Deep Learning Project: Classify Defects on Wafer Maps Using Deep Learning

A.A. 2022/23

Aldo Barletta

Gianni Eduard Balbin Canchanya

Fabio Galante

Introduction and Aim of the project

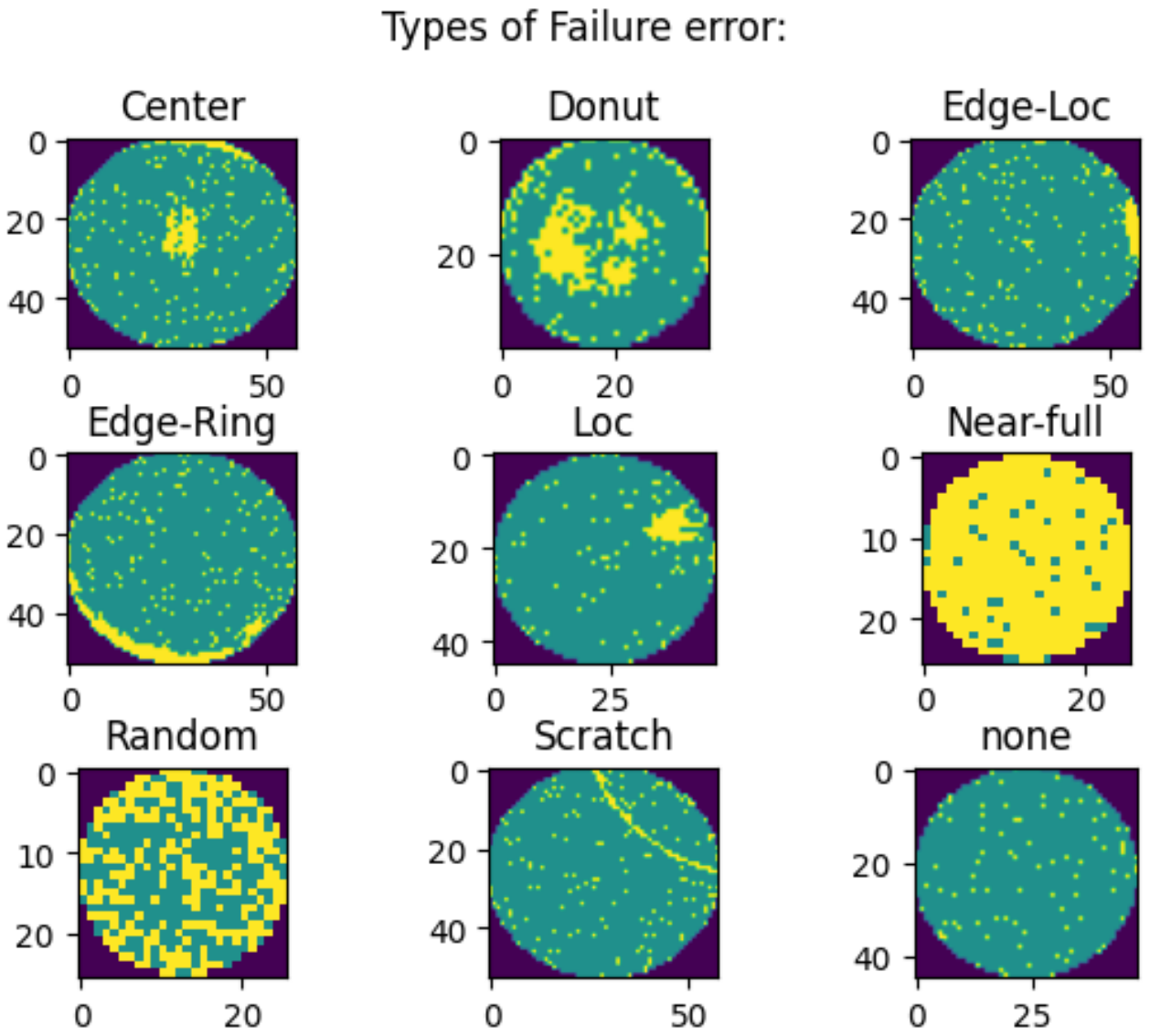
- Wafers are thin disks of semiconducting material used as the foundation for integrated circuits. Automated inspection machines test the performance of individual circuits on the wafer, producing images, called wafer maps, that show pass and fail results.
- We use deep learning classification models which can identify specific issues in the manufacturing process of semiconductor wafers.
- We perform four types of images classification models. On the last two of them we are going to use techniques that are supposed to improve the performance of a classification task.

