Implementation of Silicon Wafer Defect Classification Web application using Deep Learning

Bura Akshaya
Electrical and Electronics Engineering
Chaitanya Bharathi Institute of Technology
Hyderabad, India
ugs206116_eee.akshaya@cbit.org.in

Vavilala Rohitha Raga
Electrical and Electronics Engineering
Chaitanya Bharathi Institute of Technology
Hyderabad, India
ugs206113_eee.rohitha@cbit.org.in

Gundla Sridhar Reddy
Electrical and Electronics Engineering
Chaitanya Bharathi Institute of Technology
Hyderabad, India
ugs206140_eee.sridhar@cbit.org.in

B. Krishna Chaitanya
Electrical and Electronics Engineering
Chaitanya Bharathi Institute of Technology
Hyderabad, India
krishnachaitanyab_eee@cbit.ac.in

Abstract: **This paper presents a solution for the wafers classification of defected silicon wafers, a critical task in semiconductor manufacturing processes. The classification aims to identify various patterns of defects which are pivotal for quality control. The presented approach in this paper leverages deep learning techniques, employing Convolutional Neural Networks (CNNs) and Autoencoders, to automatically learn discriminative features from wafer map images. The trained model is then serialized into a pickle file for efficient deployment and integration. To democratize access to the solution, a user-friendly web interface is developed using Flask, enabling seamless interaction with the classification model. Users can upload wafer map images through the web page, and in return, receive real-time predictions of the defect class, facilitating quick decision-making in manufacturing processes. The presented method demonstrates promising results, achieving high accuracy in classifying defected silicon wafers across diverse patterns. This contributes significantly to enhancing the efficiency and reliability of semiconductor manufacturing, ultimately leading to improved product quality and reduced production costs.**

Keywords—Defected silicon wafers, Convolutional Neural Networks, Auto encoders, Deep learning, Web interface, Flask, Pickle file, Image classification

I. INTRODUCTION

In today's electronics manufacturing, defect detection and classification of silicon wafers is crucial to ensure the quality and reliability of electronic components. Overlooked or unclassified problems in wafers can cause serious financial losses and lead to product disruption. Traditional classification methods often rely on manual or formal procedures, which are time-consuming, laborious and prone to human error. Therefore, there is a growing need for efficient and effective solutions to identify various types of defects. The early approaches to detecting defect patterns in wafer maps [1],[2],[3],[4] were deemed insufficient for analyzing extensive datasets due to their low accuracy.

Recently, various research teams have adopted innovative strategies like feature extraction [5],[6],[7],[8] and defect clustering [6] to build comprehensive wafer map datasets. Additionally, they have employed traditional machine learning algorithms such as support vector machines [5], [6], k-nearest neighbors [9], and decision tree [7]. An effort to enhance the precision and efficacy of analyzing large wafer map datasets is made by incorporating a convolutional neural network (CNN) model, a deep learning technique, into the work. This paper introduces an approach aimed at addressing the classification of silicon wafer defects by emphasizing the identification of diverse defect patterns. It harnesses the capabilities of deep learning methodologies, specifically neural networks like CNNs and Autoencoders, to extract unique features from wafer segments. This facilitates the learning model in discerning intricate patterns and correlations inherent in various defect types. Then serialized into pickle files for easy distribution and integration into production. In order to deliver the solutions to a wider audience, a user-friendly website using Flask is created. This web interface allows users to upload wafer maps and receive real-time predictions of defect types, allowing participants in the semiconductor manufacturing community to make quick decisions.

II. PROPOSED METHODOLOGY

This paper aims to classify defected silicon wafers using deep learning. Fig. 1 shows complete workflow of the proposed work.

(i) Data Collection: The data collection process is very important for training the classification model accurately. Using the real wafer dataset WM811K from Kaggle [10] a database of wafer maps representing various defects in semiconductor manufacturing. The database curated by Wu et al contains 811,457 images collected from 811,457 plates in 46,293 batches. This data represents the information about the WM811K dataset approximately 21% of the samples were collected with nine different defects such as “Center”, “Donut”, “Edge-Location”, “Edge-Ring”, “Random”, “Location”, “Near-Full”, “Scratch” and “None”.

