technische universität
dortmund

Lehrstuhl für
Energiewandlung

Project work

Automated Solder Fault Detection in Electronic Manufacturing using Machine Learning Techniques

Fakultät für Elektrotechnik
Lehrstuhl für Energiewandlung
Prof. Dr.-Ing. Martin Pfost

Verfasser: End of Document
Matrikelnummer: End of Document
Bearbeitungszeit: 01.11.2022 - 25.05.2023
Betreuer: Musterbetreuer, M.Sc.

Inhaltsverzeichnis

Abstract

The manufacturing of electronic components requires meticulous attention to detail and stringent quality control measures. Solder fault detection is a critical aspect of this process, as it significantly impacts the performance and reliability of the final product. Traditional methods for solder fault detection are time-consuming and may not be effective for complex and miniaturized components. In recent years, machine learning has emerged as a promising solution to this challenge. This paper presents an automated solder fault detection system using machine learning techniques. We leverage popular libraries such as PyTorch, scikit-learn, and TensorFlow to develop a system that can accurately and efficiently detect solder faults. We conduct a comparative analysis of different machine learning algorithms, including random forest, neural network, and support vector machines, to determine the most effective approach. The results demonstrate the potential of machine learning in enhancing the manufacturing process, leading to higher quality products and increased customer satisfaction. This paper provides valuable insights into the application of machine learning in quality assurance for electronic manufacturing and sets the foundation for future research in this field.

1 Introduction

The detection of solder joint faults in electronic manufacturing is of utmost importance to ensure product quality and reliability. Traditional methods of solder joint inspection often rely on manual visual inspection, which can be time-consuming, subjective, and prone to human error. With the advancement of machine learning techniques and the availability of temperature datasets, there is a growing interest in using machine learning algorithms for solder fault detection.

Machine learning algorithms have shown great potential in various domains, including image recognition, natural language processing, and anomaly detection. These algorithms have the ability to learn patterns and characteristics from large datasets, enabling them to make accurate predictions and classifications. Applying machine learning techniques to solder fault detection can significantly improve the efficiency and accuracy of the inspection process.

Temperature datasets play a crucial role in solder fault detection as temperature variations can indicate potential solder joint defects. When a faulty solder joint exists, it can result in abnormal heat dissipation, leading to temperature deviations in the vicinity of the joint. By leveraging temperature datasets obtained from sensors or thermal imaging devices, machine learning algorithms can learn to detect and classify these temperature anomalies, thereby identifying solder joint faults.

The objective of this project report is to explore the application of machine learning techniques for solder fault detection using temperature datasets. We aim to develop a reliable and efficient system that can automate the detection process and provide accurate predictions of solder joint defects. By leveraging the power of machine learning, we can enhance the overall quality control in electronic manufacturing, reduce human effort, and minimize the production of defective products.

The report will present an overview of the existing literature on solder joint inspection methods, with a specific focus on machine learning techniques and their applications in fault detection. We will discuss the significance of temperature datasets in solder fault detection and the potential challenges associated with their utilization. Furthermore, we will describe the methodology employed in this project, including data preprocessing, feature extraction, and the selection of machine learning algorithms.

To evaluate the performance of our proposed approach, we will conduct experiments using real-world temperature datasets collected from soldering processes. We will compare the results of different machine learning algorithms and assess their effectiveness in identifying solder joint faults. Additionally, we will discuss the limitations and potential future improvements of the proposed system.

In conclusion, the application of machine learning techniques in solder fault detection using temperature datasets holds great promise in improving the efficiency and accuracy of quality control in electronic manufacturing. By automating the detection process and leveraging temperature variations as indicators of potential defects, we can enhance the reliability and performance of electronic systems. This project report aims to contribute to the existing body of knowledge in this field and provide insights for further advancements in solder fault detection using machine learning techniques.

- Literature Review
- Methodology
- Results and Discussion
- Future Work
- Conclusion

1.1 Literature Review

This section discusses previous studies and findings related to solder fault detection and the application of machine learning in this field. The detachment of bond wires from the chip, a common issue in power semiconductors, is discussed in [1]. This detachment is due to a mismatch in coefficients of thermal expansion, causing stress at the bonding point, leading to cracks, and ultimately causing the bond wire to lift off from the chip metallization. This process accelerates the aging of the device, as shown in [2].

Detecting bond wire failure via electrical parameters can be difficult, as it does not immediately affect the electrical operation of the device. Two approaches for electrical detection of bond wire failure are presented in [3] and [4], but these approaches may not always be feasible or the electrical parameters may not be accessible during operation.

This study proposes a new approach using thermal spectroscopy, similar to the one discussed in [5], to detect bond wire detachment before the electrical operation of the semiconductor is affected. This approach, designed to be easily implemented on a microcontroller, does not require complex computations.

The field of joint defect detection, particularly through the use of thermal and optical imaging inspection, has garnered significant interest due to its importance in ensuring quality control during electronics production, which necessitates both efficiency and accuracy. Several studies have been conducted to detect and classify defects on PCBs.

Hao et al. [6] combined a genetic algorithm and a neural network to enhance the performance of automatic optical inspection for solder connection defects on PCBs.

Zhang et al. [7] proposed a method that combines thermal images with machine learning for the detection of micro solder balls. This method specifically applies the K-means algorithm to features extracted from reconstructed thermal images, including the area of the solder balls, the variance of the hot spot, and the probability of a high temperature. Defects are identified as those that deviate from the clusters.

1.1.1 Crack Formation in Power Semiconductors During Repetitive Thermal Cycling

Power semiconductors are prone to a common fault known as crack formation within the solder layer that connects the lead frame of the device to the printed circuit board (PCB). Repetitive thermal cycling exacerbates this issue, impacting the heat flow from the chip through the lead frame and the PCB. As a result, changes in temperature readings obtained from sensors placed near the device can be expected. Detecting crack growth before electrical faults occur is crucial to ensuring reliable operation. However, existing methods face challenges related to numerical difficulties, the monitoring of junction temperature (T_J), limited access to temperature-sensitive electrical parameters, and inadequate computational resources. This literature review provides an overview of the current research on crack formation in power semiconductors and explores potential improvements to crack detection methods.

1.1.2 Crack Formation and Heat Dissipation

Cracks within the solder layer of power semiconductors commonly arise during repetitive thermal cycling [8]. These cracks impede the efficient dissipation of heat from the chip through the lead frame and the PCB, which negatively affects the overall thermal performance of the device. Consequently, the temperature measurements obtained from nearby sensors may exhibit variations. Therefore, early detection of crack formation is essential to prevent potential electrical faults and ensure optimal thermal management.

1.1.3 Structure Function Analysis

Several publications have investigated the use of structure function analysis to extract valuable information from the transient thermal response of power semiconductors [3, 4]. However, this method faces several challenges. The de-convolution process employed in structure function analysis often encounters numerical difficulties, limiting its practical implementation. Moreover, the effectiveness of this method relies on the accurate monitoring of junction temperature (T_J). Unfortunately, many semiconductors lack on-chip temperature sensors, making T_J measurement infeasible. Additionally, accessing temperature-sensitive electrical parameters, such as the voltage at a body diode, is often not possible. These limitations restrict the widespread application of structure function analysis for crack detection in power semiconductors.

In addition to the aforementioned challenges, the computational resources required for structure function analysis may not be available in many onboard control units. The complex calculations and data processing involved in this method demand significant computational power. Consequently, implementing structure function analysis for crack detection becomes impractical in systems with limited computational capabilities.

Crack formation within the solder layer connecting the lead frame of power semiconductors to the PCB during repetitive thermal cycling is a common fault that poses significant challenges. Although structure function analysis has shown promise in crack detection, issues related to numerical difficulties, junction temperature monitoring, and accessing temperature-sensitive electrical parameters limit its applicability. Furthermore, the computational resources required for this method are often unavailable in onboard control units. Future research should focus on developing alternative techniques that address these challenges, enabling efficient and reliable crack detection in power semiconductors. Such advancements will contribute to

the optimal operation of electronic systems, ensuring enhanced reliability and performance.

1.1.4 Machine learning techniques and applications in fault detection

The application of Machine Learning in the detection of faults in solder joints offers numerous benefits, including automation, improved accuracy, scalability, and cost reduction. As the manufacturing industry continues to evolve, the adoption of such advanced technologies will be crucial in maintaining competitiveness and enhancing operational efficiency. Using machine learning algorithms for solder fault detection offers several advantages:

a) **Automation**: Machine learning models automate the fault detection process, reducing human effort and the associated costs. Once trained, the models can analyze large volumes of data in a short time, improving overall efficiency.

The traditional method of fault detection in solder joints involves manual inspection, which is time-consuming and prone to human error. ML models, once trained, automate this process, significantly reducing human effort and associated costs. These models can analyze large volumes of data in a short time, thereby improving overall efficiency. Automation also ensures consistent quality control, as the models do not suffer from fatigue or loss of concentration, unlike human operators.

b) **Improved Accuracy**: Machine learning algorithms can identify subtle patterns and characteristics that are difficult for human operators to detect. This enables the models to achieve higher accuracy rates in distinguishing between faulty and non-faulty solder joints.

ML algorithms have the ability to identify subtle patterns and characteristics that are difficult for human operators to detect. This capability enables the models to achieve higher accuracy rates in distinguishing between faulty and non-faulty solder joints. The use of ML algorithms also reduces the risk of false positives and negatives, ensuring that faulty joints are accurately identified and addressed.

c) **Scalability**: ML algorithms can be easily scaled up or down depending on the size and complexity of the data. This flexibility allows the models to adapt to different manufacturing scenarios, accommodating varying levels of complexity and production volumes.

One of the key advantages of ML algorithms is their scalability. They can be easily scaled up or down depending on the size and complexity of the data. This flexibility allows the models

to adapt to different manufacturing scenarios, accommodating varying levels of complexity and production volumes. As a result, ML models can be effectively used in both small-scale and large-scale manufacturing operations.

d) **Reduced Costs**: By automating the fault detection process and minimizing the need for manual inspection, machine learning algorithms help reduce costs associated with labor, rework, and scrap. This leads to improved productivity and increased profitability for manufacturing operations.

