Applied Soft Computing 59 (2017) 77–87

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A study of overfitting in optimization of a manufacturing quality control procedure

Tea Tušar[a,*], Klemen Gantar[b], Valentin Koblar[c,d], Bernard Ženko[e], Bogdan Filipič[a,d]

[a] Department of Intelligent Systems, Jožef Stefan Institute, Ljubljana, Slovenia
[b] Faculty of Computer and Information Science, University of Ljubljana, Slovenia
[c] Kolektor Group d.o.o., Idrija, Slovenia
[d] Jožef Stefan International Postgraduate School, Ljubljana, Slovenia
[e] Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia

ARTICLE INFO

Article history:
Received 18 January 2017
Received in revised form 24 April 2017
Accepted 15 May 2017
Available online 22 May 2017

Keywords:
Quality control
Machine vision
Machine learning
Optimization
Overfitting

ABSTRACT

Quality control of the commutator manufacturing process can be automated by means of a machine learning model that can predict the quality of commutators as they are being manufactured. Such a model can be constructed by combining machine vision, machine learning and evolutionary optimization techniques. In this procedure, optimization is used to minimize the model error, which is estimated using single cross-validation. This work exposes the overfitting that emerges in such optimization. Overfitting is shown for three machine learning methods with different sensitivity to it (trees, additionally pruned trees and random forests) and assessed in two ways (repeated cross-validation and validation on a set of unseen instances). Results on two distinct quality control problems show that optimization amplifies overfitting, i.e., the single cross-validation error estimate for the optimized models is overly optimistic. Nevertheless, minimization of the error estimate by single cross-validation in general results in minimization of the other error estimates as well, showing that optimization is indeed beneficial in this context.

1. Introduction

Quality control is essential for improving any manufacturing process. It encourages quality consciousness among workers, enables a more efficient utilization of resources and results in products of better quality at reduced production costs. It is especially crucial in processes with high quality requirements as is the case in automotive industry. There, in many cases only one part per million of supplied products is allowed be defective, which yields strict demands for the involved manufacturing processes as well as their quality control procedures.

This work is concerned with quality control of the manufacturing of graphite commutators (components of electric motors used in automotive fuel pumps) produced at an industrial production plant. More specifically, two different phases of the graphite commutator production process are considered. The first is the *soldering phase*, which consists of soldering the metalized graphite to the commutator copper base. The quality of the resulting copper-graphite joints is crucial since the reliability of end user applications depends on the strength of these joints. The second is the *turning phase* where the commutator mounting hole is formed. The diameter and roughness of the hole directly influence the force required to mount the commutator on the rotor shaft. The minimum and maximum force that can be used in the mounting operation are specified by the customer, which in turn defines the feasible values for the hole diameter and roughness.

Currently, the quality control for both phases is done manually. Automated on-line quality control would bring several advantages over manual inspection. For example, it could promptly detect irregularities making error resolution faster and consequently saving a considerable amount of resources. Moreover, it would not slow down the production line and would be cheaper than manual inspection. Finally, it would not suffer from fatigue and other human factors that can result in errors. This is why we aim for an automated on-line quality control procedure capable of determining:

- whether the joints are soldered well or have any of the known defects (the *soldering problem*), and
- the mounting hole roughness (the *roughness problem*).

* Corresponding author.
E-mail address: tea.tusar@ijs.si (T. Tušar).

http://dx.doi.org/10.1016/j.asoc.2017.05.027
1568-4946/

Measuring the mounting hole diameter is considered trivial and will be exempt from this study.

Such automation can be implemented on the production line by combining machine vision (MV), machine learning (ML) and optimization methods. It consists of assessing the quality of the soldered joints and mounting hole roughness from commutator images captured by a camera. Predictions are made by a ML model that needs to be previously trained on a database of commutator images with known soldered joints quality and mounting hole roughness. Attributes used by machine learning are extracted from the commutator images with MV methods. Most MV methods have parameters that greatly affect their outcome and are at the same time hard to set. This is why optimization is used to find the MV parameter settings that result in a ML model with a low error rate.