Every image within the dataset depicts a wafer map, where pixels correspond to the status of individual dies on the wafer. Specifically, the WM811K dataset comprises wafer maps containing details such as the map itself, die count, lot identifier, wafer index within each lot, labeling for training or testing sets, and the type of failure pattern [11]. Fig. 2 illustrates example of wafer map patterns corresponding to each

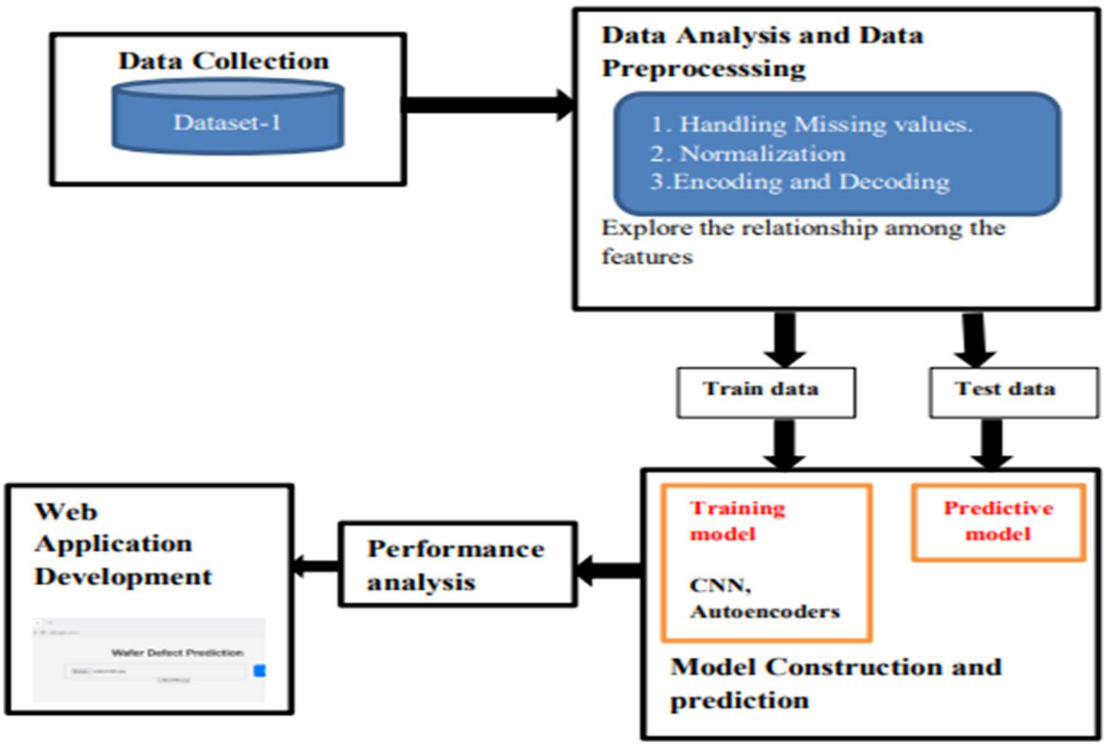


Fig. 1. Work-flow of the presented work

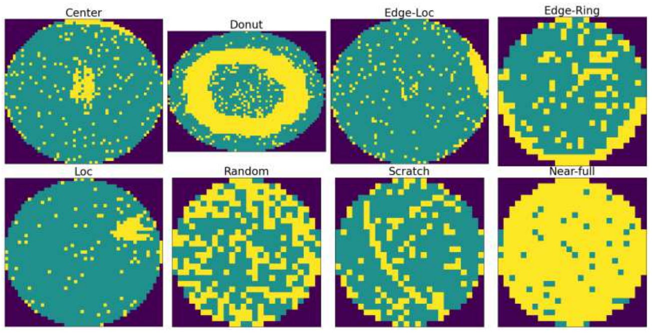


Fig. 2. Sample wafer examples for different pattern types

Table 1. Information regarding WM811K dataset

Label		Amount
Unlabeled	Total	638570
Labeled	Total	172950
	None	147431
	Loc	3593
	Edge-Loc	51589
	Edge-Ring	9680
	Center	4294
	Scratch	1193
	Random	866
	Non-full	149
	Donut	555
Total		**8,11,520**

each pattern type. To build a classification model we have used 3.1% of total WM811K dataset that means we have considered 25,519 samples which consists of patterns and labeled data. Table. 1 represents the information regarding the dataset.

