Available online at www.sciencedirect.com

ScienceDirect

Procedia CIRP 107 (2022) 594–599

www.elsevier.com/locate/procedia

55th CIRP Conference on Manufacturing Systems

Explainable Predictive Quality Inspection using Deep Learning in Electronics Manufacturing

Amal Saadallah *[a], Jan Büscher *[b], Omar Abdulaaty[a], Thorben Panusch[b], Jochen Deuse[b,c], Katharina Morik[a]

[a]Artificial Intelligence Group, TU Dortmund University, Otto-Hahn-Str. 12, 44227 Dortmund, Germany
[b]Institute of Production Systems, TU Dortmund University, Leonhard-Euler-Str. 5, 44227 Dortmund, Germany
[c]Centre for Advanced Manufacturing, University of Technology Sydney, Sydney, Australia

* These authors contributed equally to this work. Corresponding authors: amal.saadallah@cs.tu-dortmund.de, jan.buescher@ips.tu-dortmund.de

Abstract

The linkage of machines in the context of Industry 4.0 through information and communication technology (ICT) to cyber-physical systems with the aim of monitoring, controlling, and optimizing complex production systems, enables real-time capable approaches for data acquisition, analysis, and process knowledge generation. In this context, surface mount technology (SMT) in electronics manufacturing is increasingly enhanced by digitalizing the process. This allows the collection and analysis of sensor data to predict the process quality in real-time. Process control interventions can then be derived in a timely manner based on quality predictions. To further support decision-making for process control by domain experts, explanations for the model-based quality predictions should be supplemented in addition. More specifically, we employ a 1D-convolutional neural network for quality prediction of well-defined Fields Of Views (FOVs) of Printed Circuit Boards (PCBs). Explanations for the model's predictions are provided under various perspectives using a heat-mapping-based technique to highlight the contribution of both local and global PCBs' characterizing features to the quality predictions. This helps to reveal the most decisive features for a given quality assignment and understand which process parts are the most responsible for such decision. Finally, the deployment of the model-based predictive and parts of the prescriptive analytics supported by the provided explanations, are achieved using Edge Cloud Computing technology.

© 2022 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the International Programme committee of the 55th CIRP Conference on Manufacturing Systems

Keywords: Process Quality Prediction; Sensor Data; Deep Learning; Explanations; Heat-maps; Edge Cloud Computing; Electronics Manufacturing.

1. Introduction

Due to a combination of different effects such as increasing competitive pressure, globalisation and supply shortages resulting from external factors (i.e. lock down in different countries along the supply chain, chip shortages, etc.), the production of zero-defect products is becoming an important competitive factor for modern and successful electronic manufacturing companies. In SMT manufacturing, expensive high-end inspection systems (such as X-ray machines) are usually used to inspect the quality of high-volume products at the End Of Line (EOL) to ensure the delivery of zero-defect products [11]. These optical inspection systems provide quality assessments that subsequently lead to conclusions about corrective actions to correct or enhance the quality of the product [11]. While these measurements are necessary to prevent the delivery of defective products, quality testing is often prone to become a bottleneck of the production line because of the high time-consuming inspection process. In the era of industry 4.0, the linkage of production devices through information and communication technology to cyber-physical systems in order to monitor and control manufacturing systems, enables real-time capable approaches for data analysis, predictive quality analytics and knowledge discovery. In data analytics, three main stages are differentiated: first descriptive analytics describing "What has happened?", second predictive analytics capturing "What will happen?" and third prescriptive analytics answering the question "What should be done?" [14]. Recently, through the use of low-cost sensors and storage devices, extensive amounts of data have been collected and stored by manufacturers, offering enormous potential not only for real-time process monitoring but also for gaining valuable insights and knowledge about processes [13]. Consequently, model-based quality pre-