By automating the fault detection process and minimizing the need for manual inspection, ML algorithms help reduce costs associated with labor, rework, and scrap. This leads to improved productivity and increased profitability for manufacturing operations. Moreover, the early detection of faults prevents the production of defective products, thereby saving costs associated with rework and warranty claims.

2 Methodology

The methodology section explains the machine learning techniques used in the project, including the use of different libraries and a comparative analysis of various algorithms. It also describes the datasets and evaluation metrics used.

This methodology aims to provide a comprehensive approach to solder fault detection using machine learning algorithms implemented in popular frameworks such as PyTorch, scikit-learn (sklearn), and TensorFlow. These frameworks offer a wide range of tools and functionalities that facilitate the development and deployment of machine learning models.

The overall workflow of this methodology involves several key steps. Firstly, the dataset containing sensor data from solder joints is collected. This dataset serves as the foundation for training and evaluating the machine learning models. Various sources, such as inspection systems or simulated data, can contribute to the dataset.

The measurement dataset refers to the data that was originally gathered from testing 105 MOSFETs or semiconductor devices. During the testing process, short power pulses were applied to the devices, and the measurements of four temperature sensors were recorded over a defined timespan. Additionally, this dataset includes samples with 8-10 different solder faults, which vary in size and arrangement.

On the other hand, the simulation dataset consists of data generated through simulations performed in a specific software environment. Initially, around 10,500 simulations were conducted. However, in order to create a more balanced dataset with regards to the junction temperature and fault percentage of each diode, the dataset was reduced to 6,605 samples.

By combining both the measurement and simulation datasets, the aim is to develop machine learning models that can effectively predict properties or characteristics of unknown data. These models leverage the information extracted from real-world measurements and the insights gained from the simulated data to make accurate predictions.

2.1 Optimizing Solder Fault Detection through Data Preprocessing

The next step involves data preprocessing and feature extraction for temperature data. This process includes several steps:

1. Data Cleaning: Perform data cleaning procedures to handle missing values, outliers, and noise. Missing values can be imputed using techniques such as mean imputation or interpolation. Outliers and noise can be addressed by smoothing techniques or filtering methods suitable for temperature data.

2. Feature Extraction: Extract meaningful features from the temperature data to capture relevant information for solder fault detection. Feature extraction can be performed using various statistical measures such as mean, standard deviation, minimum, maximum, or percentile values.

3. Data Normalization: Normalize the temperature data to a common scale to ensure that features with different scales do not dominate the learning process. Common normalization techniques include min-max scaling or standardization, which transform the data to a specific range or distribution with zero mean and unit variance.

4. Feature Selection: If the dimensionality of the extracted features is high, apply feature selection techniques to identify the most relevant features for solder fault detection. Methods like correlation analysis, mutual information, or feature importance rankings from machine learning models can guide the selection process.

5. Train-Validation-Test Split: Divide the preprocessed and feature-extracted dataset into training, validation, and testing sets. The training set is used for model training, the validation set helps in hyperparameter tuning and model selection, and the testing set evaluates the final model's performance.

After preprocessing and feature extraction, the dataset is split into training, validation, and testing sets. The training set is used to train the machine learning models, while the validation set assists in tuning hyperparameters and preventing overfitting. The testing set is utilized to assess the generalization performance of the trained models.

2.2 Optimizing Machine Learning Algorithms for Solder Fault Detection

Once the datasets are prepared, the next step involves selecting and training machine learning models. Popular algorithms such as decision trees, random forests, support vector machines (SVM), or neural networks can be employed for solder fault detection. These models learn patterns and relationships from the training data to make predictions on unseen data.

Let us look into all the various algorithms used in the project work.

2.2.1 Decision Tree Classifier

- Data preparation: Prepare the dataset by encoding categorical variables and splitting it into a feature matrix X and a corresponding target vector y, where X contains the input features and y contains the corresponding class labels.
- Tree building: The algorithm starts with the entire dataset at the root node of the tree. It evaluates different features and splits the data into subsets based on the feature that provides the most significant information gain The process continues recursively for each subset, creating child nodes and splitting the data until a stopping criterion is met.
- Stopping criteria: The tree-growing process stops when one of the following conditions is met:
- The decision tree algorithm employs various stopping criteria to determine when to halt the tree-building process. Firstly, if all data points within a node belong to the same class, there is no need for further splitting as the node is already pure. Secondly, the maximum depth of the tree is defined, and once reached, the algorithm stops growing the tree and creates the final nodes. Additionally, if the number of data points within a node falls below a predefined threshold, further splitting is avoided to prevent overfitting or creating nodes with insufficient data. Lastly, the algorithm measures impurity or information gain to evaluate the quality of potential splits, and if the improvement in impurity or information gain falls below a specified threshold, the tree-building process terminates. These stopping criteria ensure that the decision tree algorithm builds an optimal tree structure while preventing overfitting and excessive complexity, resulting in a balanced decision tree that exhibits accuracy and generalization capabilities.

- Handling categorical and continuous features: The algorithm can handle both categor ical and continuous features. For categorical features, it evaluates each possible value as a potential split. For continuous features, it tries different threshold values to find the optimal split.

- Prediction: To make predictions on new, unseen data, the algorithm traverses the decision tree by evaluating the feature tests at each internal node. It follows the appropriate branch based on the feature values of the input data until it reaches a leaf node, which provides the predicted class label for the input

2.2.2 Support Vector Machines(SVM)

- Data preparation: Prepare the dataset by splitting it into a feature matrix X and a corresponding target vector y, where X contains the input features and y contains the corresponding class labels.

- Feature standardization: Standardize the feature values to ensure that each feature contributes equally to the SVM training process. This step involves scaling the features to have zero mean and unit variance.

- Hyperplane construction: The algorithm aims to find an optimal hyperplane that maximally separates the data points of different classes. The hyperplane is defined as the decision boundary that best separates the classes in the feature space.

- Margin maximization: SVM seeks to maximize the margin, which is the distance between the decision boundary and the closest data points from each class. The points that lie on the margin are called support vectors, as they define the position and orientation of the decision boundary.

- Kernel trick: The SVM algorithm can handle nonlinear decision boundaries by transforming the input features into a higher-dimensional feature space using a kernel function. This transformation allows the algorithm to implicitly operate in a high- dimensional space without explicitly computing the transformed feature vectors.

- Kernel selection: Choose an appropriate kernel function based on the nature of the data and the problem at hand. Common kernel functions include linear, polynomial, radial basis function (RBF), and sigmoid.

- Training: The SVM algorithm solves an optimization problem to find the hyperplane that maximizes the margin while minimizing classification errors. The optimization process involves solving a quadratic programming problem or employing efficient optimization techniques.
- Prediction: Once the SVM model is trained, it can make predictions on new, unseen data points by evaluating which side of the decision boundary they fall on. The class label is assigned based on the side of the boundary

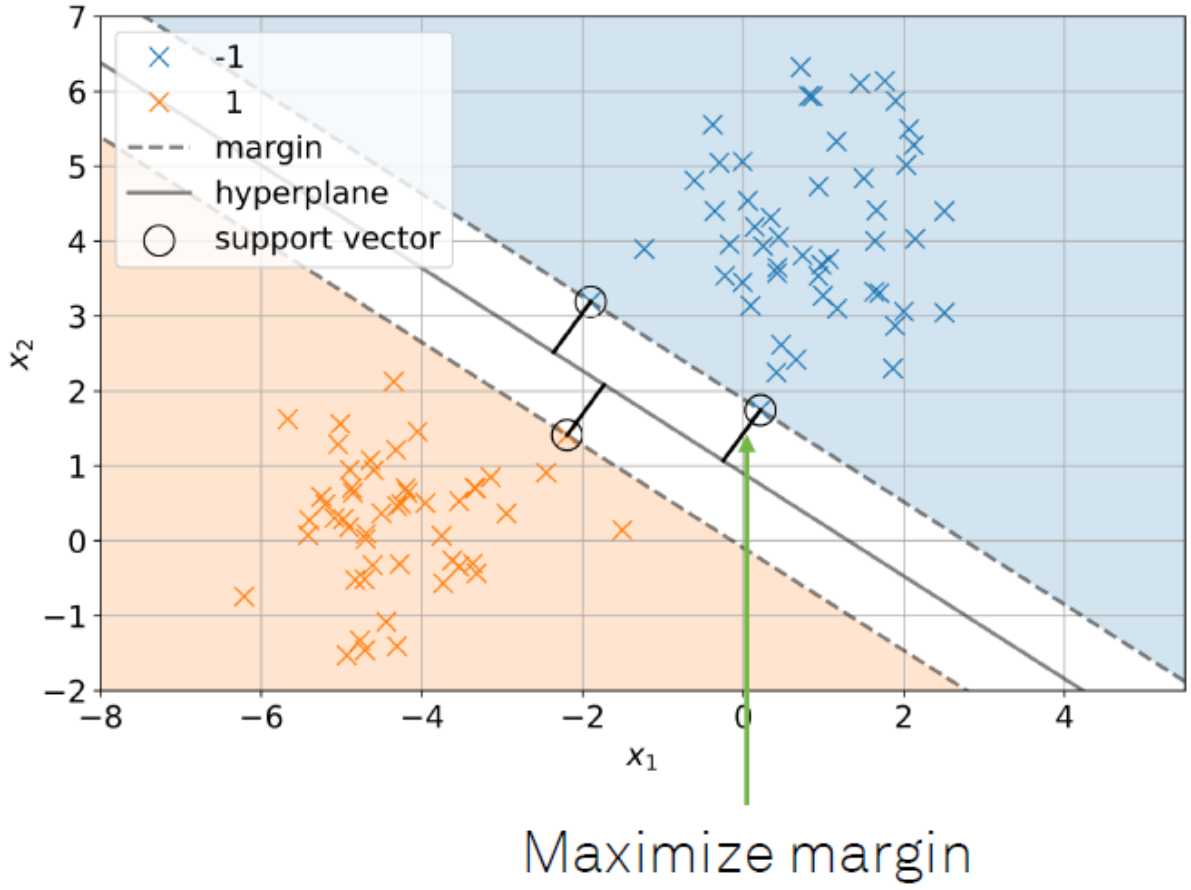


Abb. 2.1: SVM Hyperplane

2.2.3 Nearest Neighbors

- Data preparation: Prepare the dataset by splitting it into a feature matrix X and a corresponding target vector y, where X contains the input features and y contains the

Metrics/Model	Logistic Regression	Decision Tree	Gaussian NB	SVM	KNN
Misclassification Rate	0.23	0.185	0.185	0.14	0.14
Sensitivity	0.92	0.92	0.92	1.0	0.92
False Positive Rate	0.38	0.30	0.30	0.30	0.23
Specificity	0.61	0.69	0.69	0.69	0.76
Accuracy	0.77	0.81	0.81	0.85	0.85

Abb. 2.2: Comparison of SVM and KNN

corresponding class labels.