(ii) Data Preprocessing: Start by preprocessing the raw wafer map images to enhance their suitability for deep learning. This process includes actions like resizing images to a uniform size, standardizing pixel intensities, and potentially expanding the dataset through augmentation. Following data augmentation [11], the required data is reduced from 25,519 samples to 19,000 samples.

(iii) Model Building and Prediction: 12,730 samples of the preprocessed data was utilized for training the predictive model, while 6270 samples was used for testing. These samples are of 26x26 dimension. The algorithms used in this work are:

- *Auto Encoders:* Auto encoders are specifically useful for feature extraction [12]. Auto encoders can automatically learn the most important features from the wafer map data without requiring manual feature engineering. This proves especially beneficial when managing intricate patterns or data with high dimensions. By compressing the input wafer map data into a lower dimensional latent space, auto encoders reduce the complexity of the data while retaining important information.

- *Convolution Neutral Networks (CNN):* A convolutional neural network (CNN) is a type of deep learning model designed specifically for processing and analyzing visual data. CNNs [13] can capture minute details and textures present in the wafer maps, which are crucial for distinguishing between normal and defective regions. The network comprises three convolutional layers, each succeeded by Rectified Linear Unit (ReLU) activation functions. These ReLU activations introduce non-linearity into the model, enhancing its capacity to recognize complex patterns. The following flattening layer transforms the output from the convolutional layers into a one-dimensional tensor, simplifying the transition to fully connected layers [14].

Afterward, two dense (fully connected) layers are implemented, containing 512 and 128 neurons respectively, both incorporating ReLU activation functions. [15] These layers aim to extract additional higher-level features from the flattened input. The ultimate output layer comprises nine neurons, representing the nine defect pattern classes, and employs a softmax activation function to generate the probability distribution across these classes.

The model is configured by utilizing the Adam optimizer along with categorical cross-entropy loss function, with accuracy serving as the evaluation metric. This CNN architecture is pivotal as it autonomously extracts pertinent features from the original wafer map images, facilitating precise classification of defective silicon wafers without requiring manual feature engineering.

(iv) Web application development

- *NumPy:* NumPy is a Python library for numerical computing, offering efficient handling of multi-dimensional arrays and matrices. It provides a wide range of mathematical functions for array manipulation and computation, making it a cornerstone in scientific computing, data analysis, and machine learning workflows.

- *Pandas:* Pandas is a Python library that simplifies working with data organized in rows and columns, similar to spreadsheets or databases. It provides powerful data structures like DataFrame and Series, along with functions for handling missing data, merging datasets, and performing statistical operations.

- *Scikit-learn:* Scikit-learn is a Python library that provides a user-friendly interface for utilizing machine learning algorithms, conducting data preprocessing, evaluating models, and performing predictive modeling tasks.

- *Keras*: Keras is a Python-based high-level neural systems API that is built to run seamlessly on top of Tensor Flow. It is designed for rapid experimentation with deep learning models.

- *Tensorflow:* TensorFlow, an open-source framework by Google, streamlines the development, training, and deployment of deep learning models.

- *KerasClassifier:* The KerasClassifier is a class within the Keras library designed to serve as a wrapper, enabling the utilization of Keras models. This integration enables seamless use of Keras models within the scikit-learn ecosystem, facilitating tasks like cross-validation, grid search, and pipeline construction.

This work first instantiates a KerasClassifier object by specifying the model-building function i.e., Convolutional Neural Network (CNN) along with certain hyperparameters such as the number of epochs, batch size, and verbosity level. The function of CNN is assumed to return a compiled Keras model, as it is used to define the architecture and compilation settings of the neural network.