2212-8271 © 2022 The Authors. Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (https://creativecommons.org/licenses/by-nc-nd/4.0)
Peer-review under responsibility of the International Programme committee of the 55th CIRP Conference on Manufacturing Systems
10.1016/j.procir.2022.05.031

diction using Machine Learning (ML) models trained on sensor data are employed to replace the offline time-consuming traditional testing procedures. Thereby, supervised learning models algorithms are trained on recorded process data to predict the quality of unseen products [14, 12]. The prediction of the final product quality in conjunction with sample-based physical testing is then used to derive corrective measurements for quality control [12]. Thus, a paradigm shift from descriptive (monitoring and summarising process data) to predictive analytics (using predictive models to forecast possible future quality) has been observed [13]. The application of model-based quality prediction combined with an interlinked production environment leads to real-time capable approaches to resolve quality deviations through corrective measurements early in the process and to improve the overall product quality. Hence, quality prediction in electronics manufacturing industry is a widely discussed topic in literature (e.g. [6]), reflecting the wide diffusion of ML for quality prediction tasks in electronic manufacturing. By now, it exists huge amounts of different ML models, such as Support Vector Machine (SVM), Gradient Boosted Trees (GBT), K-Nearest Neighbors (KNN) and Multi-Layer Perceptron (MLP) [12, 9].

In particular, the utilization of Deep Neural Networks (DNNs) lead to a steady improvement in the accuracy of industrial quality prediction tasks [10]. DNNs are capable of learning abstract features from raw data automatically. In the quality prediction task, these feature represent essentially generalizations of important class-specific criteria to decide whether a piece is defective or not [11]. However, due to their lack of interpretability, DNNs are conceived as black-box models [8]. While accuracy is an important factor for the application, understanding model's predictions is often equally important. It is often necessary to establish a sufficient level of trust in the model, especially when including it in decision-making by domain experts, e.g. process parameters adjustment or early stopping of a process. In literature, several approaches for DNNs explainability have been proposed ranging from visualisation-based approaches, over conceptual explanations to model-based approaches, a comprehensive overview is given in [5]. In this work, we employ a one dimensional Convolutional Neural Network (1D-CNN) for quality prediction in SMT manufacturing. As a baseline we compare the obtained results with conventional methods such as MLPs, Gradient Boosted Trees (GBT) and Random Forests (RF). However, for explainability we use the 1D-CNN given the flexibility that it offers with regards to data reshaping so that a distinction between "global" and "local" features and their corresponding importance can be easily made. The task is to map process features of previous processing steps to binary quality labels of the EOL-testing, differentiating between the "Ok"- and "Not-Ok" (NOk) pieces and can therefore be described as a binary classification. We use a saliency map-based visualisation technique to make the decision of the 1D-CNN explainable to the user. As a further step, the edge cloud computing architecture is described for the model deployment to give the reader a guideline on how explainability methods can be applied in industries for process optimization and recommendation system building.

2. Case Study

The case study is conducted on a SMT production line where a X-ray inspection system is used for quality control at the End Of the production Line, i.e., EOL quality testing. Since the X-ray inspection induces high resource consumption, the quality control is to be made faster by the application of ML model-based quality prediction. This use case has been part of other publications that serve as reference for our experiments [12, 11]. The SMT assembly process is a process chain consisting of the following consecutive steps: First, a raw printed circuit board (PCB) is inserted into a printer via a conveyor belt, then the solder paste is printed onto the PCB. Following this, solder paste inspection (SPI) is applied in a visual inspection station to assess the quality of the solder paste position. Afterwards, individual components are automatically placed on the board and are then transported by conveyor belts to the reflow soldering process, where the applied solder paste is melted in various heat zones to connect the components with the PCB. After the soldering process, the components of the PCBs are examined by an Automatic Optical Inspection (AOI). Depending on the product type, subsequent additional X-ray inspection is carried out. The AOI can be executed in tact time, capturing surface properties. Opposingly, the X-ray inspection, which is used to detect pins located beneath the surface of the components, must be performed in a separate batch process. As a result, a long inspection time is expected.