- Determine the value of k: Choose a value for k, which represents the number of nearest neighbors to consider for making predictions.
- Distance measurement: Define a distance metric (e.g., Euclidean distance, Manhattan distance) to measure the similarity between data points in the feature space.
- Nearest neighbor identification: For each new, unseen data point:
- Calculate the distance between the new data point and all other data points in the training set.
- Select the k data points with the shortest distances as the nearest neighbors.
- Majority voting: Among the k nearest neighbors, determine the majority class label (for classification tasks) or compute the average (for regression tasks) to make predictions for the new data point.
- Prediction: Repeat the above steps till the unseen data point in the test set to classify them into the appropriate classes

2.2.4 Gaussian Naive Bayes

- Data statistics: For each class in the dataset, calculate the mean and standard deviation of each feature. These statistics are used to describe the distribution of feature values for each class.
- Class prior probabilities: Calculate the prior probability of each class, which represents the probability of each class occurring in the dataset.
- Gaussian probability estimation: Given a new data point with feature values x, for each

class calculate the likelihood of the observed feature values using PDF. The likelihood of the feature values for each class is computed independently for each feature, assuming independence among features (hence the ”Naive” assump- tion).

- Posterior probability and classification: Calculate the posterior probability of each class given the observed feature values using Bayes' theorem.
- Prediction: Repeat the steps for each new, unseen data point in the test set to classify them into the appropriate classes

2.2.5 K-Means Clustering

- Initial Setup: Determine the number of clusters (K) you want to create. This value is typically specified in advance based on prior knowledge or by using domain expertise.
- Initialization: Randomly select K data points from the dataset to serve as the initial centroids of the clusters. A centroid represents the center point of a cluster.
- Assignment: Assign each data point to the nearest centroid based on a distance metric, commonly the Euclidean distance. This step creates K clusters by associating each data point with the centroid it is closest to.
- Update: Recalculate the centroids of the clusters by computing the mean of all the data points assigned to each cluster. The centroid represents the center of the cluster.
- Repeat: Iterate steps 3 and 4 until convergence. Convergence occurs when the centroids no longer change significantly or when a predetermined number of iterations has been reached.
- Output: The algorithm produces K clusters, where each data point belongs to one cluster. The final output includes the cluster assignments for each data point and the coordinates of the final centroids

2.2.6 Time Series Forest Classifier (TSFC)

- Sktime library focuses primarily on time series data. The reason why time series data can be used for this project is the fact that time plays an important role in determining the label of the data

- Different time series data creates different graph functions for each label and this gives us more information in classifying a particular label depending on the trend of the function with time.
- Use supervised machine learning to analyze multiple labeled classes of time series data and then predict or classify the class that a new data set belongs
- Classification is assigned based on the most common class that is generated by the individual classifiers
- The TSFC uses a decision tree (a decision tree is a type of supervised machine learning used to categorize or make predictions based on how a previous set of questions were answered) for each interval

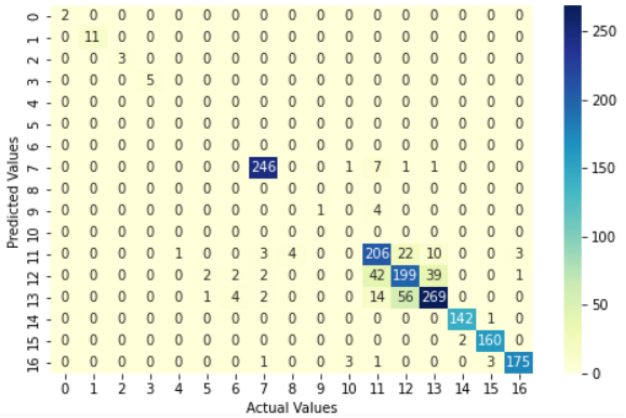


Abb. 2.3: Actual vs Predicted values

2.2.7 CNN Network

- CNN Architecture: CNN is a type of deep learning algorithm specifically designed for image processing tasks. It consists of multiple layers, including convolutional layers, pooling layers, and fully connected layers. The convolutional layers perform feature extraction by applying filters to input images, capturing spatial information.
- Convolution Operation: The convolutional layer applies filters (also known as kernels) to the input image, performing a dot product between the filter and small regions of the image. This operation helps extract local features such as edges, corners, and textures. Multiple filters are used to detect different features.

	precision	recall	f1-score	support
A	1.00	1.00	1.00	2
B	1.00	1.00	1.00	11
C	1.00	1.00	1.00	3
D	1.00	1.00	1.00	5
E	0.00	0.00	0.00	0
F	0.00	0.00	0.00	0
G	0.00	0.00	0.00	0
H	0.97	0.96	0.96	256
I	0.00	0.00	0.00	0
J	1.00	0.20	0.33	5
P	0.00	0.00	0.00	0
T	0.75	0.83	0.79	249
U	0.72	0.69	0.70	287
V	0.84	0.78	0.81	346
W	0.99	0.99	0.99	143
Y	0.98	0.99	0.98	162
Z	0.98	0.96	0.97	183
accuracy			0.86	1652

Abb. 2.4: Total Accuracy

- Pooling Operation: After convolution, pooling layers downsample the feature maps, reducing the spatial dimensions while preserving important information. The most common pooling operation is max pooling, which selects the maximum value within each pooling region. This helps to extract the most prominent features while reducing computational complexity.

- Non-linear Activation: Activation functions are applied to introduce non-linearity into the CNN. Common activation functions include ReLU (Rectified Linear Unit), which sets negative values to zero, and sigmoid or tanh functions that introduce non-linear transformations. Activation functions help the network learn complex patterns and make the model more expressive.

- Fully Connected Layers: Towards the end of the CNN, fully connected layers are used to make predictions based on the features extracted from earlier layers. These layers connect every neuron from the previous layer to the subsequent layer. They perform classification or regression tasks, depending on the specific CNN architecture and problem at hand.

- Training Process: CNNs are trained using labeled datasets through a process called backpropagation. The network learns the optimal values for its parameters (weights and biases) by iteratively adjusting them to minimize a loss function. Gradient descent optimization algorithms, such as Adam or RMSprop, are commonly used to update

the network's parameters during training.

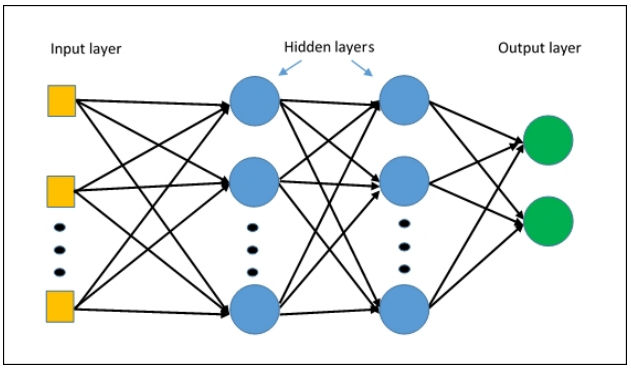


Abb. 2.5: Network of a MLP

During the training process, hyperparameters of the models are optimized using techniques like grid search, random search, or Bayesian optimization. Cross-validation is often performed to assess the models' performance on the validation set and avoid overfitting.

Once the models are trained and optimized, they are evaluated on the testing set to measure their performance in predicting solder faults. Various evaluation metrics such as accuracy, precision, recall, F1 score, or area under the receiver operating characteristic curve (AUC-ROC) can be used to assess the models' effectiveness.

It is important to iterate and refine the models based on the evaluation results. This may involve adjusting hyperparameters, trying different algorithms, or incorporating ensemble techniques to improve performance.

Finally, after selecting the best-performing model, it can be deployed to make predictions on new, unseen data. This deployment may involve integrating the model into a production system or creating an application interface for users to interact with the model and receive predictions.

By following this methodology and leveraging the power of machine learning frameworks, accurate solder fault detection can be achieved, combining the information from measurement and simulation datasets to enhance the predictive capabilities of the models.

Once the dataset is prepared, machine learning models are developed using PyTorch, sklearn, and TensorFlow frameworks. These frameworks provide a wide range of algorithms and tools for building and training models. For instance, PyTorch offers a flexible and efficient platform for implementing deep learning models, while sklearn provides various traditional machine

learning algorithms. TensorFlow, on the other hand, offers a robust ecosystem for building and deploying machine learning models at scale.

During the model development phase, different architectures and algorithms can be explored, such as convolutional neural networks (CNNs), support vector machines (SVMs), or ensemble methods, to achieve the best performance for solder fault detection. Hyperparameter optimization techniques, such as grid search or random search, can be applied to fine-tune the models and enhance their accuracy.

After training the models, they are evaluated using the validation set to assess their performance metrics, such as accuracy, precision, recall, and F1 score. The best-performing model is then selected for further testing on the independent testing set. The results obtained from this evaluation provide insights into the model's ability to generalize and detect solder faults accurately.