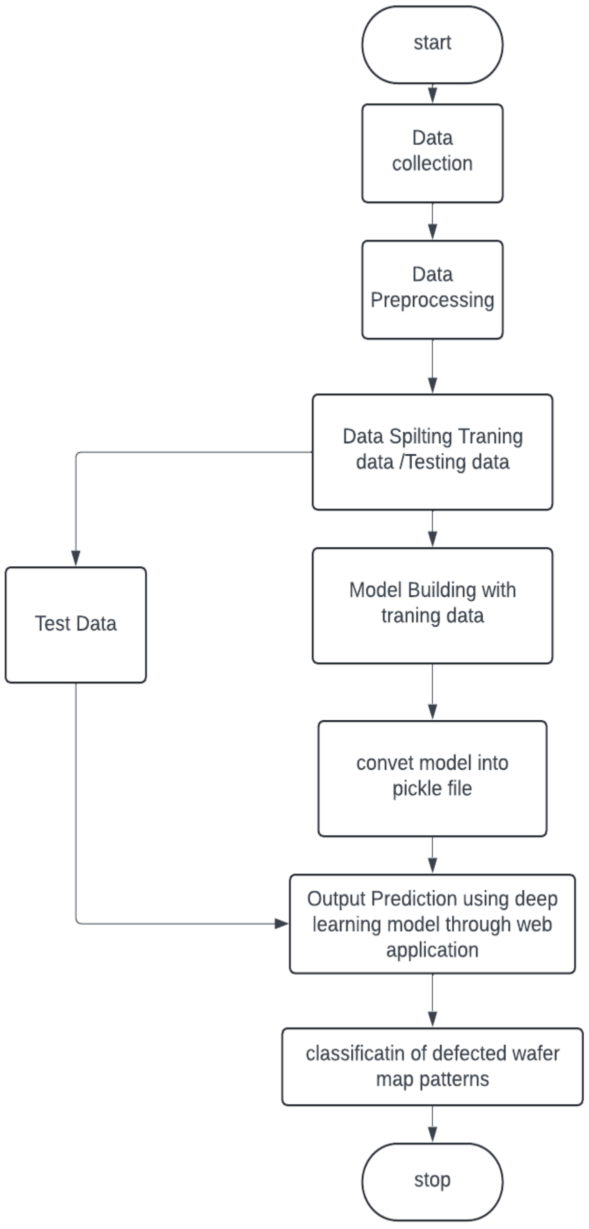

start
Data collection
Data Preprocessing
Data Spilting Traning data /Testing data
Test Data
Model Building with traning data
convet model into pickle file
Output Prediction using deep learning model through web application
classificatin of defected wafer map patterns
stop

Fig. 3. Flowchart

Fig. 3 illustrates the end-to-end workflow for developing and deploying a deep learning model for water map pattern detection, from data collection to model deployment and classification of detected patterns.

- *Cross-validation:* Next, a 3-fold cross-validation is set-up using the K-Fold class from scikit-learn. The training dataset is split into three folds, and for each fold, a model is trained on the training data and evaluated on the validation data. During cross-validation, the KerasClassifier trains and evaluates the model for each fold, keeping track of performance metrics such as accuracy. The cross-validation function calculates the accuracy scores for each fold and returns them as an array results. Finally, the mean accuracy across all folds is computed using numpy library and print the result to the console, then the model is tested using testing dataset providing an estimate of the model's performance through cross-validation.

- *Pickle files:* Pickle files in Python are used for serializing and deserializing Python objects, allowing data to be saved to disk and loaded back into memory. After computing the testing accuracy the pickle file is generated.

- *Flask:* Flask is a Python framework that helped to create web application easily by providing tools for handling web requests.