The case study covers a specific product variant of a connector PCB and considers the corresponding manufacturing process in the SMT line as well as the information from the SPI and the X-Ray inspection. During the manufacturing process, the individual PCBs are grouped together by 48 units as one panel. Depending on the orientation of the pannel, each PCB has different number of pins, 79 for $X1$ orientation (Top) and 52 for $X2$ (Bottom). Since it would take too much time to assess the quality at pin level, the quality information of the panels is aggregated at a field-of-view (FOV) level, which corresponds to the aggregation level of the X-Ray inspection. One FOV consists of 6 PCB boards and is denoted as "NOk" if one PCB is detected as defective, whereas it is declared as "Ok" when all PCBs are defect-free. Similarly, Each PCB is labeled as defective if at least one pin in at least one orientation is defective. Otherwise, the PCB is considered as defect-free. Each data point corresponds to one pin described by a set of numerical features from the SPI (see Table 1).

SPI feature	Height	Shape 2D	Shape 3D	Surface	Volume	Offset X	Offset Y
Unit	%	%	%	%	%	μm	μm

Table 1: Descriptive PCB features on the pin level

For the case study, the dataset covers a period of five production months containing a total of $1,461,037,321$ data points. Because decisions on dynamic X-ray inspection or alternative routings can be made only on higher aggregation levels and because the supervision of the PCBs on the pin-level is not feasible, the solder pins are aggregated to PCB level leading to a

high dimensional quality prediction task. Afterwards, decisions for the quality of FOVs can be derived as explained above.

3. Methodology

3.1. Notations

Let the input data be denoted as a multi-set $\mathcal{D} = (X, Y) = \{(x^{(i)}, y^{(i)})\}_{1\leq i\leq N}$ which consists of $|\mathcal{D}| = N$ input data points $x^{(i)} \in \mathbb{R}^d$ and their corresponding label $y^{(i)}$. The data is transformed such that each data point corresponds to one PCB in a given orientation. Each data point is described by $x^{(i)}$ which consists of a d-dimensional vector of feature values and a corresponding quality label, denoting whether the PCB is "OK" or "NOK". In general, the common goal of predictive ML models is to approximate some true function $f^* : \mathcal{X} \rightarrow \mathcal{Y}$, where $\mathcal{X}$ denotes the input feature space and $\mathcal{Y}$ is the target variable to be predicted. In our case, the predictive ML model is built to predict a discrete binary quality label given an input feature vector. Therefore, $\mathcal{X} = \mathbb{R}^d$. $\mathcal{Y} = \{0, 1\}$ is the target variable that can take two possible values mutually exclusive. If $y_i = 0$, the corresponding PCB is considered as "NOK". In the opposite case, it is treated as "OK". The learning task can be formalized as a binary classification task. The prediction of $\mathcal{Y}$ is carried out by the application of a model $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$. Usually, a fixed structure of f_θ is chosen that is parametrized by some vector $\theta \in \mathbb{R}^p$. Learning from data $X \in \mathcal{X}$ consists then of basically fitting θ to X, so that $f_\theta \approx f^*$. Random Forest (RF) [1], Gradient Boosting Trees (GBT) [4], Multi-Layer Perceptron (MLP) [2] and Convolutional Neural Networks (CNNs) [2], are special classes of predictive models.

3.2. Model Building

3.2.1. Random Forest

A RF is a collection of tree predictors $h_{\theta_k}(X)$, where $k \in [1, K]$ and θ_k are independent and identically distributed (*iid*). For classification problem, the RF prediction is decided by a majority vote over the predicted class outputs of the ensemble of trees. Since $K \rightarrow \infty$, the Law of Large Numbers ensures $PE^*_F = \mathbf{E}_{X,Y}\Big(Y - \overline{h}(X)\Big)^2 \rightarrow \mathbf{E}_{X,Y}\Big(Y - E_\theta h_\theta(X)\Big)^2 = PE^*_t$, where PE^*_F is the generalization error of the RF and PE^*_t is the average prediction for an individual tree defined by $h_\theta(X)$. The convergence in the equation implies that RF does not overfit. Assume that for all θ, the tree is unbiased, i.e., $\mathbf{E}(Y) = \mathbf{E}(X)h_\theta(X)$. Then $PE^*_F \leq \overline{\rho} PE^*_t$, where $\overline{\rho}$ is the weighted correlation between the residuals $Y - h_\theta(X)$ and $Y - h_{\theta'}(X)$ for independent θ and θ'. The inequality pinpoints what is required for an accurate RF. Hence, a low correlation between the residuals of different tree members of the forest via randomization and number of covariate selection and a low prediction error via growing individual trees to maximum depth should be established. In practice, decision trees are created, which keep partitioning the data recursively in the classification space until the amount of variation in the subspace is small. The partition process for RF is greedy and, as a result, does not generally converge to the globally optimal tree. To mitigate this issue, an ensemble of locally optimal trees, where each tree is created by randomly sampling from the original subset, is considered. Such procedure is called bagging [1].