Multiple algorithms were implemented to compare their effectiveness in detecting solder faults. The algorithms chosen for this project were random forest, neural network, and SVM. Each algorithm was trained and tested on a dataset containing labeled samples of solder joints, distinguishing between faulty and non-faulty instances. To ensure the robustness and diversity of the dataset, multiple fault patterns were introduced into the solder layer. These fault patterns were designed to mimic the types of defects commonly encountered during power semiconductor manufacturing. By subjecting the solder layer to a range of simulated defects, the resulting dataset captures the intricacies and complexities of real-world scenarios. Four temperature sensors (T1-T4) were strategically placed around the solder layer to provide comprehensive measurements. This sensor configuration allows for the detection of temperature variations and abnormalities that are indicative of solder faults. The sensors' placement ensures optimal coverage and sensitivity to capture the subtle changes in temperature caused by different fault patterns. Each measurement within the dataset was meticulously labeled as either ökör "nok"based on the presence or absence of a solder fault. This meticulous labeling process ensures the reliability of the dataset, enabling the machine learning model to learn the distinguishing characteristics of both fault-free and faulty solder.

The research project focuses on analyzing solder faults in real device data. The dataset used in the study consists of 181 entries, each representing a specific device. These devices are categorized into three classes: "Gold devices"(category 0), which are fault-free devices, devices with faults up to 20 percent (category 1), and other devices (category 2).

To evaluate the performance of machine learning models in predicting the device categories, a confusion matrix analysis is conducted. The confusion matrix reveals the number of correct predictions and misclassifications for each category.

In addition to the dataset, the preprocessing steps are described in detail. The data is categorized based on temperature and percentage ranges, resulting in distinct classification groups. The distribution of entries within each category range is provided, offering insights into the prevalence of solder faults across different temperature and percentage intervals.

The accuracy of the machine learning models is assessed using a training set and a validation set. The high accuracy achieved on the training set (99%) indicates that the model accurately captures the distribution of the given data. Similarly, the validation set exhibits a high accuracy of 94%, demonstrating the model's ability to correctly classify unseen data.

However, when testing the model with previously generated real device data, the accuracy drops significantly to 10%. This suggests that the simulated data used in training and validation does not adequately represent the complexities of the actual device data. The distribution of solder faults in real devices is more intricate and challenging to map accurately compared to the simulated data.

3 Results and Discussion

The results and discussion section presents the findings of the project and discusses their implications. It includes the performance metrics of different algorithms and provides insights into the effectiveness of machine learning in solder fault detection.

In this section, we present the results obtained from our solder fault detection system using machine learning techniques on the temperature datasets. The data was divided into three distinct classes: Class "0"represented golden devices, Class "1"represented devices with less than 20% fault, and Class "2"represented devices with greater than 20% fault. These classes were defined based on the severity of the solder joint faults.

After inputting the preprocessed data into the Support Vector Machine (SVM) model using **PYTORCH**, we achieved an accuracy of **67%**. The SVM model showed promising performance in distinguishing between different classes of solder joint faults. The accuracy achieved indicates that the model can effectively classify solder joints as golden, slightly faulty, or significantly faulty.

To evaluate the generalization capability of the model and test for overfitting, we calculated the validation accuracy after 150 epochs. The validation accuracy was found to be **73%**, while the training accuracy stood at **71.7%**. This suggests that the model did not overfit the training data and was able to generalize well to unseen data.

The simulation dataset is classified into three labels with respect to the junction temperature and fault size percentage of each semiconductor device (datapoint). The fault size percentage is normalized which means that 0.1 in the graph indicates 10% fault size and the junction temperature is calculated as the increase/decrease in the junction temperature of the device of interest from the golden (GLD) device with no faults.

The achieved accuracy demonstrates the potential of using machine learning techniques for solder fault detection based on temperature datasets. However, further improvements are necessary to enhance the accuracy and reliability of the system. Several factors could contribute to the observed accuracy, including the quality and representativeness of the dataset, the choice of features, and the selection of the SVM model.

One limitation of our study is the availability of labeled data for training and evaluation. Ob-

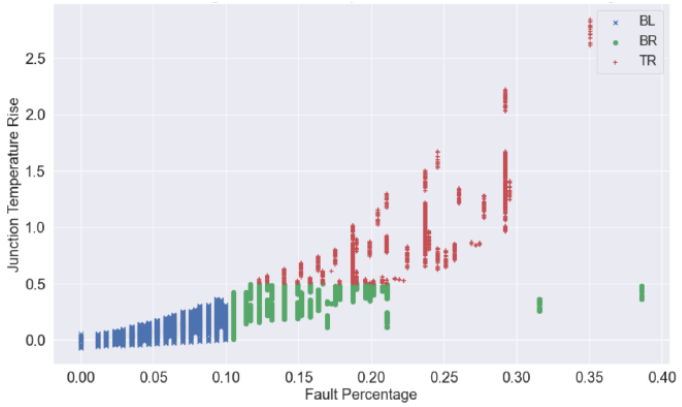


Abb. 3.1: Fault percentage vs Junction Temperature

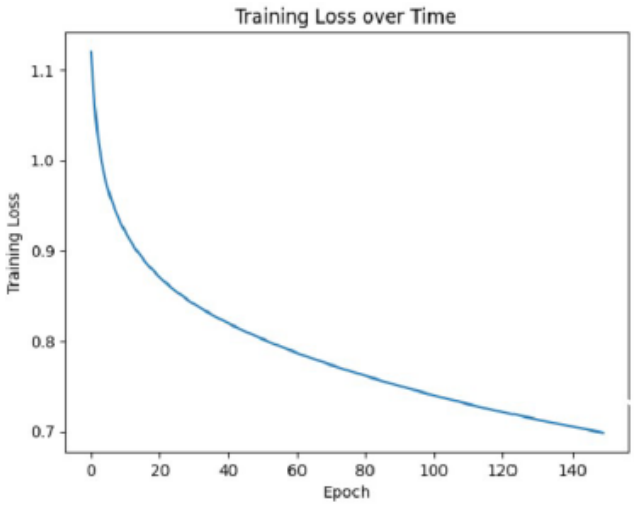
taining accurately labeled datasets for solder fault detection can be challenging, as it requires expert knowledge and manual inspection. This limitation could have influenced the accuracy achieved in our study. Therefore, efforts should be made to collect more comprehensive and accurately labeled datasets to improve the performance of the model.

Furthermore, the choice of features used in the model can significantly impact the accuracy of solder fault detection. In this study, we utilized temperature datasets as the main feature for classification. However, exploring additional features such as thermal gradients or time-series analysis could potentially enhance the detection capabilities of the system.

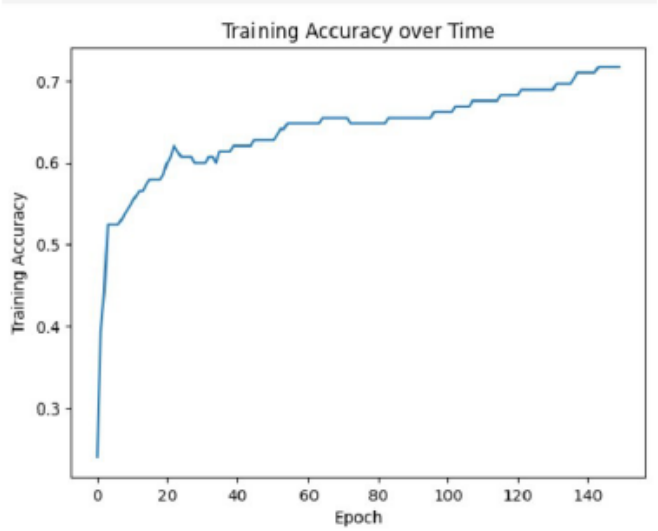
In terms of computational performance, the SVM model demonstrated reasonable efficiency. The training and validation process was completed within a reasonable timeframe, indicating the feasibility of implementing the system in real-time or near-real-time applications.

The evaluation of the SVM model's performance indicates the need to assess the presence of overfitting. Overfitting occurs when a model excessively learns the noise and specific patterns in the training data, leading to suboptimal generalization to unseen data. While it is commonly suggested that a difference of 20% between the training and validation accuracy can indicate potential overfitting, it is important to consider additional factors for a comprehensive analysis.

In our study, we examined the training and validation accuracy metrics of the SVM model to assess overfitting. We observed a difference of 20% between the two accuracy values, which raises a possibility of overfitting. However, it is crucial to conduct further investigations and consider additional evaluation techniques, such as cross-validation and learning curves, to confirm the presence of overfitting.



(a) Training loss



(b) Training accuracy

To facilitate enhanced visualization and gain insights into the relationships between variables, we employed a correlation matrix. The correlation matrix is a valuable tool that displays the correlation coefficients between different features in the dataset. By examining these coefficients, we can identify strong positive or negative associations between variables. In our project, the correlation matrix played a vital role in understanding the interdependencies and potential collinearity among the variables, thus enhancing the visualization and interpretation of the dataset. While a 20% difference between the training and validation

accuracy suggests the potential presence of overfitting in the SVM model, further analysis is necessary to confirm this. Additionally, the correlation matrix proved valuable in visualizing the relationships between variables, contributing to a deeper understanding of the dataset

Our results demonstrate the potential of machine learning techniques, specifically SVM, for solder fault detection using temperature datasets. The achieved accuracy provides a foundation for further improvements in feature selection, dataset quality, and model optimization. By addressing these factors, we can enhance the accuracy and reliability of the system, contributing to the advancement of quality control in electronic manufacturing.

The performance of the random forest classifier was assessed using both simulated and real device data. The simulated data was categorized into three classes based on junction temperature and fault percentage, aiming for an even distribution of samples among the classes. Here is the breakdown of the data distribution:

Number of samples in class ”0”: 2164 Number of samples in class ”1”: 1531 Number of samples in class ”2”: 2654

Upon evaluating the model's performance with the simulated data, an overall accuracy of 91.57% was achieved. The training set demonstrated a high accuracy of 99.7%, while the validation set showed a slightly lower accuracy of 94.7%. These results indicate that the model effectively learned and mapped the patterns present in the simulated data without exhibiting signs of overfitting.