Dataset

- The dataset has been downloaded from MatLab and it's a pickle file that contains 6 columns and 811457 observations, but only 172950 entries are actually labelled.
- For this project we are going to use only two attributes which are *failureType* and *waferMap.*
- The first one refers to 9 types of level defects, namely: *Center*, *Donut*, *Edge-Loc*, *Edge-Ring*, *Loc*, *Near-Full*, *Random*, *Scratch and none.*
- The last attribute contains arrays of *uint8* type which refer to the wafer images of their respective label.

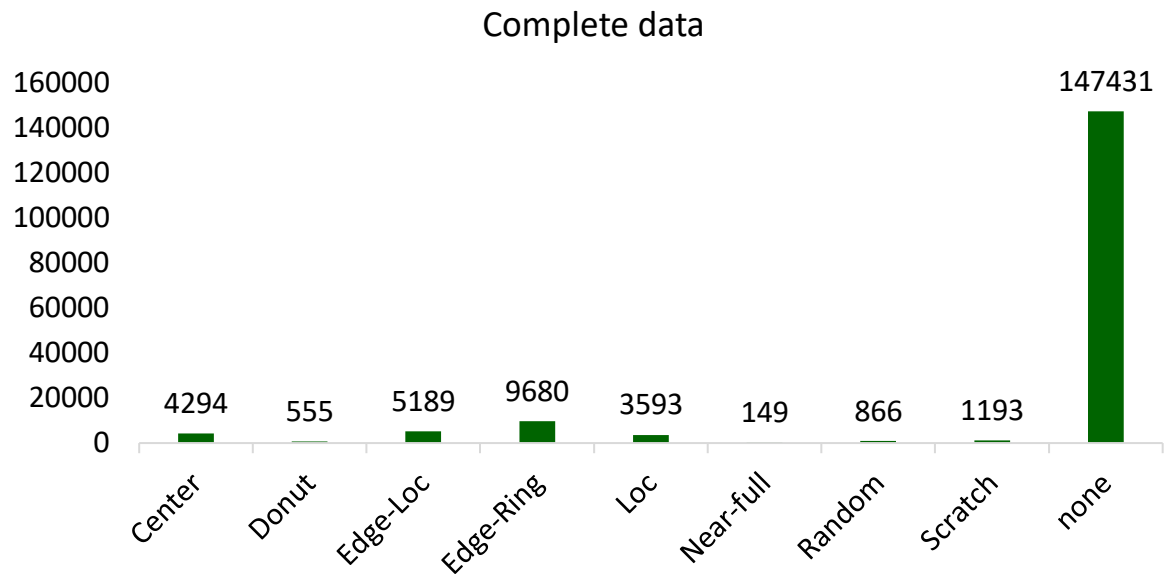
Data Exploration

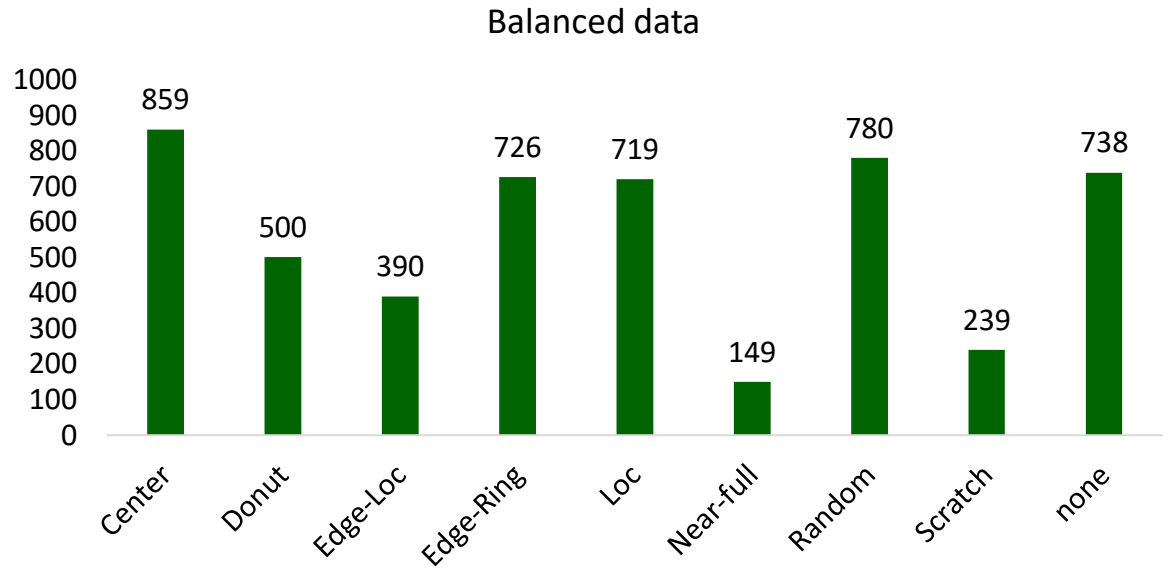
- Now we show the types of possible defects that the automated inspection machines could detect on the wafer and which type of wafer we want to have, that is *none*.



Preprocessing

- As previously said, we are going to use only 172 950 observations of the original dataset.
- The first plot shows the frequency of each failure type and, as it is possible to see, about 85% of the total entries refer to the *none* type. So, the dataset is heavily imbalanced.