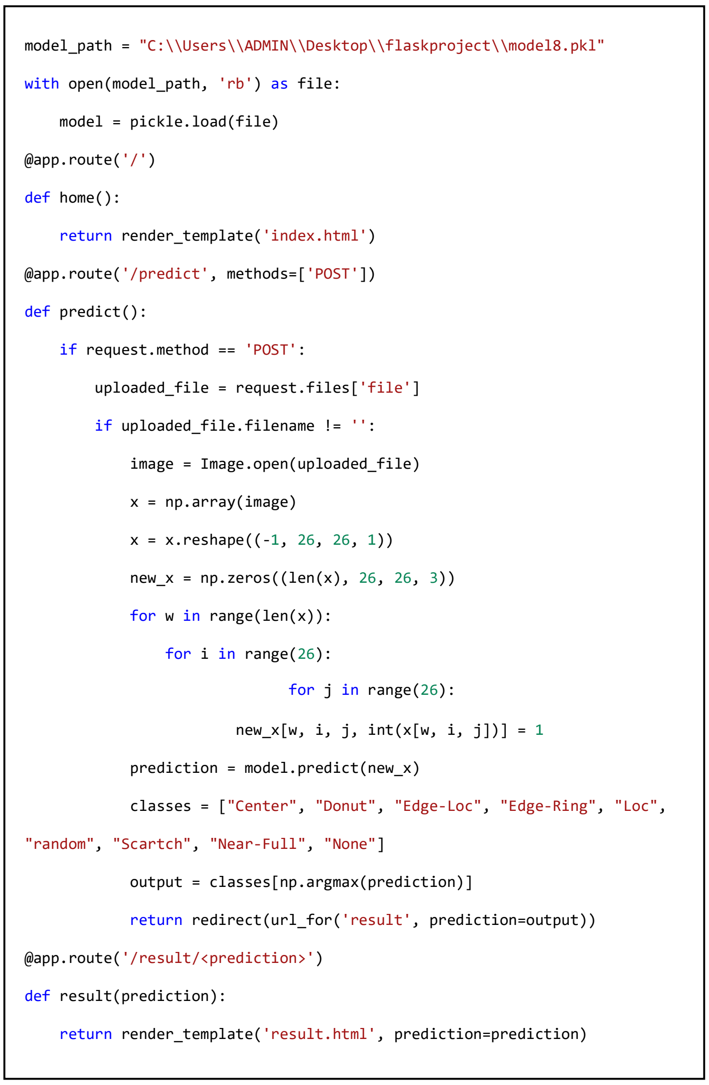
- *VS Code:* Visual Studio Code (VS Code) is a robust source code editor created by Microsoft, known for its light weight design featuring support for numerous programming languages, built-in Git integration, and extensive customization through extensions.

III. Results and Discussion

In this section, after developing the model, results show the achieved testing accuracy of 99% using deep learning classification model, affirming its efficacy in accurately classifying defected silicon wafer map patterns. In addition to testing accuracy, confusion matrix is meticulously evaluated. Fig. 4 sample code of the work presented.

3.1 Implementation of web application:

The integration of developed classification model into a web application is achieved through Flask, demonstrating the practical application of deep learning in real-world scenarios. Model serialization is facilitated using Pickle, ensuring deployment and utilization within the web environment. Meanwhile, the front-end interface is designed using HTML and CSS, providing users with an intuitive and visually appealing platform to interact with the classification model.



```
model_path = "C:\\Users\\ADMIN\\Desktop\\flaskproject\\model8.pkl"
with open(model_path, 'rb') as file:
    model = pickle.load(file)
@app.route('/')
def home():
    return render_template('index.html')
@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        uploaded_file = request.files['file']
        if uploaded_file.filename != '':
            image = Image.open(uploaded_file)
            x = np.array(image)
            x = x.reshape((-1, 26, 26, 1))
            new_x = np.zeros((len(x), 26, 26, 3))
            for w in range(len(x)):
                for i in range(26):
                              for j in range(26):
                        new_x[w, i, j, int(x[w, i, j])] = 1
            prediction = model.predict(new_x)
            classes = ["Center", "Donut", "Edge-Loc", "Edge-Ring", "Loc",
"random", "Scartch", "Near-Full", "None"]
            output = classes[np.argmax(prediction)]
            return redirect(url_for('result', prediction=output))
@app.route('/result/<prediction>')
def result(prediction):
    return render_template('result.html', prediction=prediction)
```

Fig. 4. Sample code

Flask stands out as a Python-based web framework, empowering users to seamlessly integrate application functionalities. Fig. 5 illustrates the fundamental file structures of the developed application, which consists of five distinct program modules. These modules delineate the development process as outlined below as follows:

- ➢ *Model8.pkl*: This file includes the trained deep learning model for predicting defects in silicon wafer maps. The CNN and Autoencoders model achieved an accuracy of 99.2% when utilizing all features. This model has been integrated into the predictive model stored in the "model8.pkl" file.