3.2.2. Gradient Boosting Trees

The GBT enhances a decision tree (DT) using the boosting algorithm [4], which is based on the idea of aggregating weak models to form one single strong model. In a DT, the feature space is firstly classified into sub-regions in order to model the dependent variables $x^{(i)} \in \mathbb{R}^d, i \in [1, N]$ for each region. Then, each individual region is further divided into new sub-regions to model the relative variables. Repeating these processes until all the completion conditions are met. Within each region, the best fitness can be achieved by selecting the split point and variables. Among them, the end node count is defined as the size of a single tree. GBT can be built by calculating the classification value p by training a model H via a least-squares regression. Adding an estimator further improves the model in a forward stage-wise strategy: $H_t(X) = H_{t-1}(X) + \eta_t h_t(X)$, where H_t refers to the GBT model consisting of t DTs, t donates the total number of DTs; η_t the learning rate; and h_t the weak learner. A new DT h_t is added to H at each boosting iteration t. The value of h_t can be calculated for each input data point: $H_t(x^{(i)}) = H_{t-1}(x^{(i)}) + h_t(x^{(i)}) = p^{(i)}$ and thus $h_t(x^{(i)}) = p^{(i)} - H_{t-1}(x^{(i)})$, where $p^{(i)}$ is the predicted class probability of $x^{(i)}$ by H_t. Then, a weighted sum function $\hat{H}$ is applied for approximating the classified h_t as given by: $\hat{H}(X) = \sum_{i=1}^{t} \eta_i h_i(X) + const$. Let $\mathcal{H}$ be a set of differentiable functions, and $h_t \in \mathcal{H}, \forall t \in [1, T]$ where T is the total number of DTs. For each DT, the GBT model applies the variables that contribute most to the reduction of loss function L to minimize the ultimate L, that is, applying the empirical risk minimisation principle. The minimisation process can be addressed by the steepest descent method.

3.2.3. Multi-Layer Perceptron

A MLP is a class of feed-forward Artificial Neural Networks ANNs. An ANN incorporates new representations of the raw data in $\mathcal{X}$ through transformations $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ that map the raw features to another space $\phi(\mathcal{X}) = \mathbb{R}^{d'}$, into the learning process, i.e., $\phi = \phi_\theta$, fitted to some data. ANNs learn ϕ_θ in multiple levels of abstraction. Each level corresponds to one layer of the network. The overall ANN can be viewed as nested layer transformations $f(x) = f^{(l)}(f^{(l-1)}(\cdots f^{(1)}(x))$ where l is the depth of the network. Each layer $f^{(i)} : \mathbb{R}^{d^{(i-1)}} \rightarrow \mathbb{R}^{d^{(i)}}$ applies two operations. First, an affine transformation that consists of a weighted sum or convolution, is computed. Then, an activation function, e.g., tanh or ReLU activation, is applied. Each layer is parametrized by the weights used in the affine transformation, while the activation function is usually fixed. The most common used loss function in DNNs for classification task is the categorical cross-entropy loss [2]: $L(x) = -(y \log \hat{y} + (1 - y) \log (1 - \hat{y}))$ with $\hat{y}$ denotes the probability of x having the class y and L the loss of classifying the input time series x. Similarly, we can define the average loss when classifying the whole training set of $\mathcal{D}$ by: $J(\omega) = \frac{1}{N} \sum_{i=1}^{N} L(x^{(i)})$ with ω is the set of weights to be learned by the network. The loss function is minimized to learn the weights in ω using a gradient descent method:

$w = w - \alpha \frac{\partial L}{\partial w} | \forall w \in \omega$ with α is the learning rate of the optimization algorithm. By subtracting the partial derivative, the model is actually auto-tuning the parameters $w \in \omega$ in order to reach a local minimum of J. Since the partial derivative with respect to certain parameters $w \in \omega$ can not be in most of the cases computed directly, a chain rule of derivative is employed using the backpropagation algorithm [7].