- Therefore, we managed to balance the observations for each type, as shown in the second bar chart, thus obtaining 5100 final observations to use for image classification.





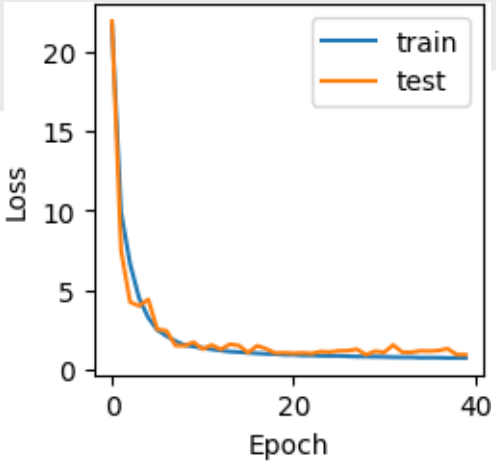
Model 1

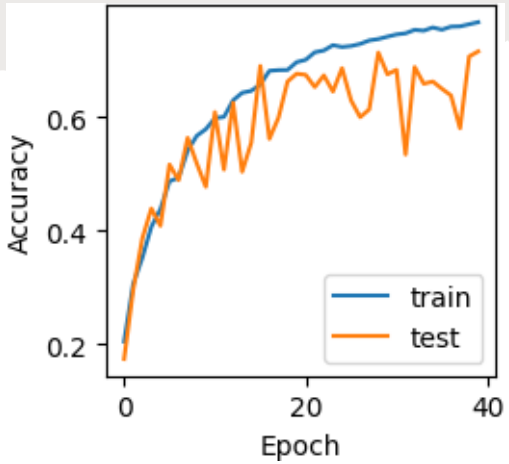
- The dataset was divided into two parts: a training dataset and a validation dataset, with a ratio of 80% for training and 20% for validation.
- We are going to use a Convolutional Neural Network (CNN) architecture designed to capture features through convolutional and pooling layers, enabling precise predictions across multiple classes.
- It consists of two convolutional layers. The first layer has 32 kernels, while the second layer has 64 kernels, both with a size of 3x3. Following each convolutional layer, we apply the *ReLU* activation function. This introduces non-linearity, enabling the network to learn intricate and complex patterns from the data.

