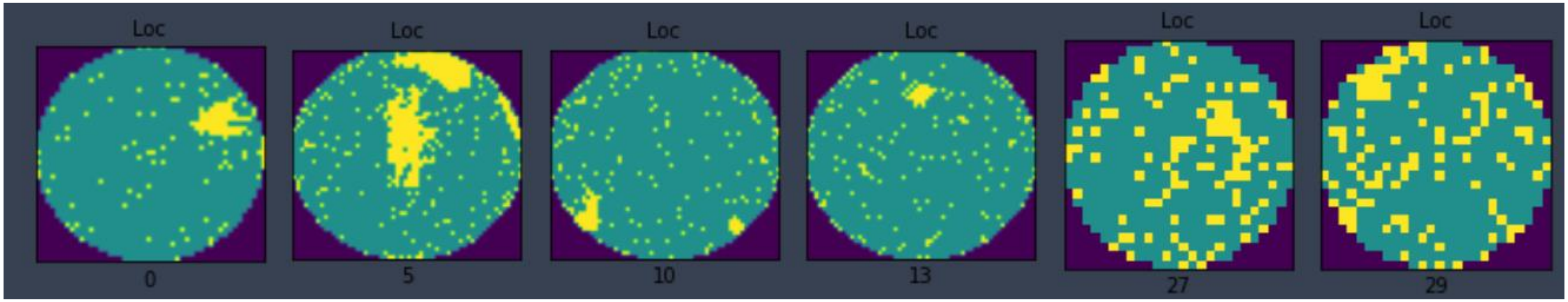
Category description (4)

- Edge-Ring (abnormal)



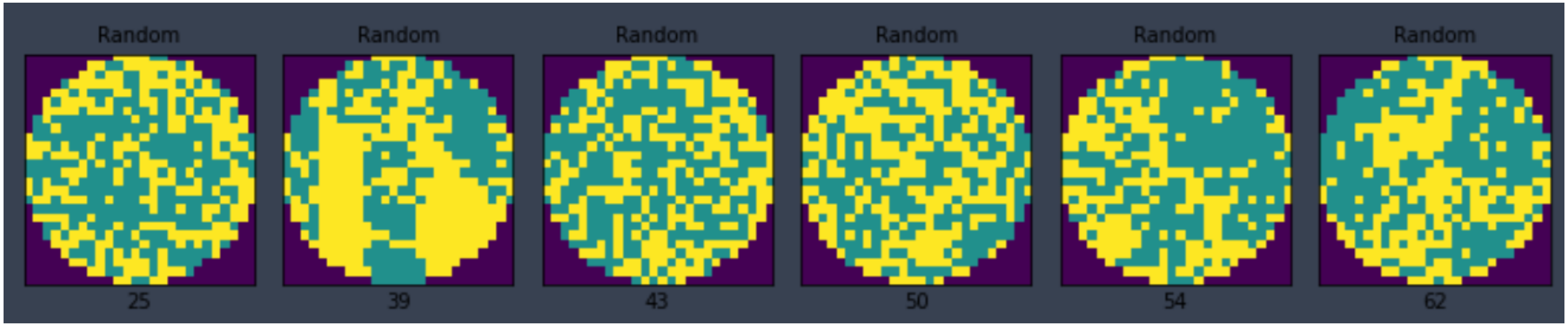
Category description (5)

- Loc (abnormal)



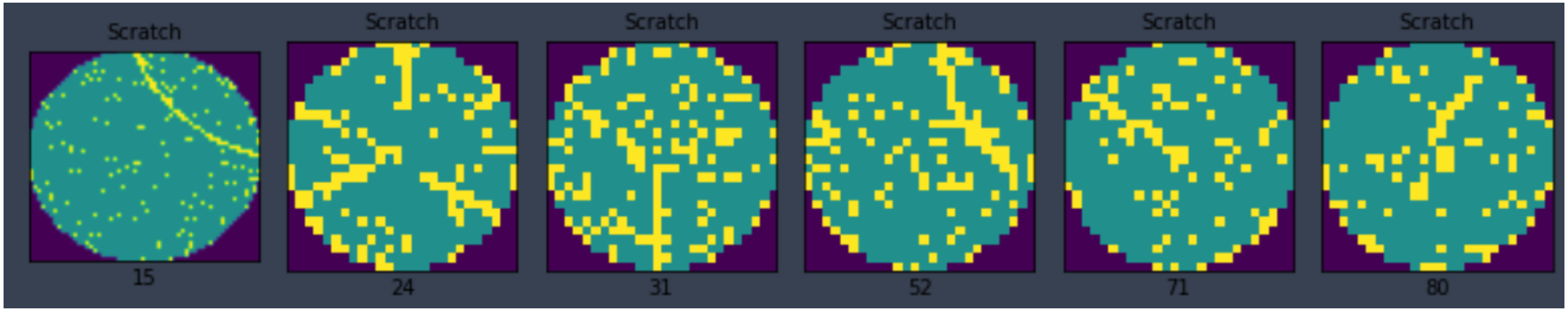
Category description (6)