In a MLP, the data flows forward to the output continuously without any feedback and is typically composed of several layers of nodes. The first or the lowest layer is an input layer. The last layer is an output layer where the problem solution is obtained. The input and output layers are separated by one or more intermediate layers called the hidden layers.

3.2.4. 1D-Convolutional Neural Network

As their name indicates, CNNs are based on ANNs structure and the principle of convolutions. Convolution is used as an alternative to layers that compute a complete weighted sum in the affine transformation stage where every neuron in the layer is connected to every neuron in the previous layer. Convolution shares weights across all connections. For example, in the discrete one-dimensional (1D) case: $(d * K)(x) = \sum_a \sum_b I(x - a)K(a)$ is a convolution of a data sample d with the kernel (or filter) K. The scalars x and a are the coordinates in the output and the kernel. CNN consists usually of two blocks: convolutional and fully connected [3]. In this work, the convolutional block consists of several convolution layers followed by dropout layer, then a pooling layer and a flatten layer. Dropout is a technique used to prevent the model from overfitting. Dropout works by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each update of the training phase. The pooling layer performs a pooling operation like average or maximum to each of the received feature map from the convolutional layers. After the convolution blocks, the features are fed into a Max Average Pooling (MAP) layer. Then, flattening is applied to convert the data into a 1-dimensional (1-D) array for inputting it to the next layer. The second block is a fully-connected (FC) block that includes the concatenation layer for merged features of the convolutional block using two dense layers and and an output layer with 1 neuron with softmax activation function to output a class probability.

3.3. Explanations Generation

We are mainly interested in predicting and understanding the resulting quality of the FOVs for each orientation of panel, namely X_1 and X_2. Each orientation has a different number of pins where each pin is described by the 7 various physical quantities in Table 1. As a result, the two orientations have different number of features. Therefore, we train a specific machine learning model for each orientation. We refer to these physical quantities in Table 1 that describe all the pins as *global features* and to the set of pins describing each PCB in a given orientation with their corresponding physical quantities as *local features*. It is interesting and more practical to explain the models' decisions from the perspective of both *global* and *local* features, so that engineers can understand which physical quantities needs to be considered more closely and which pins are more prone to contain defects (i.e. which process parts are responsible for such pins). Feature importance analysis can be considered as a tool to provide these explanations [8]. Some models like RF and GBT are equipped with feature importance measures. However, they provide such measures only for the input features. Since the features describing the PCBs are the set of *local features*, we can get feature importance only on the pin level. Also, due to the high number of features, it is very difficult to assess their importance with quick analysis using standard feature importance analysis and visualization, like SHAP Feature Importance [8]. Opposingly, the 1D-CNN can provide different overviews of the input data by allowing the convolutions either along the physical quantities or the pins for each orientation. This would results in two 1D CNNs for each orientation. Afterwards, a visualization-based explanation methods is used for each model to highlight the most important features for each class. In this way, feature importance analysis can be conducted for both *global* and *local* features. In addition, visualization based approaches are much easier to investigate and interpret. The employed visualization method in this work uses the concept of heat-mapping.

Grad-CAM is one of the most popular methods for producing heat maps. Grad-CAM is a generalization of Class Activation Mapping (CAM), a method that does not require a particular architecture. The "Grad" in Grad-CAM stands for "gradient". Grad-CAM is applied to an already-trained neural network after training is completed and the parameters are fixed. We feed the input features into the network to calculate the Grad-CAM heat-map for that input belonging to a given class of interest. The output of Grad-CAM is a "class-discriminative localization map", i.e. a heat-map where the hot part corresponds to a local region in the input that is of most importance for assigning a particular class to this input by the classifier. In order to obtain this map for any class $y \in \mathcal{Y}$, the gradient of y before applying the softmax (i.e. the score of the class y) with respect to feature maps $A_f, \forall f \in [1, f_{maps}]$ of the last convolutional layer, is computed. These gradients flowing back are global average-pooled to obtain the neuron importance weights $\alpha_f^y = \frac{1}{Z} \sum_u \frac{\partial y}{\partial A_f^u}$, where Z is the total number of units u in A_f. This weight α_f^y represents a partial linearization of the deep network downstream from A_f, and depicts the importance of this feature map for the class y. Afterwards, a weighted combination of a forward activation maps is computed: $L^y_{Grad-CAM} = ReLU(\sum_{f=1}^{f_{maps}} \alpha_f^y A_f)$. The ReLU is applied to remove the negative contributions since we are mainly interested in the input part that has a positive influence on the class of interest y.