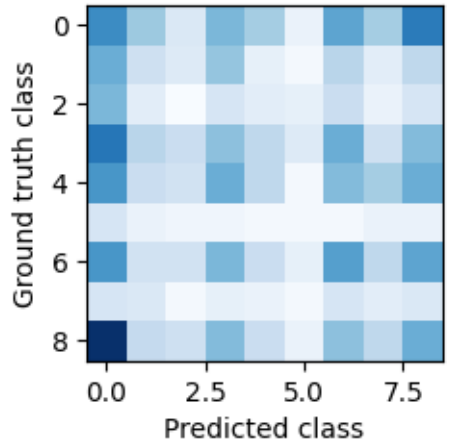
Model 2

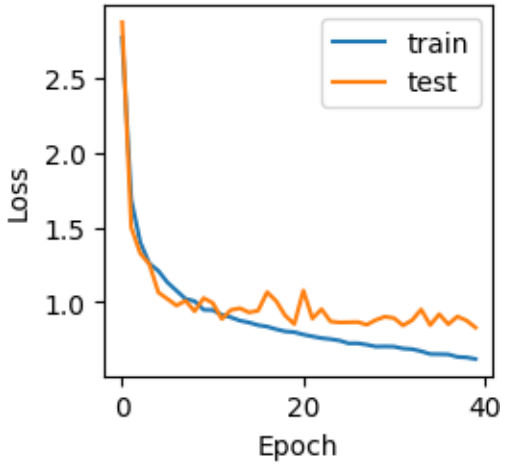
- The second model we use is basically the same as the previous one, the only difference is the addition of a *Batch Normalization*
- It is a technique used to improve the training process and performance of neural networks, normalizing the activations of each layer. By doing so, we manage to have more stable and faster training because networks converge quicker.

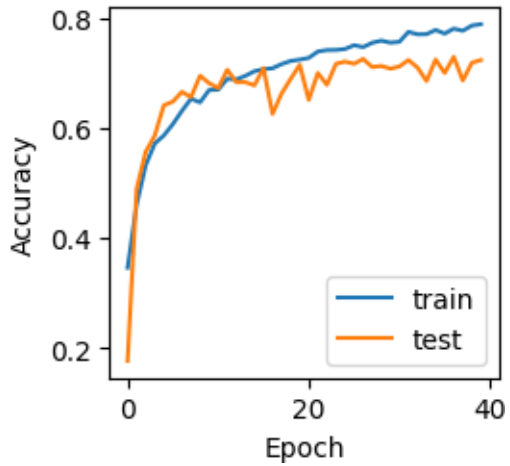
Comparison between the two models‘ performances