- ➢ *app.py*: Contained within this package are Flask APIs which receive information on defective silicon wafer maps either via a graphical user interface (GUI) or through API calls. These APIs then utilize our CNN model to compute the predicted value and return it.

- *Templates*: The index.html file within this directory enables users to upload images of defective wafer map patterns, while the result.html file displays the predicted outcome.
- *Cnn-wm.ipynb*: The content of this Jupyter notebook demonstrates the application of Convolutional Neural Network (CNN) and autoencoder algorithms for making predictions on the wafer map dataset.
- *LSWMD.pkl.zip*: This is wafermap dataset which contains information of 811k samples.

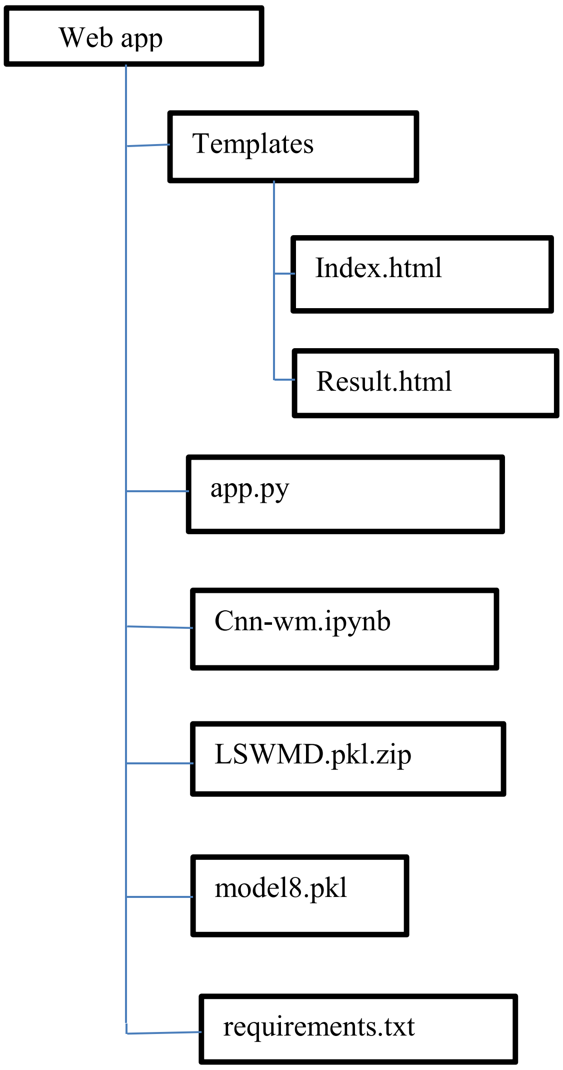
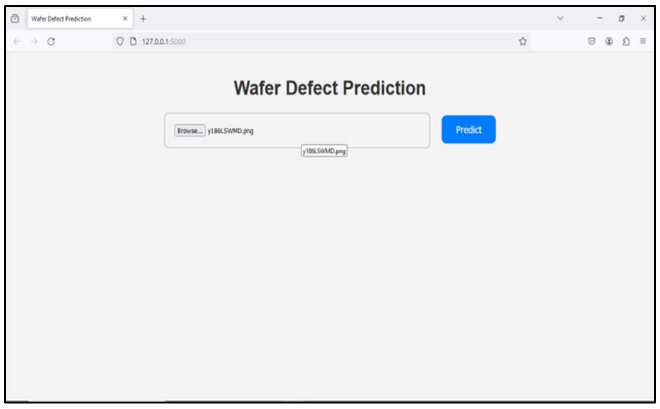


Fig. 5. File structure of the web application

3.2 Prediction results of web application

INPUT: Once the Flask server is up and running, the webpage is accessed by navigating to http://localhost:5000 in our web browser. Once the webpage is opened, we typically see an interface with a button designated for uploading images. Click on the button to drop an image file from computer into this area. After uploading the image, the Flask application will preprocess it to prepare it for input into the trained model. Flask application will use the trained model to make predictions on the uploaded image Fig. 6 depicts the homepage of Web application.