3.4. Model Deployment

The technical details of the deployment for this particular use case are already described in [12]. We therefore focus on the conceptual details and briefly summarize the architecture. We then discuss the extension part that is necessary for the

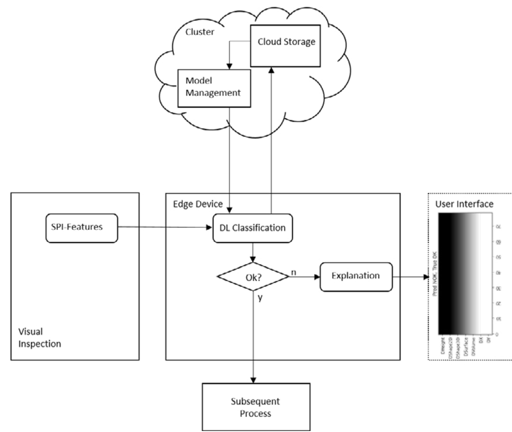

Fig. 1: Explainable predictive quality inspection in SMT manufacturing

use of the explainable predictive quality inspection. The overall architecture is illustrated in Figure 1. Model management, i.e. training, updating and storing the machine learning model, takes place on a computing cluster, which in this case is a Spark cluster, while storage of the data is handled in a datalake or datawarehouse. The SPI features are transmitted directly from the visual inspection machine to the edge device, which in this case is an industrial PC. The three components, i.e. datalake or datawarehouse, computing cluster and edge device, are connected via a network layer. To reduce latency and bandwidth-related issues, the model is stored on the edge device. The machine learning model classifies the PCBs of the FOV level to enable rerouting of the product. If the classifier detects that a PCB is defective, the graphical user interface visualises two perspectives, i.e. the *local* as well as the *global* features providing an explanation by depicting the location of the defective pin areas and the deviating SPI features.

4. Experiments

4.1. Experimental set-up

The available dataset $\mathcal{D}$ is split into 75% for training and 25% for testing. A 10 folds cross-validation procedure is employed to evaluate the models. The reported results are the average metric values over the folds. The dataset reveals a high class-imbalance ratio, therefore, random over- and under-sampling strategies are combined and applied to the training data. For the GBT, RF, we use grid-search procedure to tune their hyper-parameters. For the MLP, we use five hidden layers. The MLP's weights are learnt using a variant of Stochastic Gradient Descent (SGD), namely Adam. The number of training epochs is set to 1000. For the 1D-CNN, depending on the orientation and whether we perform the convolution along the pins or the SPI (*global*) feature, the first convolution contains filters with number equal to the pins with a filter length equal to 7 (number of the SPI features) or the way around. The size and lengths of the filters of the following layers depend on the orientation ($X1$ or $X2$) and the convolution direction (pins or SPI features). The number of neurons in the final softmax classifier is equal to 1 outputting a class membership probability. Similarly to MLP, The model's weights are learnt using Adam with 2000 training epochs.

4.2. Evaluation Metrics

The commonly used metrics for evaluating a classification task are derived using the confusion matrix with the true predicted values $TP = TruePositive$, $TN = TrueNegative$ and the false predicted values $FalsePositive$ and $FalseNegative$. In this notation, the positive class corresponds to the non-defective class and the negative class to the defective class.

The most commonly used metric for classification tasks is the accuracy: $Accuracy = \frac{TruePredictedClasses}{TotalPredictions} = \frac{TP+TN}{TP+FP+TN+FN}$.