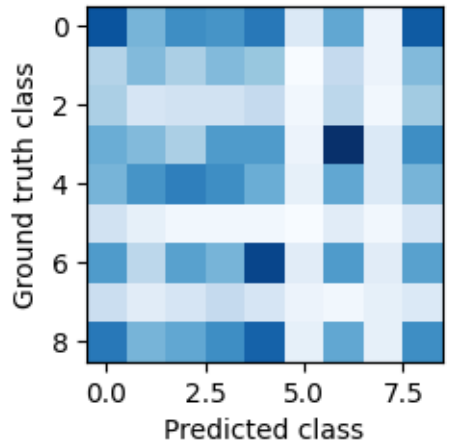








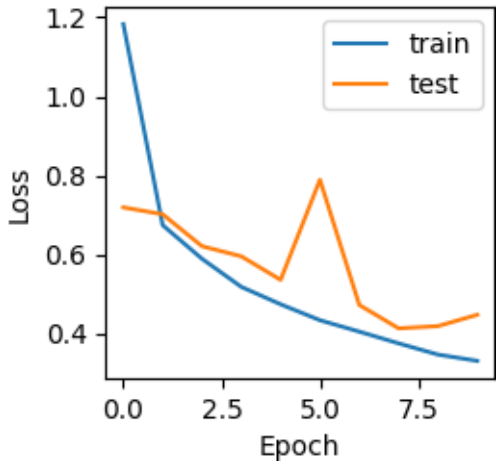


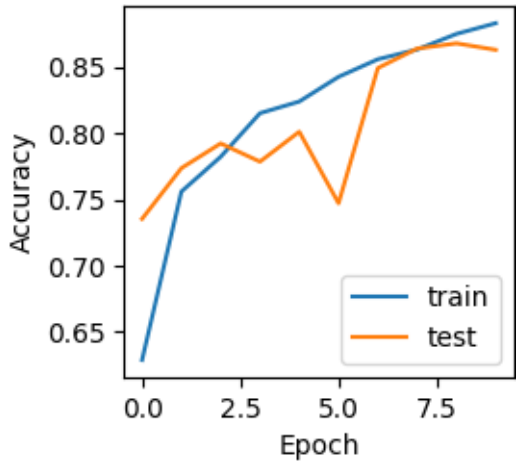


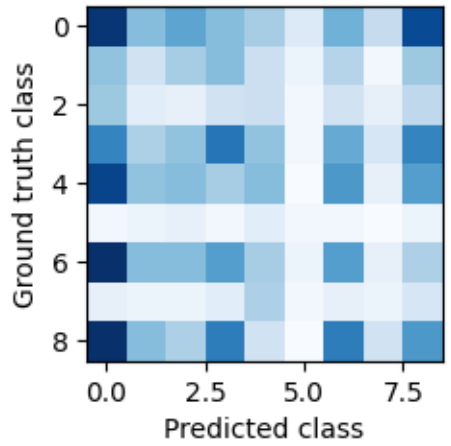
Model 3

- The third model we use exploits the technique of *Transfer Learning*.
- We used the model *MobileNetV2*, that is a model that takes into account a convolutional neural network pretrained on a high size dataset, that extracts weights that contain useful information on how to recognize meaningful features about images. It has a really low parameter count that guarantees speed and efficiency.