- Random (abnormal)



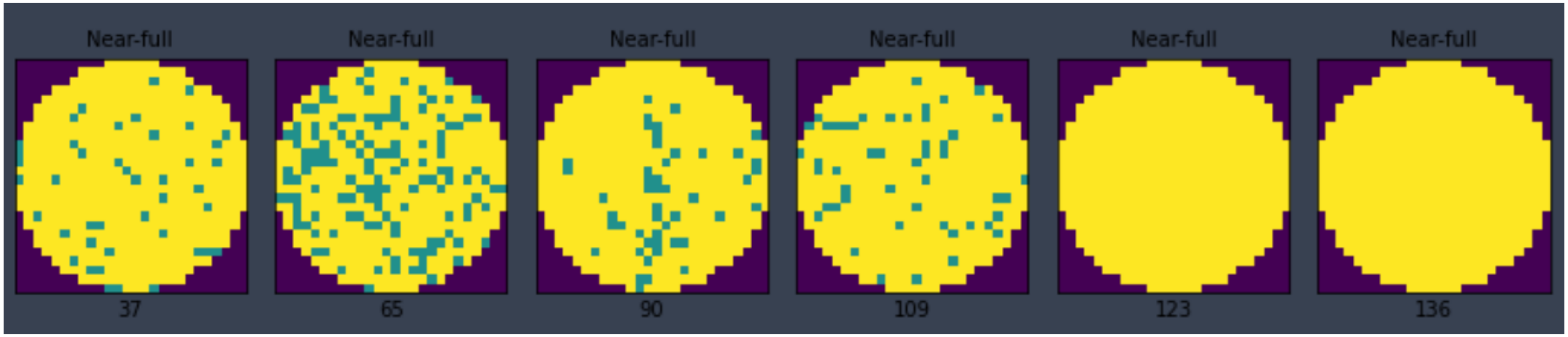
Category description (7)

- Scratch (abnormal)



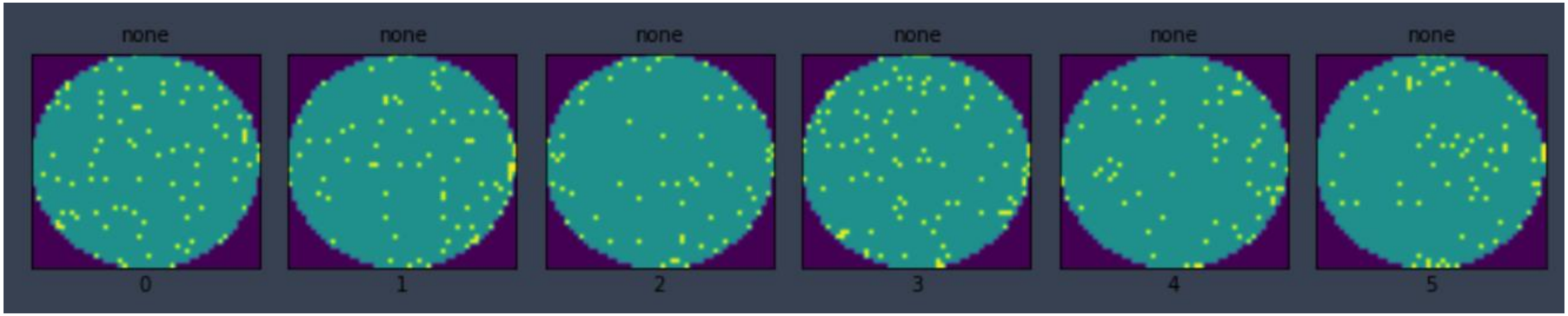
Category description (8)