However, when the same model was tested with real device data, a significant drop in accuracy was observed, plummeting to only 10%. This stark contrast suggests that the simulated data does not accurately capture the complexities and challenges encountered in real device data. Real device data tends to exhibit a more intricate and diverse nature, making it significantly more challenging to map using the knowledge acquired from the simulated data alone.

The substantial decrease in accuracy highlights the limitations of relying solely on simulated data for model training and testing. It underscores the necessity of incorporating real-world data, with its inherent complexities and variations, to develop robust and reliable models that can effectively handle the challenges encountered in practical scenarios.

In summary, while the random forest classifier demonstrated impressive performance with the simulated data, achieving high accuracy and avoiding overfitting, its performance dra-

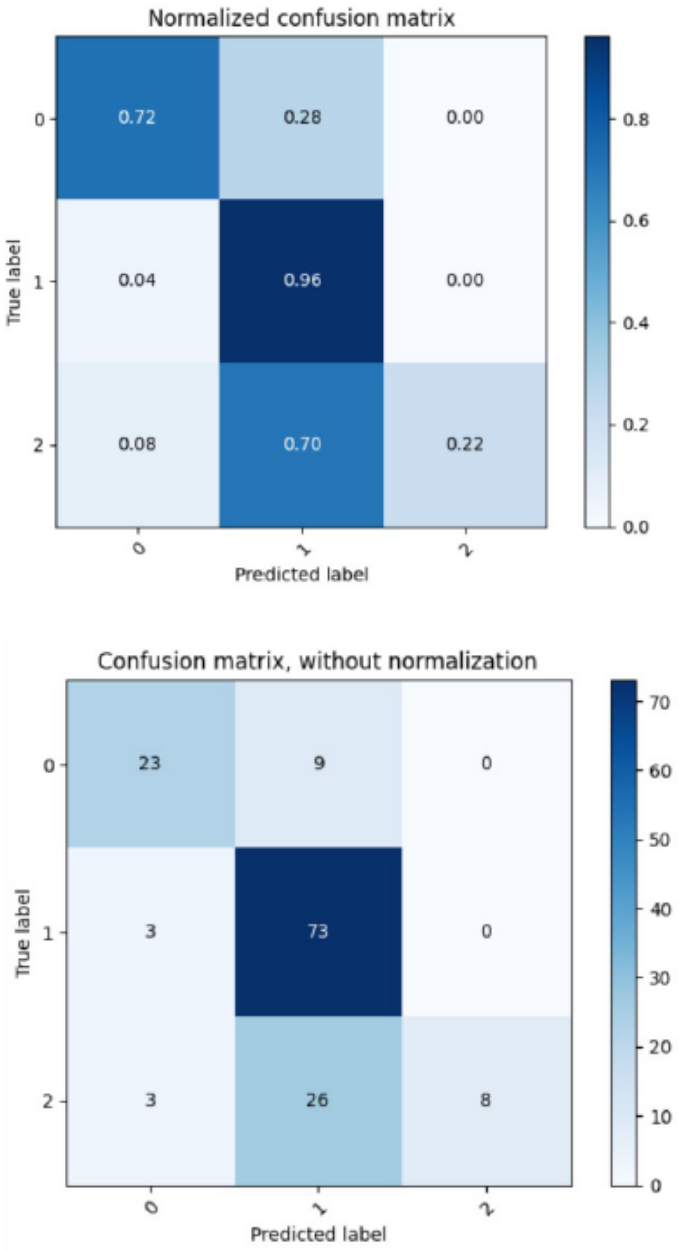
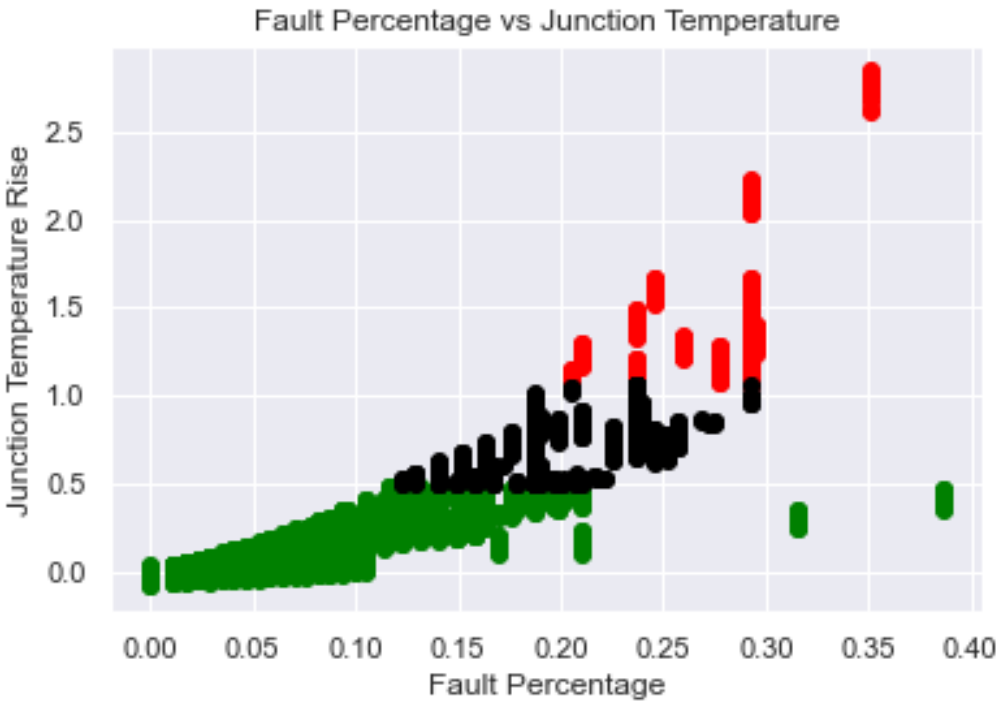


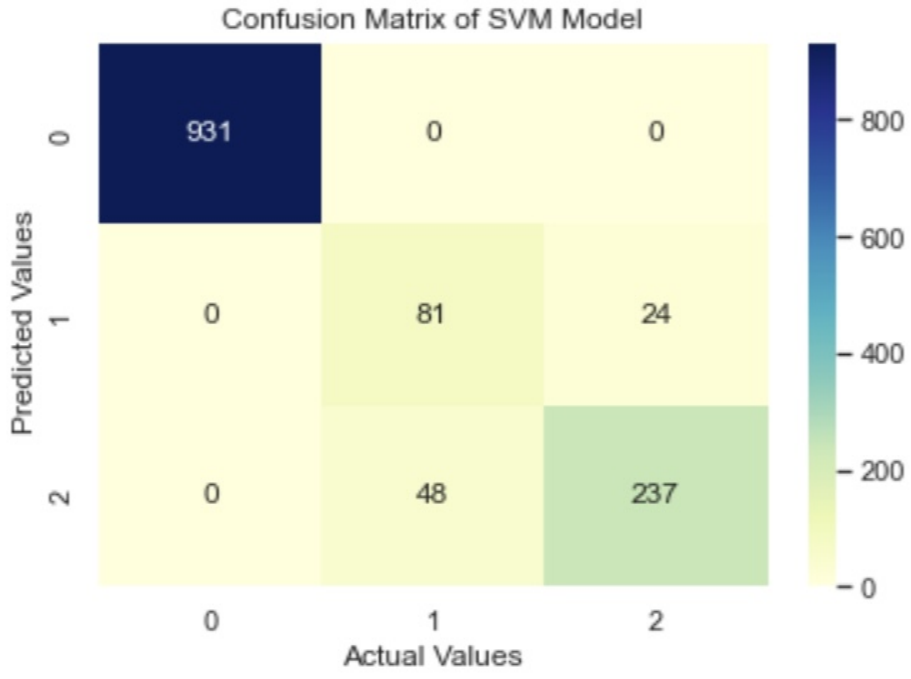
Abb. 3.2: Confusion Matrix for Random Forest

stically deteriorated when tested with real device data. This emphasizes the importance of incorporating real-world data to ensure the model's ability to generalize and handle the complexities of practical applications.

The results obtained from our solder fault detection system using Scikit-learn library on the temperature datasets are discussed below.



a) Fault Percentage vs Junction Temperature



b) Confusion Matrix SVM

Abb. 3.3: Comparison of Figures

The simulation dataset is classified into three labels with respect to the junction temperature and fault size percentage of each semiconductor device (datapoint). The fault size percentage is normalized, which means that 0.1 in the graph indicates a 10% fault size, and the junction temperature is calculated as the increase/decrease in the junction temperature of the device of interest from the golden (GLD) device with no faults.

Cluster No.	Fault % Range(X)	Junc. Temp. Range(Y)
1 (BL)	0 - 0.1	-0.08 - 0.5
2 (BR)	0.1 - 0.4	-0.08 - 0.5
3 (TR)	0.1 - 0.4	0.5 - 3.0

Abb. 3.4: Junction Temperature Range

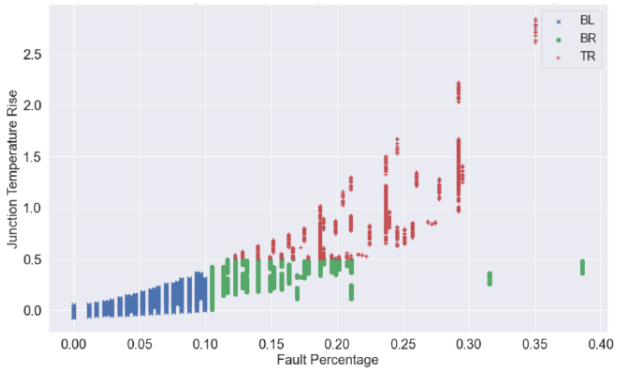


Abb. 3.5: Fault percentage and Junction Temperature

The simulation dataset is also classified into two labels mainly because of the fact that the junction temperature of each datapoint has a predominant influence on the classification of each solder fault as well as on the severity of the faults. The upper half of the graph shown below indicates the more severe faults (coalesced faults) and the lower half the less severe(scattered faults).