Wafer Defect Prediction

127.0.0.1:5000

Wafer Defect Prediction

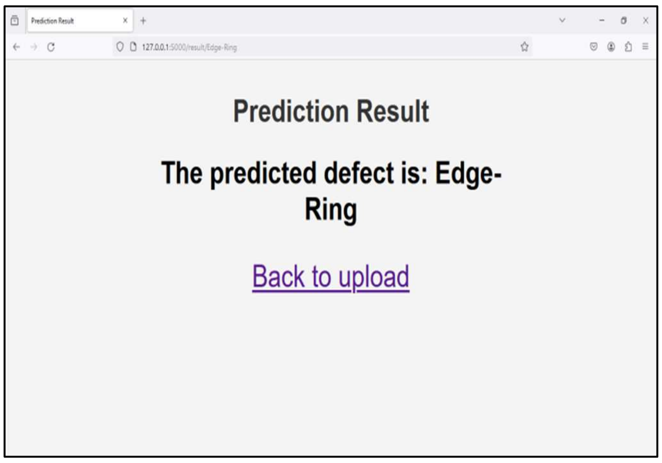
Browse... y186LSWMD.png

Predict

y186LSWMD.png

Fig. 6. Homepage of Web application

OUTPUT: This prediction will classify the image into one of the predefined classes that are "Center", "Donut", "Edge-Location", "Edge-Ring", "Random", "Location", "Near-Ful", "Scratch" and "None", based on the patterns detected in the image. Repeat the process by uploading different images and observing the model's predictions on those images. Fig. 7 shows the result Page of Web Application.


Prediction Result

127.0.0.1:5000/result/Edge-Ring

Prediction Result

The predicted defect is: Edge-Ring

Back to upload

Fig. 7. Result Page of Web Application

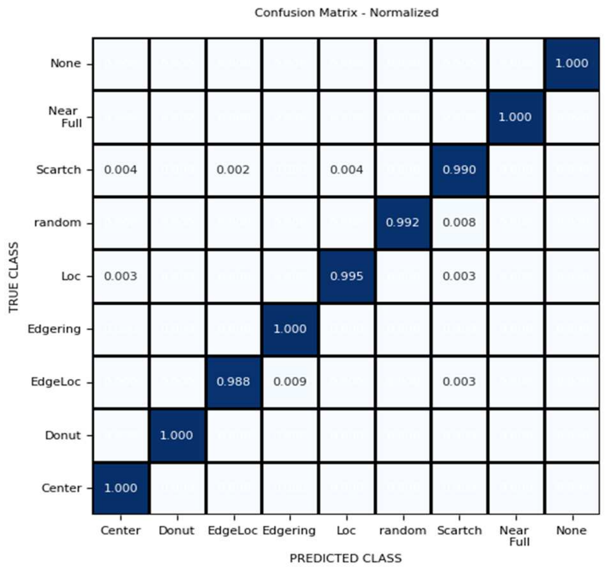


Fig. 8. Confusion matrix of the classification model

The confusion matrix provided in Fig. 8, is a visual representation of the presented model's classification performance.

IV. CONCLUSION

In conclusion, this paper focuses on classifying defected silicon wafers using deep learning, leveraging the WM811K dataset. Here 3.1% of the total dataset is utilized, comprising 25,519 samples, which were consisting of labels and patterns which were preprocessed. After preprocessing, the dataset was reduced to 19,000 samples. A comprehensive deep learning model is developed comprising Convolutional Neural Networks (CNNs), Autoencoders, and a Keras Classifier. This model was trained using 12,730 preprocessed samples and subsequently assessed for its performance using 6,270 test samples. The model achieved an impressive testing accuracy of 99%. Additionally, a meticulous evaluation of the confusion matrix is conducted to gain insights into the model's performance across different defect classes. The classification model was seamlessly integrated into a Flask-based web application, enabling practical deployment. Utilizing Pickle for model serialization ensured efficient utilization within the web environment. The user-friendly front-end interface, crafted with HTML and CSS, facilitated intuitive interaction with the classification model. The testing accuracy attained by presented model underscores its potential for enhancing quality control processes in the industry, leading to improved efficiency and product reliability. Through empirical testing and real-world guidance, this paper demonstrates the effectiveness and reliability of presented approach, paving the way for advancements in management and manufacturing in the semiconductor industry.