- Near-full (abnormal)



Category description (9)

- None (normal)



Outline

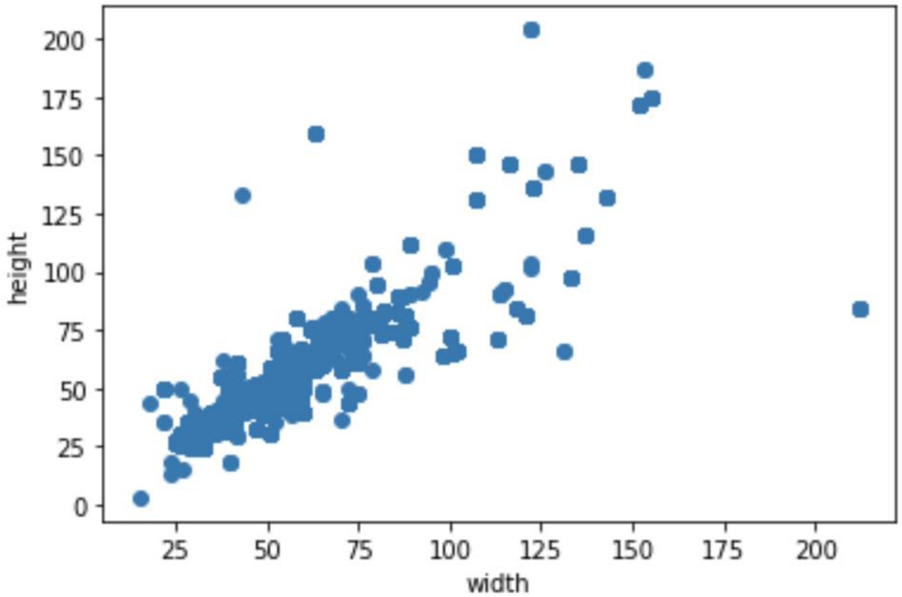
- ☑ Introduction
- ☐ **Data**
- ☐ Preprocessing Codes

Providing data

- Datasets as pickle files
  - We provide two labeled datasets ‘train.pkl’ and ‘test_gt.pkl’ which are for training and test, respectively.
  - We evaluate the model on ‘test_gt.pkl’
    - Note that the evaluation metric is **macro F1-score**

Dataset (1)

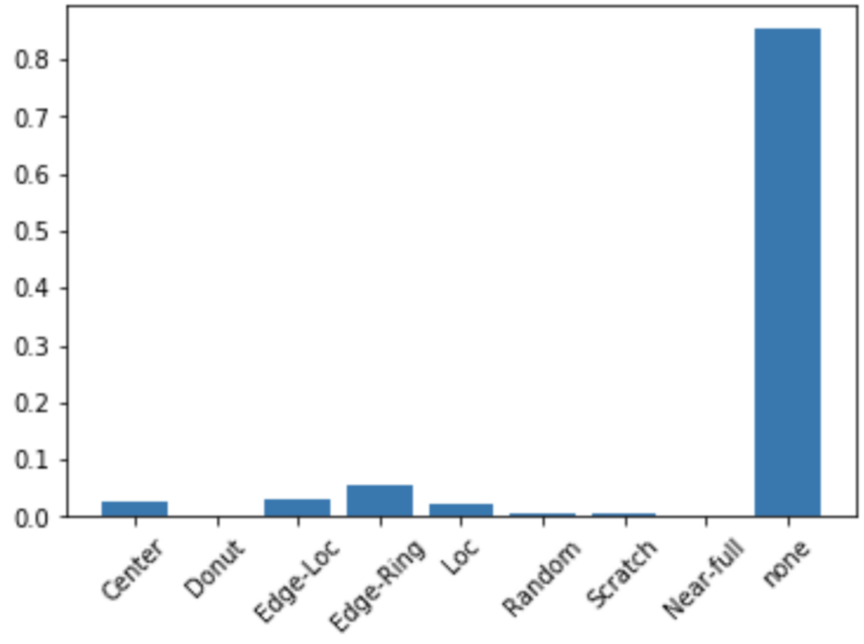
- 143,115 wafer map images
  - 112,294 training examples, 30,821 test examples
- Wafer map images vary in size