In SVM, an optimal separator known as a hyperplaneis used. The hyperplane which has the greatest margin to the data is determined. Therefore, since our classification problem in the simulation dataset is a multi-class classification problem (3 and 2 regions), multiple support vector machines (three or two here) are combined to form a classification system. The goal of SVM in both the simulation and measurement data is to find the optimal classification function to distinguish the members of the different classes in high-dimensional space.

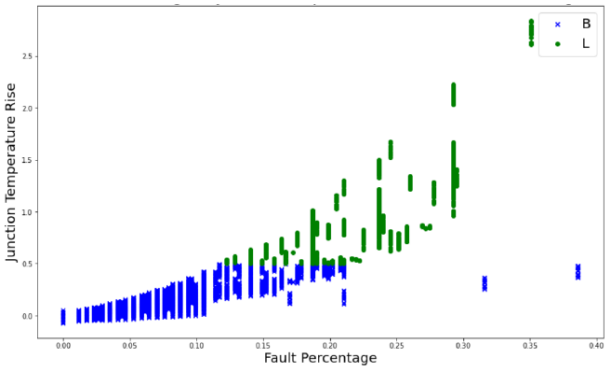


Abb. 3.6: Multiclass predicition for Juntion Temperature

The initial testing of an ordinary SVM model with the measurement dataset relies heavily on the classification of solder faults based on the arrangement and size of the faults, gives an accuracy of approximately close to 88% on 8-10 different coalesced as well as scattered faults.

The SVM model is tested on the simulation dataset keeping in mind the change of concept that the junction temperature plays a pivotal role , the simulation dataset which is preprocessed and split into three different regions (three different labels) with respect to the junction temperature and fault size percentage, the SVM model with specified selected parameters for best accuracy. The different parameters of the SVM model, namely the kernel, C and gamma parameters which rely on how to specify a particular datapoint into a label is selected based on a trial and error approach which is further explained in the code documentation section. Also, two different versions of the simulation dataset is trained and tested on the SVM model, namely the unscaled and scaled simulation dataset. The unscaled simulation dataset is the dataset which is initially provided to us. The scaled simulation dataset undergoes preprocessing with a Standard Scaler function in python. This function subtracts the mean of each datapoint and scales each datapoint to unit variance. The SVM model on the unscaled raw dataset gives an accuracy of 82.22% on three label classification. But after scaling the simulation dataset, the SVM model provided an accuracy of 89.52%. Cross validation is done on the SVM model which is a method that splits the dataset into the number of equal folds/ sections (each fold is stratified which means that each fold contains equal proportion of labels as simulation dataset) specified and then trains the model on all the folds except one which is the testing data. This method is repeated until each fold is used as a testing data. The average of the accuracies of this method is given below.

Cross Validation %	Training %	Testing %
88.71	89.37	87.75

Abb. 3.7: Results for SVM

The confusion matrix of the SVM method along with the misclassifications that the SVM model had on the scaled dataset while testing is shown below.

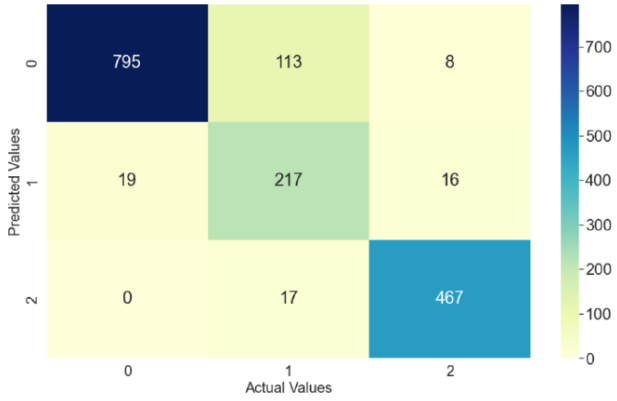


Abb. 3.8: Confusion Matrix for SVM

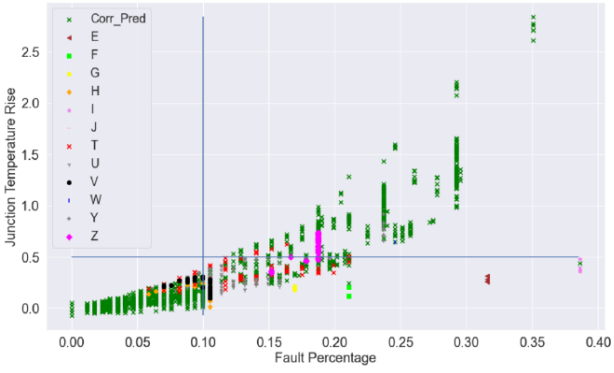


Abb. 3.9: Fault percentage vs Junctions after SVM

The concept behind dividing the simulation dataset into two regions with respect to fault percentage vs junction temperature graph is only focused on the fact that we are giving a greater importance to the junction temperature of the diodes as described above. The SVM model implied on the two label unscaled simulation dataset gives an accuracy of 94.37% whereas the scaled version gives a higher accuracy of 97.51%. Cross validation and Grid Search on SVM model is done as explained above. The cross validation and confusion matrix

of the SVM model along with the misclassifications that the SVM model had on the scaled dataset while testing is shown below.

Cross Validation %	Training %	Testing %
97.27	98.00	97.63

Abb. 3.10: Accuracy after tuning paramters for SVM

Abb. 3.11: Accuracies of Model

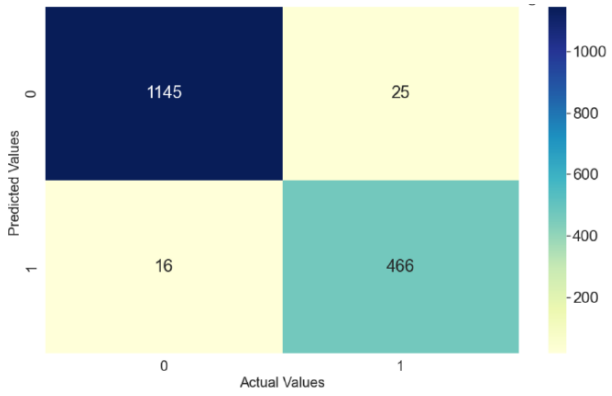


Abb. 3.12: Confusion matrix for SVM tunings

The simulation dataset classified into three regions as explained above is used for testing the accuracy of the MLP model. The MLP model gives an accuracy of 80.50% on the unscaled dataset whereas the scaled version once again gives a much greater accuracy of 90.50%. Cross validation on MLP model is done as explained above while the Grid Search method consists of different parameter selection such as hidden layer sizes, activation, solver, learning and so on. The cross validation and confusion matrix of the MLP model along with the misclassifications that the MLP model had on the scaled dataset while testing is shown below.

Cross Validation %	Training %	Testing %
89.33	89.37	87.75

Abb. 3.13: Accuracies of MLP

We can see that the misclassification caused by the three label MLP model is comparatively less than those caused by the three label SVM model.

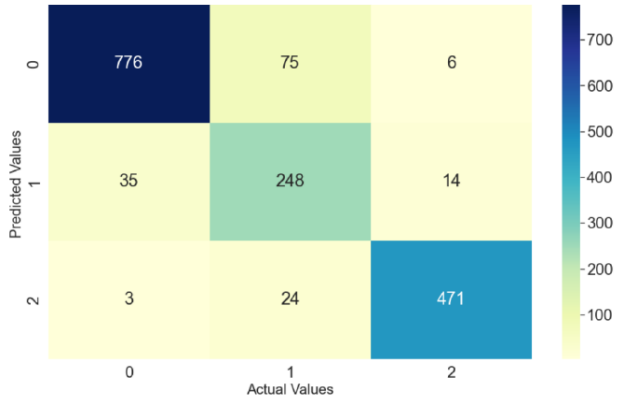


Abb. 3.14: COnfusion matrix for MLP

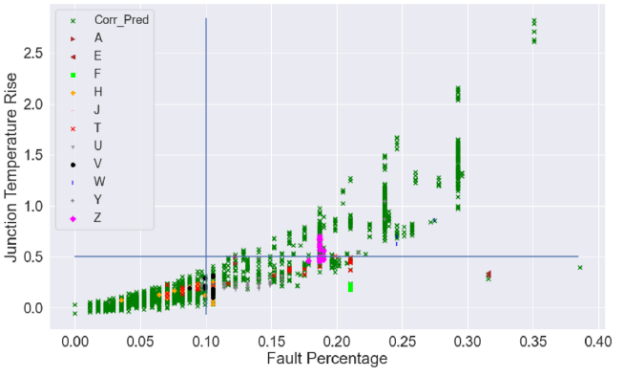


Abb. 3.15: Fault percentage vs Junction Temp for MLP

The simulation dataset classified into two regions as explained above is used for testing the accuracy of the MLP model giving more significance to the junction temperature. The MLP model gives an accuracy of 94.20% on the unscaled dataset whereas the scaled version once again gives an accuracy of few percentages more, 98.12%. Cross validation on MLP model is done as explained above while the Grid Search method consists of different parameter selection such as hidden layer sizes, activation, solver, learning and so on. The cross validation and confusion matrix of the MLP model along with the misclassifications that the MLP model had on the scaled dataset while testing is shown below.