However, the accuracy is insensitive to class imbalance and therefore misleading. Therefore, the recall measures the partition of correct prediction to the total amount of true values per class is also reported: $Recall = \frac{TruePredictedValues}{TotalTrueValues}$, $Recall(NOk) = \frac{TN}{TN+FP}$ and $Recall(Ok) = \frac{TP}{TP+FN}$. The recall of the "NOK" class is of a high importance since the manufacturer can not tolerate "NOK" being classified as "OK".

4.3. Results and Discussion

The results are presented in Table 2 and are averaged over the X_1 and the X_2 direction. The Table encloses descriptive statistics trained per PCB, so that in X_1 direction one data point consists of $79 * 7 = 553$ features and in X_2 direction of $52 * 7 = 364$ features. The results of the prediction are then aggregated to the FOV. Referring to Table 2, it can be seen that the Recall

Metric	Recall(NOk)	Recall(Ok)	Accuracy
RF	69.83%	53.76%	73.46%
GBT	73.98%	39.70%	43.26%
MLP	16.08%	**95.13%**	80.15%
$1D-CNN^{global}$	66.92%	42.78%	65.12%
$1D-CNN^{local}$	**75.4%**	43.85%	**74.71%**
GBT [12]	7.6%	29.4%	29.63%

Table 2: Average Performance Comparison of the different classifiers

on "NOK" class has the highest value for the 1D-CNN trained using convolution over the pins. It also maintains a relatively good recall on the "OK" class. The reshaping of the data to highlight the global physical SPI feature has slightly worse performance since aggregation of these features over all the pins is performed. However, a good trade-off of the recall of both classes is maintained. Traditional baselines like RF and GBT have comparable performance. The difference in performance between our GBT and the GBT in [12] is explained by the application of class re-balancing strategy on the training data before training the ML models, which seems to be necessary since a high-imbalance ratio introduces a high bias towards the majority class. MLP has a higher recall compared to the remaining methods on the "OK" class but lower recall (except for GBT [12]) on the class of interest "NOK". From a practical perspective, it is possible to decrease the average testing volume by around 75% when the 1D-CNN is applied, which corresponds to the amount of parts correctly detected by the classifier. Based

on the performance of both 1D-CNNs on the ”NOK” class, we make the argument, that they can be used to provide insights to the user which physical quantities (*global* features) and which specific pins (*local* features) influence the most the quality prediction. To do so, we collect samples that are correctly classified by both 1D-CNNs per class and we compute the Grad-CAM values and average them over all the samples for each class. Computing the maps in an averaged manner over many samples leads to conclusions about process optimization or product specific measurements that can be taken by domain experts. Heat-maps for both classes for both CNNs are shown in Figure 2. As it can be seen in Figure 2, a user can locate and infer the most discriminative features for each class allowing to better understand the importance of *global* physical SPI features and localize the main influencing pins (i.e. *global* features). It can be seen that different *global* and *local* features are highlighted for each class. Most importantly, for correctly predicting the ”NOK” class the focus of the 1D-CNN is mainly on the SPI features ”DX” and ”DY”, a little less on ”DVolume” and ”DSurface”. The remaining features seem to be irrelevant for making the decision towards assigning on average the ”NOK” pieces correctly to the ”NOK” class. This may give some hints to domain experts to investigate further ”DX” and ”DY” signals and more importantly to discover the causes of deviations between these signals for the ”OK” and ”NOK” classes in the manufacturing environment. Looking also to the pin-level, it can be seen that the model’s focus is on some particular pins, namely from 1 t0 5 and 15 to 25. This shows that some pins are more prone to contain defects. This may also give domain experts some hints on investigating which process parts or machines are most often responsible for these particular pins and need to be tracked back.