- There are nine categories
  - 8 abnormal, 1 normal

Dataset (2)

- There are nine categories
  - "None" indicates a normal wafer map
  - The others indicate the types of abnormality
  - The categories are **severely imbalanced**



Outline

- ☑ Introduction
- ☑ Data
- ☐ **Preprocessing Codes**

Import libraries

- Import the libraries: numpy, pandas, pyplot, and tensorflow

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import tensorflow as tf
```

Load the Dataset

- Load the pickle files

```
import pandas as pd
df_train = pd.read_pickle("./data/train.pkl")
df_test = pd.read_pickle("./data/test_gt.pkl")
```

Explore the Dataset (1)

- Number of instances
  - 112,294 training examples
  - 30,821 test examples
- Each row corresponds to each wafer map

```
df_train
```

	waferMap	failureType
0	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 2,...	[[none]]
1	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[[none]]
2	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[[none]]
3	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2,...	[[none]]
4	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1, 1, 1,...	[[none]]
...	...	...
112289	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 2, 1, 1,...	[[Center]]
112290	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 1, 1, 1,...	[[Edge-Loc]]
112291	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[[none]]
112292	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 1, 2,...	[[none]]
112293	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 1,...	[[none]]

112294 rows × 2 columns

```
df_test
```

	waferMap	failureType
0	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 2,...	[[none]]
1	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 2, 1,...	[[none]]
2	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 2, 2, 2,...	[[none]]
3	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,...	[[none]]
4	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[[none]]
...	...	...
30816	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2,...	[[none]]
30817	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...	[[none]]
30818	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 2,...	[[none]]
30819	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 2, 1,...	[[none]]
30820	[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 1, 2,...	[[none]]

30821 rows × 2 columns

Explore the Dataset (2)

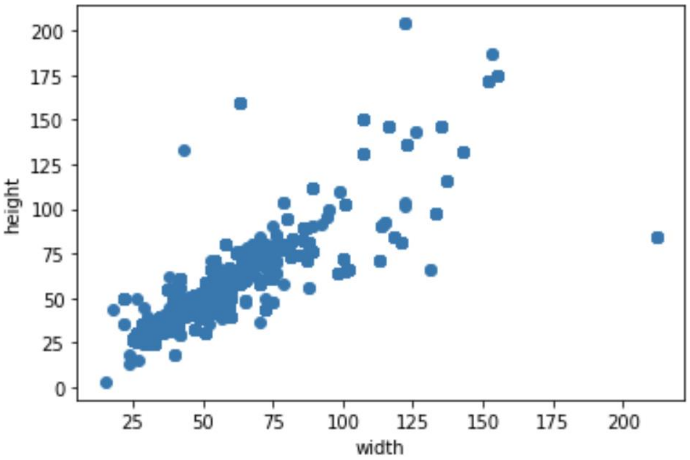
- Attributes explanation
  - waferMap: wafer map represented as (width*height) size numpy array
    - 0: area that dies do not exist
    - 1: area that normal die exists
    - 2: area that defective die exists
  - failureType: type of failure
    - none: normal wafer without defect pattern
    - others: abnormal wafers with their types

Explore the Dataset (3)

- Wafer maps vary in size

```
def find_dim(x):
    dim0=np.size(x,axis=0)
    dim1=np.size(x,axis=1)
    return dim0,dim1
df_train['waferMapDim']=df_train.waferMap.apply(find_dim)
df_test['waferMapDim']=df_test.waferMap.apply(find_dim)
```

```
shapes = df_train.waferMapDim.values
shapes = [[width, height] for (width, height) in shapes]
shapes = np.array(shapes)
plt.scatter(shapes[:, 0], shapes[:, 1])
plt.xlabel('width')
plt.ylabel('height')
plt.show()
```



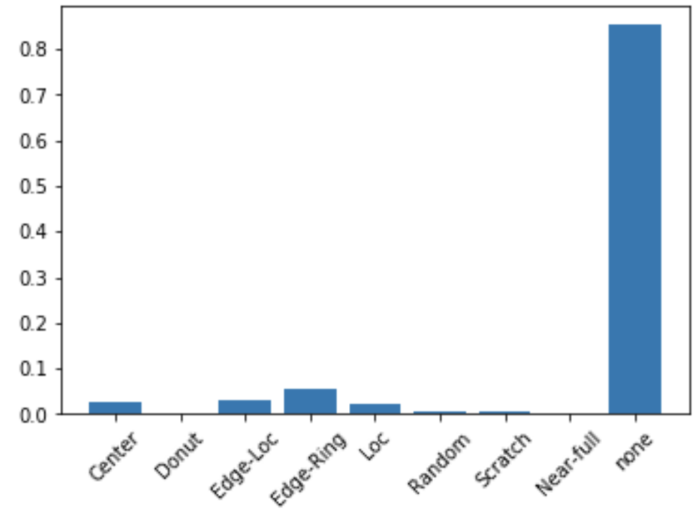
Explore the Dataset (4)

- Each category has different number of examples
  - Severely imbalanced