REFERENCES

[1] Wang, and Chih-Hsuan, “Recognition of semiconductor defect patterns using spatial filtering and spectral clustering,” Expert Systems with Applications, vol. 34, no. 3, pp. 1914-1923, 2008.

[2] Tao Yuan, Suk Joo Bae and Jong In Park, “Bayesian spatial defect pattern recognition in semiconductor fabrication using support vector clustering,” The International Journal of Advanced Manufacturing Technology, vol. 51, no. 5-8, pp. 671-683, 2010.

[3] Hwang, Jung Yoon, and Way Kuo, “A model-based clustering for integrated circuit yield enhancement,” European Journal of Operational Research, vol. 178, no. 1, pp. 143-153, 2007.

[4] Yuan, Tao, and Way Kuo, “A model-based clustering approach to the recognition of the spatial defect patterns produced during semiconductor fabrication,” IEEE Transactions, vol. 40, no. 2, pp. 93-101, 2007.

[5] Ming-Ju Wu, Jyh-Shing R. Jang, and Jui-Long Chen, “Wafer map failure pattern recognition and similarity ranking for large-scale data sets,” IEEE Transactions on Semiconductor Manufacturing, vol. 28, no. 1, pp. 1-12, 2014.

[6] Mengying Fan, Qin Wang and B. van der Waal, "Wafer defect patterns recognition based on OPTICS and multi-label classification," Advanced Information Management, Communicates, Electronic and Automation Control Conference, Xi'an, China, pp. 912-915, 2016.

[7] M. Piao, C. H. Jin, J. Y. Lee and J.Y. Byun, “Decision Tree Ensemble-Based Wafer Map Failure Pattern Recognition Based on Radon Transform-Based Features,” IEEE Transactions on Semiconductor Manufacturing, vol. 31, no. 2, pp. 250-257, 2018.

[8] Yu, Jianbo, and Xiaolei Lu, “Wafer map defect detection and recognition using joint local and nonlocal linear discriminant analysis,” IEEE Transactions on Semiconductor Manufacturing, vol. 29, no. 1, pp. 33-43, 2015.

[9] B. Kim, Y.S. Jeong, S. H. Tong, I.K. Chang and M.K. Jeongyoung, “A regularized singular value decomposition-based approach for failure pattern classification on fail bit map in a DRAM wafer,” IEEE Transactions on Semiconductor Manufacturing, vol. 28, no. 1, pp. 41-49, 2015.

[10] "wm811k wafer map," Kaggle, [Online]. Available: https://www.kaggle.com/ingyi/wm811k-wafer-map.

[11] Shouhong Chen, Yuxuan Zhang, Xingna Hou, Yuling Shang, and Ping Yang, “Wafer map failure pattern recognition based on deep convolutional neural network,” Expert Systems with Applications, vol. 209, pp. 118254, 2022.

[12] Takeshi Nakazawa and Deepak V. Kulkarni, “Anomaly Detection and Segmentation for Wafer Defect Patterns Using Deep Convolutional Encoder-Decoder Neural Network Architectures in Semiconductor Manufacturing,” IEEE Transactions on Semiconductor Manufacturing, vol. 32, no. 2, pp. 183-190, May 2019.

[13] Muhammad Saqlain, Casim Abbas, and Jong Yun Lee, “A Deep Convolutional Neural Network for Wafer Defect Identification on an Imbalanced Dataset in Semiconductor Manufacturing Processes,” IEEE Transactions on Semiconductor Manufacturing, vol. 33, no. 3, pp. 436-444, Aug. 2020.

[14] Krishevsky, Alex, Sunkyver, and Geoffrey E. Hinton, “Imagenet classification with deep convolutional neural networks,” Advances in Neural Information Processing Systems, Communications of the ACM, vol. 60, pp. 84-90, 2012.

[15] K. Simonyan, A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv 1409.1556, 2014.

[16] "wm-811k wafer map," Kaggle, [Online]. Available: https://www.kaggle.com/ashishpatel26/wm-811k-wafermap.