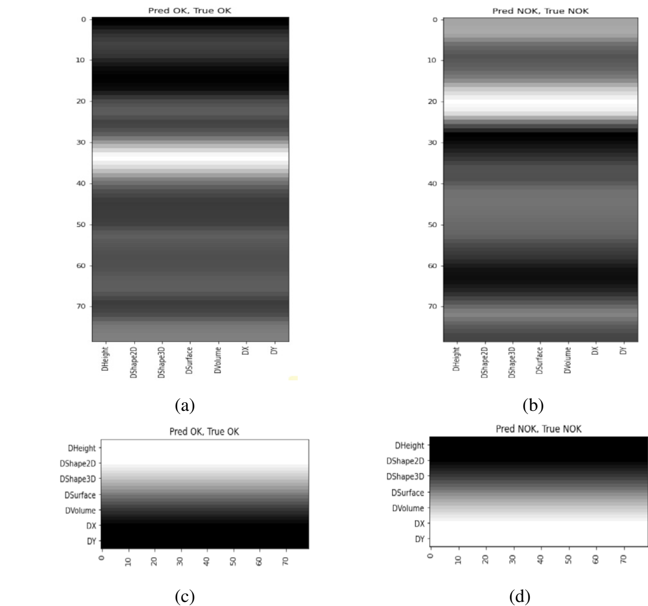

Fig. 2: Heat-maps averaged over many samples for $X1$-direction.(a-b) x-axis shows the SPI-features, y-axis the numbering of the pins, (c-d) the axes are inverted, and the white coloring highlights the most discriminative regions of the PCB (most important input features). (a-b) highlighting *local* features (pins) (c-d) highlighting the *global* features (physical SPI).

5. Concluding Remarks

An explainable model-based quality prediction in SMT manufacturing is presented. This is achieved using 1D-CNN networks and a heat-mapping based approach, namely Grad-CAM. First, the predictive model demonstrates a satisfactory performance for the detection of quality deviations. Second, the heat-mapping based method is used to reveal the causes of quality deviations yielding prescriptive measurements to enhance the quality. Lastly, the deployment within an edge cloud computing-based architecture is described.

Acknowledgements This work is supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 “Providing Information by Resource-Constrained Analysis”, and the Federal Ministry of Education and Research of Germany as part of the competence center for machine learning ML2R (01—S18038A).

References

[1] Breiman, L., 1996. Bagging predictors. Machine learning 24, 123–140.
[2] Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A., 2019a. Deep learning for time series classification: a review. Data Mining and Knowledge Discovery 33, 917–963.
[3] Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A., 2019b. Deep neural network ensembles for time series classification. 2019 International Joint Conference on Neural Networks (IJCNN) , 1–6.
[4] Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Annals of statistics , 1189–1232.
[5] Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D., 2018. A survey of methods for explaining black box models. ACM computing surveys (CSUR) 51, 1–42.
[6] Köksal, G., Batmaz, I., Testik, M.C., 2011. A review of data mining applications for quality improvement in manufacturing industry. Expert systems with Applications 38, 13448–13467.
[7] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324.
[8] Molnar, C., 2020. Interpretable machine learning. Lulu. com.
[9] Nikolai, W., SCHLEGL, T., DEUSE, J., 2021. Feature extraction for time series classification using univariate descriptive statistics and dynamic time warping in a manufacturing environment, in: 2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE), IEEE. pp. 762–768.
[10] Saadallah, A., Finkeldey, F., Morik, K., Wiederkehr, P., 2018. Stability prediction in milling processes using a simulation-based machine learning approach. Procedia CIRP 72, 1493–1498.
[11] Schmidt, K., Thielen, N., Voigt, C., Seidel, R., Franke, J., Milde, Y., Bönig, J., Beitinger, G., 2020. Enhanced x-ray inspection of solder joints in smt electronics production using convolutional neural networks, in: 2020 IEEE 26th International Symposium for Design and Technology in Electronic Packaging (SIITME), IEEE. pp. 26–31.
[12] Schmitt, J., Bönig, J., Borggräfe, T., Beitinger, G., Deuse, J., 2020. Predictive model-based quality inspection using machine learning and edge cloud computing. Advanced engineering informatics 45, 101101.
[13] Stolpe, M., 2016. The internet of things: Opportunities and challenges for distributed data analysis. Acm Sigkdd Explorations Newsletter 18, 15–34.
[14] Stolpe, M., Blom, H., Morik, K., 2016. Sustainable industrial processes by embedded real-time quality prediction, in: Computational sustainability. Springer, pp. 201–243.