```
df_train['failureNum']=df_train.failureType
df_test['failureNum']=df_test.failureType
mapping_type={'Center':0, 'Donut':1, 'Edge-Loc':2, 'Edge-Ring':3, 'Loc':4,
              'Random':5, 'Scratch':6, 'Near-full':7, 'none':8}
df_train=df_train.replace({'failureNum':mapping_type})
df_test=df_test.replace({'failureNum':mapping_type})
```

```
uni_pattern = np.unique(df_train.failureNum, return_counts=True)
idxs = uni_pattern[0]
ratios = uni_pattern[1]/df_train.shape[0]
labels = list(mapping_type.keys())

ax = plt.subplot()
plt.bar(idxs, ratios, align='center')
plt.xticks(uni_pattern[0], labels, rotation=45)
plt.show()
```

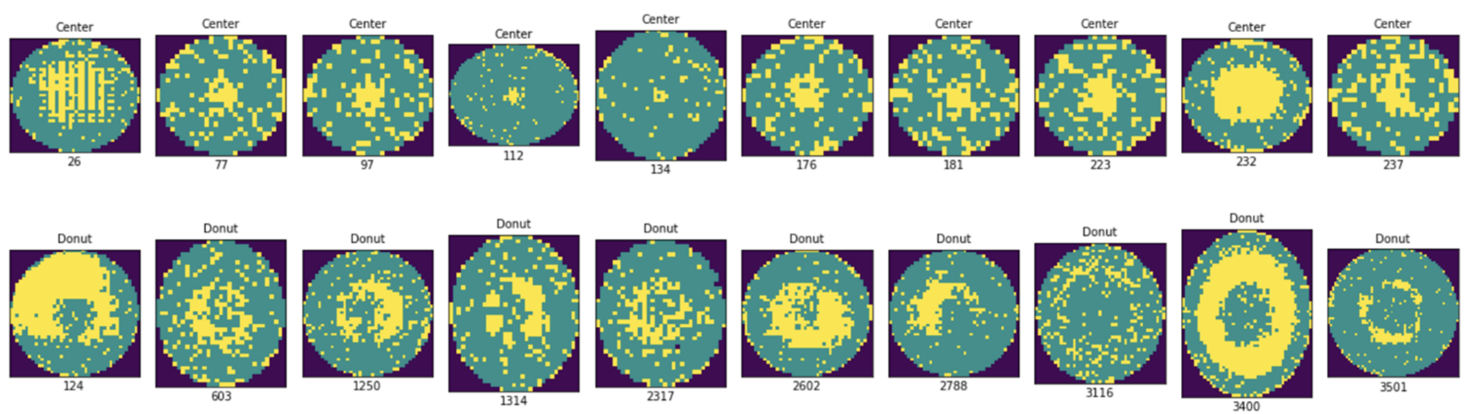


Explore the Dataset (5)

- Visualize the wafer maps

```python
label_name = list(mapping_type.keys())
label_idx = list(mapping_type.values())

for k in label_idx:
    fig, ax = plt.subplots(nrows = 1, ncols = 10, figsize=(18, 12))
    ax = ax.ravel(order='C')
    for j in [k]:
        img = df_train.waferMap[df_train.failureType==label_name[j]]
        for i in range(10):
            ax[i].imshow(img[img.index[i]])
            ax[i].set_title(df_train.failureType[img.index[i]][0][0], fontsize=10)
            ax[i].set_xlabel(df_train
                             .index[img.index[i]], fontsize=10)
            ax[i].set_xticks([])
            ax[i].set_yticks([])
    plt.tight_layout()
    plt.show()
```



Questions?