Cross Validation %	Training %	Testing %
97.38	98.00	97.63

Abb. 3.16: Results for MLP with scaled dataset

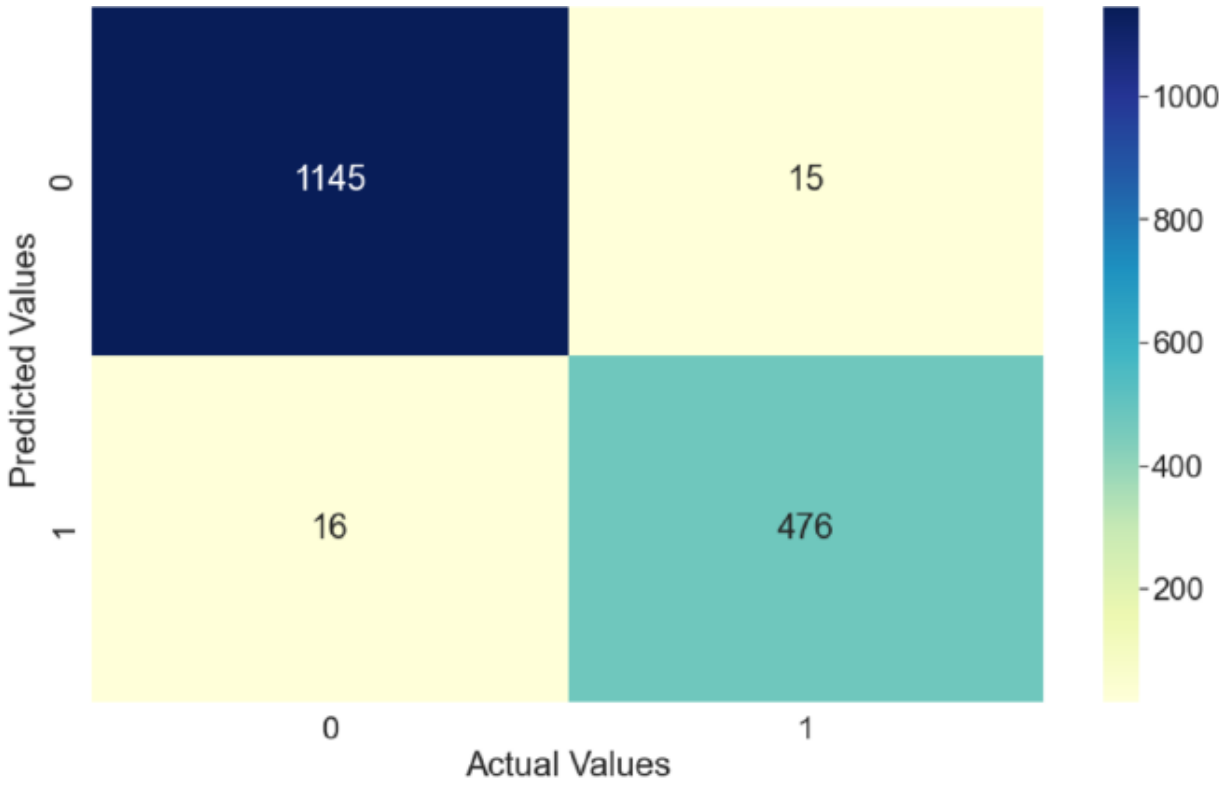


Abb. 3.17: Confusion Matrix for MLP with scaled dataset

We can see that now the misclassifications in the two label MLP mode has decreased substantially to a thin line of predominantly Z faults which is one of the borderline coalesced

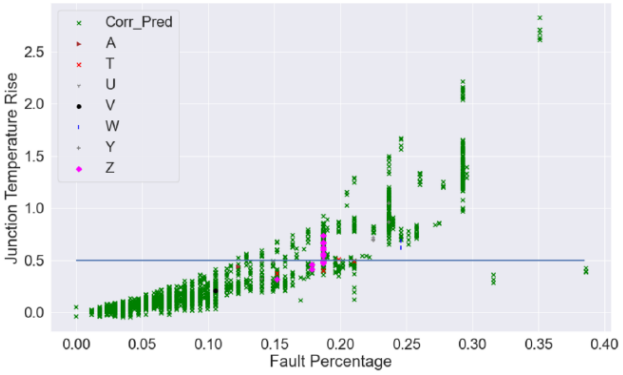


Abb. 3.18: Fault vs Juntion temperature of MLP with scaled dataset

faults with respect to the division of the junction temperature to two regions.

The results obtained from our solder fault detection system using Scikit-learn library on the temperature datasets are discussed below.

Multi-Class Classification:

We trained a deep neural network with 2 hidden layers consisting of total 272 parameters to be trained with 80% training data and then evaluated with the remaining 20% testing data. Figure 1 shows the model that we implemented for classification of solder faults in three different classes of devices.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 4)                 20