While previous work studied different setups for this automated quality control procedure (see [1–3] for the soldering problem and [4] for the roughness problem), this paper exposes the overfitting that emerges when searching for an accurate predictive model. Overfitting is shown for three ML methods with different sensitivity to it (trees, additionally pruned trees and random forests) and assessed in two ways (repeated cross-validation and validation on a set of unseen instances). The original contribution of the paper is the investigation of the effect of overfitting in such a procedure. We wish to test whether optimization can be beneficial despite using an overly optimistic error estimate. This work is an extended version of the initial overfitting study from [5] that included only the soldering problem, was limited to decision trees and used only repeated cross-validation to assess overfitting.

The automated quality control procedure is explained in more detail in Section 2, while Section 3 discusses overfitting. Afterward, Section 4 presents previous work in this domain and other related work. Performed experiments and their results are detailed in Section 5. Finally, Section 6 summarizes the paper.

2. The automated quality control procedure

This section starts with a general overview of the proposed automated quality control procedure followed by more details for the two separate problems.

2.1. Overview

The goal of this automated quality control procedure is to accurately predict the quality of the soldered joints (or mounting hole roughness) from images of the commutator. This entails the following three steps (see Fig. 1):

1. Preprocess the original image (adjust its position, extract the regions of interest etc., see Sections 2.2 and 2.3 for details).
2. Extract image attributes from the preprocessed image using machine vision with the given settings.
3. Predict the quality of soldered joints or mounting hole roughness from image attributes using the given ML model.

The inputs to this prediction are, beside the original image, the settings to MV methods and the ML model. These are retrieved from the optimization procedure presented in Fig. 2.

The goal of optimization is to find the MV settings that yield the best ML model (the one with the lowest prediction error, which is estimated on a set of original images). In optimization terminology, the MV settings represent one *solution* to this optimization problem, while the ML model error is the objective function to be minimized. The optimization procedure starts by preprocessing the whole set of original images. Then, an evolutionary optimization algorithm is used to search for the best MV settings. It starts with

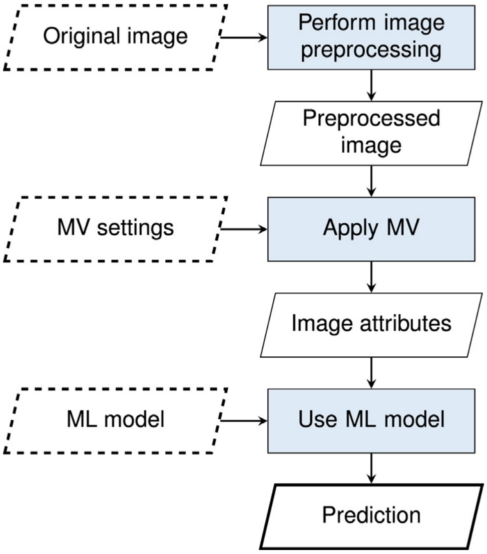


Fig. 1. The procedure for predicting the quality of soldered joints (or mounting hole roughness) from an image of the commutator using the given MV settings and ML model.

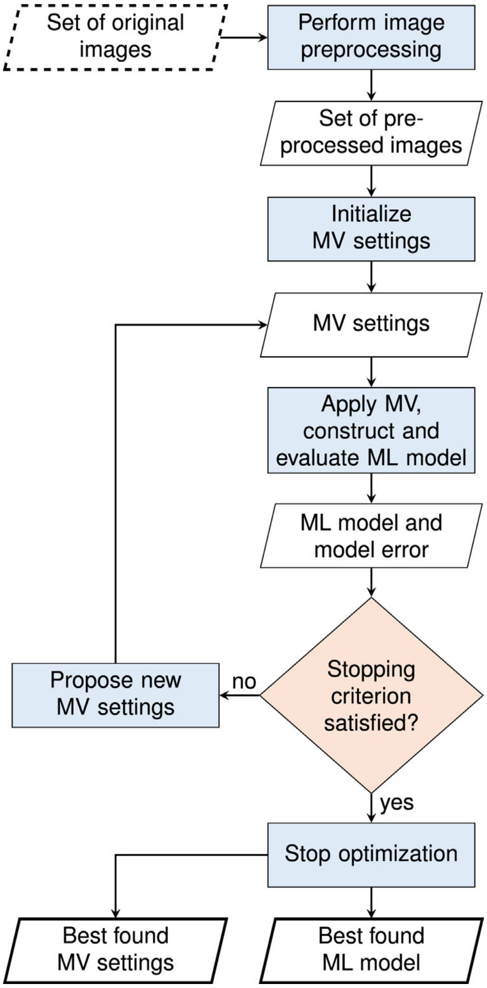


Fig. 2. The optimization procedure that searches for the MV settings that result in the best ML model (the one with the lowest error on the given set of original images).