Model 3‘s performance



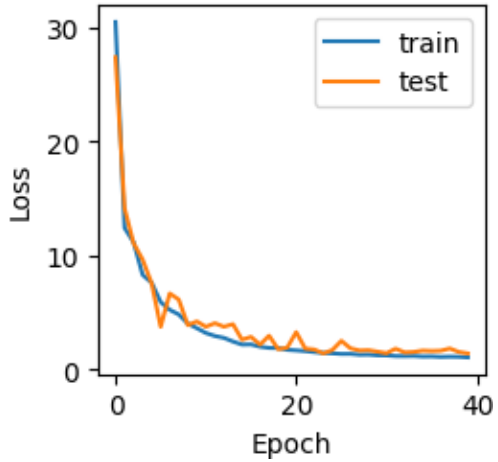


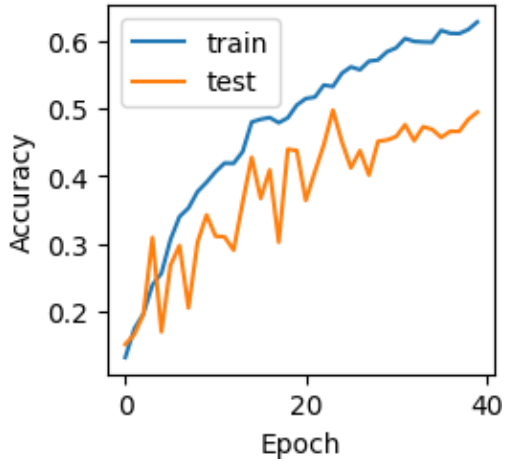


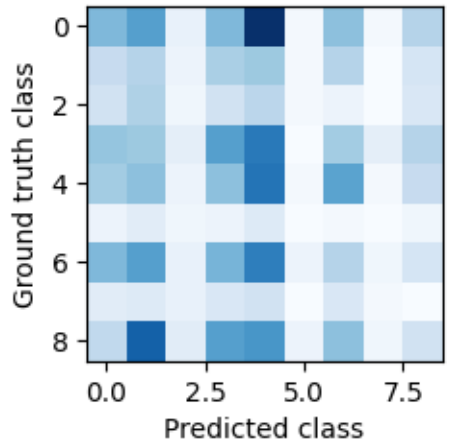
Model 4

- The last model we used implements a technique called *Data Augmentation*.
- This technique aims at exposing the model to different variations of the train dataset, in order to represent a possible real life-scenario. To do that various transformations are applied to the data such as flipping, cropping or adding noise.

Model 4‘s performance







Classification Report Comparison

Classification report of model 1:

	precision	recall	f1-score	support
0	0.13	0.18	0.15	175
1	0.11	0.10	0.11	96
2	0.00	0.00	0.00	66
3	0.14	0.13	0.14	154
4	0.17	0.09	0.11	149
5	0.04	0.04	0.04	24
6	0.18	0.18	0.18	150
7	0.06	0.11	0.08	44
8	0.15	0.15	0.15	162
accuracy			0.13	1020
macro avg	0.11	0.11	0.11	1020
weighted avg	0.13	0.13	0.13	1020

Classification report of model 2:

	precision	recall	f1-score	support
0	0.21	0.18	0.19	175
1	0.14	0.17	0.15	96
2	0.06	0.11	0.07	66
3	0.15	0.14	0.14	154
4	0.11	0.12	0.12	149
5	0.00	0.00	0.00	24
6	0.15	0.14	0.15	150
7	0.12	0.07	0.09	44
8	0.15	0.14	0.15	162
accuracy			0.14	1020
macro avg	0.12	0.12	0.12	1020
weighted avg	0.14	0.14	0.14	1020

Classification report of model 3:

	precision	recall	f1-score	support
0	0.18	0.21	0.19	175
1	0.08	0.08	0.08	96
2	0.04	0.06	0.05	66
3	0.20	0.18	0.19	154
4	0.16	0.11	0.13	149
5	0.09	0.08	0.09	24
6	0.16	0.15	0.15	150
7	0.07	0.07	0.07	44
8	0.15	0.14	0.15	162
accuracy			0.14	1020
macro avg	0.13	0.12	0.12	1020
weighted avg	0.15	0.14	0.14	1020

Classification report of model 4:

	precision	recall	f1-score	support
0	0.18	0.14	0.16	175
1	0.08	0.17	0.11	96
2	0.06	0.03	0.04	66
3	0.18	0.19	0.19	154
4	0.16	0.26	0.20	149
5	0.00	0.00	0.00	24
6	0.12	0.11	0.11	150
7	0.08	0.02	0.04	44
8	0.12	0.06	0.08	162
accuracy			0.14	1020
macro avg	0.11	0.11	0.10	1020
weighted avg	0.13	0.14	0.13	1020

Conclusion

The best model we obtained is the third one.

The reason is the value of accuracy of about 0.86, that compared to the other is much higher (at least 0.14 better than all the others).

Given that we still cannot say that the third model performs really good since it obtain a very low recall for all the failure type, meaning that the model often fails to correctly classify the observation, as well as every other model we obtained.

A way to improve the model would be increasing the number of images for every class in a way that makes them almost equally balanced, or try other models such as *ResNet50*.

Alternative Experiments

- The original dataset had an attribute classifying each observation as train or test, we decided not to use this attribute since the test's observations were much more than the train's one. Mostly because once separated and used as input of the model, the latter did not work well, so we choose autonomously how to divide the dataset in training and test set,

- If we could solve the problem about low recall and accuracy, obtaining an optimal and improved model, then we could use it to classify the observations in the original dataset that did not have any label, that were 638 507 out of 811 457, as we said in the beginning.