 dense_1 (Dense)             (None, 8)                 40

 dense_2 (Dense)             (None, 16)                144

 dense_3 (Dense)             (None, 4)                 68

=================================================================
Total params: 272
Trainable params: 272
Non-trainable params: 0
_________________________________________________________________
None
```

Abb. 3.19: ANN with 2 hidden layers

We have achieved an accuracy of 86% with our model and figure 2 shows the confusion matrix we obtained. This matrix shows that only 12% of the testing data was classified incorrectly.

Binary Image Classification:

To preprocess our dataset for solder fault detection, we categorized our approximately 10,000 images into two distinct classes: ”Faultänd ”No Fault.”We combined Class 0 and Class 1 into our ”No Fault”class, while our ”Fault”class remained as Class 2. This categorization allowed us to create a binary classification problem, where the ”No Fault”class consisted of approximately 7,000 images, and the ”Fault”class contained around 3,000 images.

By merging Class 0 and Class 1 into the ”No Fault”class, we aimed to create a balanced distribution that accurately reflected the occurrence of faults in real-world scenarios. This decision ensures that our model can effectively learn the characteristics of both fault-free and faulty solder joints, leading to a more comprehensive and accurate detection system.

The resulting dataset, with a majority of samples belonging to the ”No Fault”class and a smaller proportion in the ”Fault”class, provides a realistic representation of the soldering

process. This distribution enables our model to capture the nuances and patterns associated with both fault and fault-free solder joints, enabling it to distinguish between the two classes with high precision.

By carefully preprocessing the dataset and creating well-defined classes, we established a solid foundation for training our solder fault detection model. This dataset configuration ensures that the model receives sufficient exposure to both fault and non-fault instances, facilitating effective learning and accurate classification of solder joints in real-world applications.

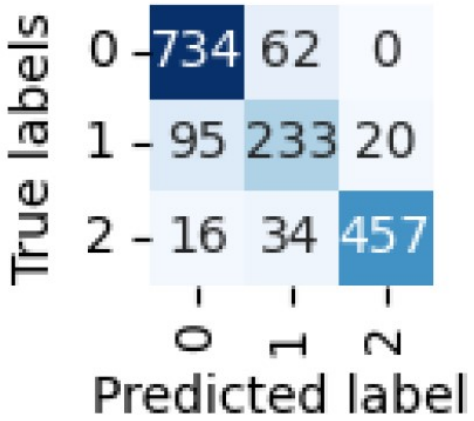
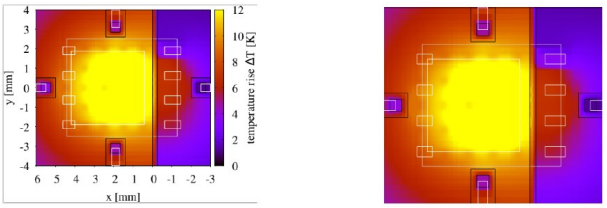


Abb. 3.20: Confusion matrix

To start working with our dataset first we had to extract the PDF files in our dataset and convert them into the PNG format, as all image recognition algorithms accepts PNG format. Moreover, PNG format is better as compared to jpg format as not much information is lost during conversion from PDF to PNG. After converting all PDF files to PNG format we cropped our images to a uniform dimension of 594*666. Here figure shows our raw data in PDF format and final preprocessed image.

To increase the number of images in our dataset for better performance of our model and overcoming the problem of less data we used data augmentation. We flipped our images from top to bottom and added contrast of 1.1 to generate new data. After data augmentation we had approx. 15,000 images in our new dataset that can now be trained and tested. In figure 4 we can see the new images that we obtained from augmentation.

With our new augmented dataset we trained and tested Convolutional Neural Network with



(a) Raw data PDF file (b) Preprocessed Image

Abb. 3.21: Comparison of Raw and Preprocessed Data

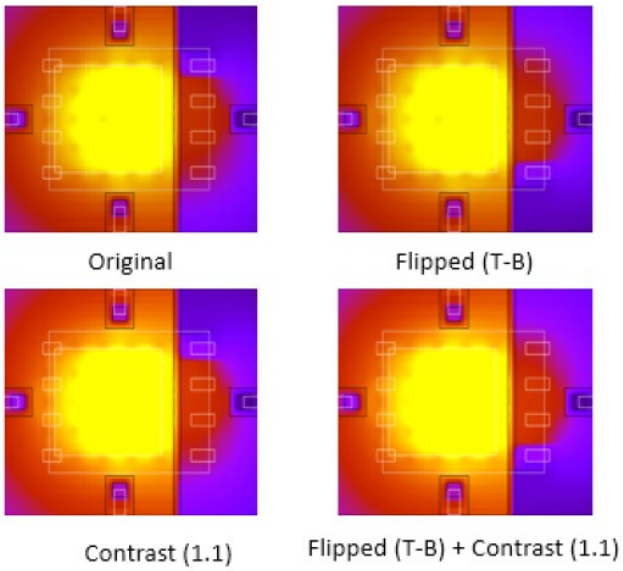
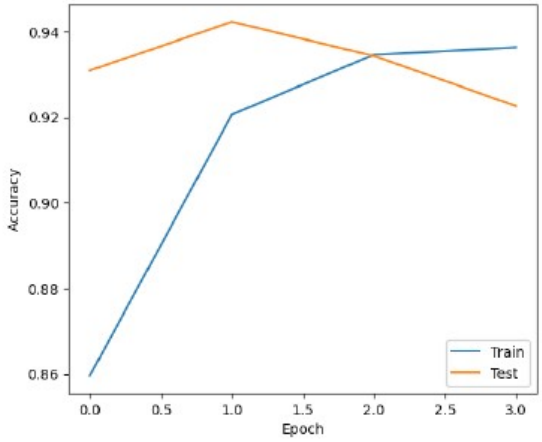


Abb. 3.22: Results for Data Augmentation

80% and 20% data split for training and testing respectively. In table ?? and figure 5 we see the results obtained with our CNN model. These results show that model can effectively learn and map the patterns from simulated data to real world data but testing with real world data would be more challenging as not all parameters and features can be modelled exactly in a simulated system but a close approximation can be made.

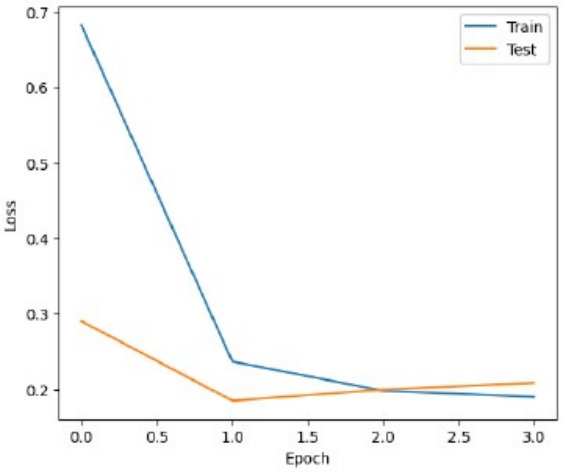
Accuracy	Precision	Recall	F1-Score
92%	86%	82%	0.83

Abb. 3.23: Test results with CNN



(a) Accuracy Curve

Abb. 3.24: Accuracy curve for CNN



(b) Loss Curve

Abb. 3.25: Loss curve for CNN

4 Conclusion and Future Work

The conclusion and future work section summarizes the main points of the paper, discusses the potential for future research, and outlines the next steps in this field.
a) Data Augmentation: Data augmentation is a crucial technique in solder fault detection that can significantly enhance the model's ability to generalize and detect faults in various orientations and configurations. By incorporating data augmentation techniques such as rotation, translation, and scaling, the original solder fault dataset can be expanded with augmented versions of the existing samples, providing the model with a more extensive and diverse set of training data.

Rotation augmentation can be applied to simulate faults occurring at different angles or orientations in real-world scenarios. By randomly rotating the solder fault samples, the model becomes more robust and capable of accurately detecting faults regardless of their orientation. This augmentation technique ensures that the model can effectively identify faults regardless of the angle at which they occur during the soldering process.

Translation augmentation is valuable in simulating faults appearing at different positions within the soldering image. By randomly shifting the samples in different directions within the image space, the model learns to detect faults irrespective of their spatial location. This variation in position helps the model adapt to real-world scenarios where faults can appear at different positions on the soldered components.

Scaling augmentation plays a crucial role when faults can occur in different sizes or when the relative size of the fault compared to the surrounding solder is essential for accurate detection. By randomly scaling the samples, the model learns to identify faults at various scales, improving its ability to handle variations in fault sizes. This augmentation technique ensures that the model can detect faults regardless of their size, from subtle microscopic defects to more prominent anomalies.

By augmenting the original solder fault dataset with these techniques, the model becomes more resilient to variations in fault orientation, position, and size. It learns to generalize better and adapts to a wider range of fault scenarios, leading to improved solder fault detection performance. It is essential to ensure that the augmented samples still represent realistic solder fault patterns and do not introduce artificial biases or distortions during the

augmentation process.

In conclusion, data augmentation techniques such as rotation, translation, and scaling offer a powerful approach to enhance the model's ability to detect solder faults in different orientations and configurations. By expanding the solder fault dataset with augmented samples, the model learns to generalize effectively, leading to improved performance in real-world soldering scenarios where faults can appear in various ways.

b)Model Ensemble: Model ensemble is a powerful technique in solder fault detection that can improve the overall performance and robustness of the detection system. By combining multiple individual models, model ensemble techniques such as bagging or boosting can mitigate the biases and uncertainties associated with a single model.

In the context of solder fault detection, model ensemble works by training multiple individual models, each with a slightly different perspective or focus on the dataset. These individual models can be based on different machine learning algorithms or variations of the same algorithm with different hyperparameters.

Bagging, or bootstrap aggregating, is one popular ensemble technique where each individual model is trained on a bootstrap sample of the original dataset. The final prediction of the ensemble is obtained by aggregating the predictions of all individual models, typically through voting or averaging. Bagging helps to reduce the variance and overfitting of the models, leading to more reliable and accurate predictions.

Boosting, another widely used ensemble technique, works iteratively by training individual models in a sequence. Each subsequent model focuses on the samples that the previous models struggled to classify correctly. By assigning higher weights to the misclassified samples, boosting effectively emphasizes the challenging instances, leading to improved performance. The final prediction is made by combining the predictions of all individual models, typically through weighted voting.

The key benefit of model ensemble in solder fault detection is its ability to combine the strengths of multiple models, compensating for their individual weaknesses. It helps to improve the detection accuracy, handle complex fault patterns, and enhance the generalization capability of the system.

However, it is important to note that model ensemble comes with additional computational complexity and resource requirements. The training and prediction processes involve multiple

models, which can increase the computational cost and memory requirements. Therefore, careful consideration should be given to the computational resources available and the trade-offs between performance gains and implementation constraints.

In conclusion, model ensemble techniques such as bagging and boosting offer a powerful approach to improve the performance and robustness of solder fault detection systems. By combining multiple individual models, ensemble methods mitigate biases and uncertainties, leading to more accurate and reliable predictions. However, the implementation of model ensemble should consider the computational requirements and trade-offs to ensure a practical and effective solder fault detection solution.

c) Real-time Deployment: Integrating the trained models into a real-time production environment would be a significant advancement. This would enable continuous monitoring and detection of solder faults during the manufacturing process, facilitating immediate corrective actions.

d) **Transfer Learning**: Investigating the use of transfer learning, where pre-trained models from related domains are fine-tuned, could expedite the model development process and improve the detection accuracy.

The goal was to develop models that can accurately classify solder faults based on the temperature patterns observed during the manufacturing process.

Several machine learning algorithms were explored, including Decision Tree (DT), Random Forest (RF), K Nearest Neighbor (KNN), Support Vector Machine (SVM), and others. These algorithms were trained and tested on a dataset consisting of temperature data from sensors in order to identify patterns associated with different solder fault types.

The results demonstrated the effectiveness of machine learning models in solder fault detection. KNN, with its ability to classify samples based on their nearest neighbors, performed particularly well in this context. SVM also showed promising results, especially when the dataset was appropriately scaled using a Standard Scaler function.

Furthermore, the project considered the influence of junction temperature as a pivotal factor in solder fault detection. By incorporating this changing concept and dividing the dataset into different regions or labels based on junction temperature and fault size percentage, the SVM model achieved higher accuracy in classification.

Although the project achieved significant improvement in solder fault detection using ma-

chine learning algorithms, there are areas for future improvement and extension. Data augmentation techniques, such as rotation, translation, and scaling, can be explored to enhance the models' ability to detect faults in various orientations and configurations.

Additionally, the implementation of model ensembles, such as bagging or boosting, may further improve overall performance by combining multiple models and reducing individual biases. Real-time deployment of the trained models in production environments would enable continuous monitoring and immediate corrective actions, enhancing manufacturing quality.

Furthermore, the application of transfer learning, leveraging pre-trained models from related domains, could expedite model development and potentially improve detection accuracy by leveraging knowledge from similar applications.

In conclusion, this project has demonstrated the potential of machine learning algorithms, including KNN and SVM, in solder fault detection using temperature data from sensors. By accurately classifying solder faults based on temperature patterns, these models can contribute to improving manufacturing quality, reducing costs, and enabling proactive maintenance. The identified areas for future improvement and extension provide exciting avenues for further research and development in the field of solder fault detection.

5 Verfasser and Matrikelnummer

- **Heeraj Ayyappan -236708**
- **Nikhil Amala Jerrin James Edward Suresh - 235642**
- **Jithin Cherian - 234138**
- **Sonabayim Huseynzade - 234138**
- **Elvedina Sabotic - 127349**
- **Nadeem Shoukath - 236985**
- **Ghulam Mohyuddin - 230300**
- **Abdullah Bin Ali - 230145**

References

[1] Mauro Ciappa. „Selected failure mechanisms of modern power modules“. In: *Microelectronics reliability* 42.4-5 (2002), S. 653–667.

[2] Zhenlei Li und Jincong Wang. „Failure analysis of IGBT bonding wire based on multi-physics coupling“. In: *2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA)*. IEEE. 2022, S. 175–180.

[3] Ui-Min Choi und Frede Blaabjerg. „Separation of wear-out failure modes of IGBT modules in grid-connected inverter systems“. In: *IEEE Transactions on Power Electronics* 33.7 (2017), S. 6217–6223.

[4] Pengju Sun u. a. „Condition monitoring IGBT module bond wires fatigue using short-circuit current identification“. In: *IEEE Transactions on Power Electronics* 32.5 (2016), S. 3777–3786.

[5] Nils Jahn und Martin Pfost. „Void detection in the solder layer between power semiconductor and PCB“. In: *2022 28th International Workshop on Thermal Investigations of ICs and Systems (THERMINIC)*. IEEE. 2022, S. 1–6.

[6] Wu Hao u. a. „Solder joint inspection based on neural network combined with genetic algorithm“. In: *Optik* 124.20 (2013), S. 4110–4116.

[7] Qianru Zhang u. a. „Deep learning based defect detection for solder joints on industrial x-ray circuit board images“. In: *2020 IEEE 18th International Conference on Industrial Informatics (INDIN)*. Bd. 1. IEEE. 2020, S. 74–79.

[8] Cristina Andersson u. a. „Thermal cycling aging effect on the shear strength, microstructure, intermetallic compounds (IMC) and crack initiation and propagation of reflow soldered Sn-3.8 Ag-0.7 Cu and wave soldered Sn-3.5 Ag ceramic chip components“. In: *IEEE Transactions on Components and Packaging Technologies* 31.2 (2008), S. 331–344.