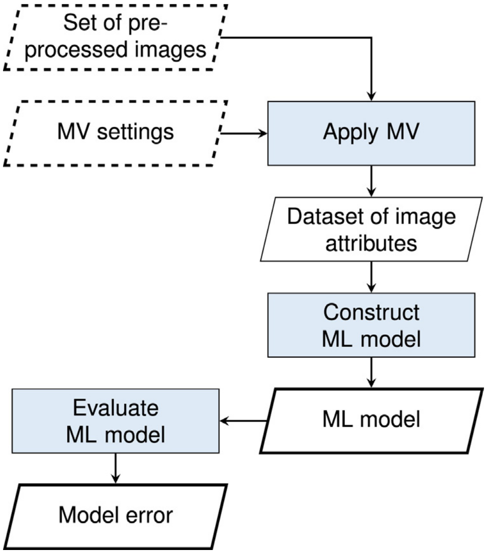


Fig. 3. The evaluation procedure that produces a ML model and its error rate for the given MV settings.

an initial population of random solutions and repeats the following steps until the stopping criterion is met:

1. Evaluate each solution in the population (see also Fig. 3):
   (a) Apply machine vision with the given MV settings to the entire set of preprocessed images to get a dataset of image attributes.
   (b) Construct and evaluate the ML model on the dataset of image attributes.
2. Construct a new population of solutions. This means proposing new MV settings based on the performance of the previously evaluated ones (exactly how this is done depends on the employed optimization algorithm).

While the quality control procedure is equal for both problems in general, it differs in almost every detail.

2.2. The soldering problem

Recall that in the soldering process, the metalized graphite is soldered to the commutator copper base. The resulting joints can be soldered well or have any of the four known defects:

1. *Metalization defect* means visible defects are present on the metalization layer.
2. *Excess of solder* shows as solder spots on the copper pad.
3. *Deficit of solder* is manifested as lack of solder in the graphite-copper joint.
4. *Disorientation* means that the copper body and the graphite disc are not correctly aligned.

Commutators are made up of a number of segments, depending on the model (the considered commutator model from Fig. 4 consists of eight segments). If a single segment has any of the listed defects, the whole commutator is labeled as defective and removed from the production process. Since the inference from the quality of segments to the quality of the whole commutator is rather straightforward, this work is concerned only with predicting the quality of the graphite-copper joints for each segment separately, which means that the input images contain one segment per image.

The defects occur in different regions of the commutator segment. For example, the region where the excess of solder is usually detected is different from the region where disorientation can be observed. Therefore, images of commutator segments can be divided into four regions of interest (ROIs), one for each defect.

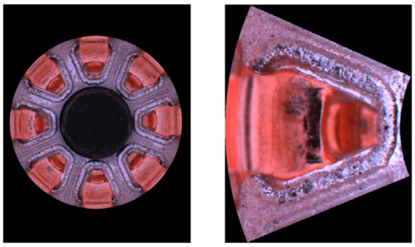


Fig. 4. Image of a graphite commutator with eight segments (left) and a close-up of one of its segments (right).

Because five different outcomes are possible (rare cases where two or more defects appear on a single commutator segment are labeled with just one defect and are not differentiated further), the soldering problem is treated as a *classification problem with five classes*.

Image preprocessing. Fig. 5 shows the image preprocessing steps for this problem. Each input image of a commutator segment has a resolution of 1294 × 964 pixels. First, the image needs to be properly aligned. Next, the four ROIs need to be detected. This is done by applying four predefined binary masks to the image, one for each ROI. Each of the ROIs is further processed as follows. Depending on the ROI, the image in RGB format is converted into a gray-scale image by extracting a single color plane. Based on expert knowledge, red is used for all ROIs except the ROI for excess of solder, which uses the blue color plane.

Attribute extraction. To extract attributes from the preprocessed images, several steps are needed (see Fig. 6). First, a 2-D median filter of size 1 × 1, 3 × 3 or 5 × 5 is applied to reduce noise, resulting in a smoother image. Next, a threshold that can take values from {0, 1, . . ., 255} is used to eliminate irrelevant pixels. Finally, an additional particle filter is employed to remove all particles (connected pixels with similar properties) with a smaller number of pixels than a threshold value from {1, 2, . . ., 1000}. Note that because of the diversity of the defects, it is reasonable to assume that these three MV parameters should be set independently for each ROI. This means that a total of 12 MV parameters need to be set for the soldering problem.

In the last step, the following six attributes are extracted from each of the four ROIs:

- number of particles,
- cumulative size of particles in pixels,
- maximal size of particles in pixels,
- minimal size of particles in pixels,
- gross/net ratio of the largest particle,
- gross/net ratio of the cumulative size of particles.

To summarize, machine vision is used to convert each commutator segment image into a vector of 24 attribute values.

Solution evaluation. Machine vision with given MV settings is applied to commutator segment images with known classification to construct a database of instances, upon which a machine learning model can be built. Classification trees and ensembles of classification trees (random forests) are chosen to ease implementation in the on-line quality control procedure. All instances and attributes from the four ROIs are used to build a single classifier with five classes: no defect, metalization defect, excess of solder, deficit of solder and disorientation. Its performance is used to evaluate the MV settings.

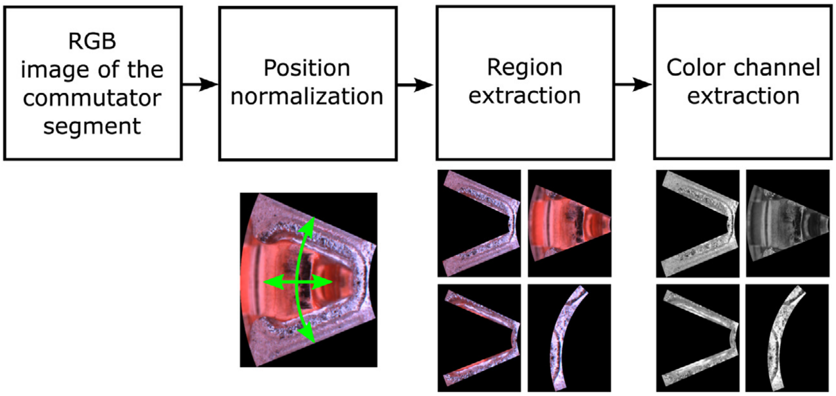


Fig. 5. Image preprocessing steps for the soldering problem. The four images below the region and color channel extraction steps represent (from upper left in clockwise direction) the ROIs for metalization defect, excess of solder, deficit of solder and disorientation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

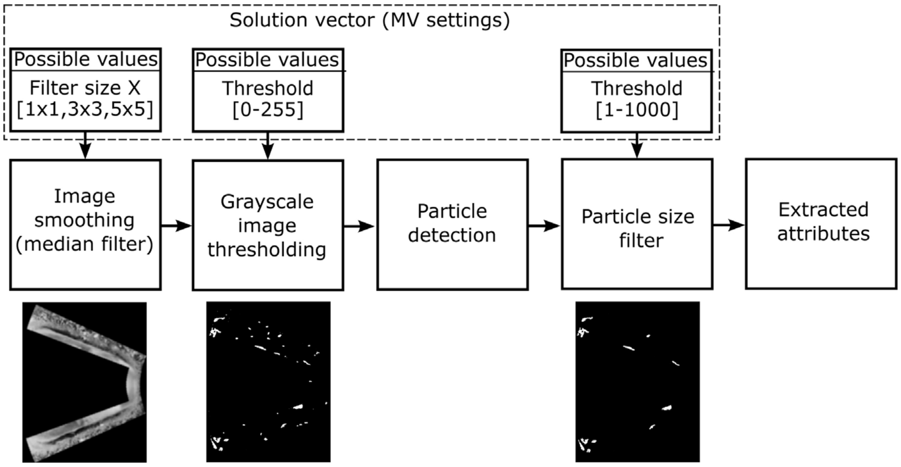


Fig. 6. Attribute extraction steps for the soldering problem.

The quality of a model can be measured with several measures, ranging from classification error to the F-measure. In addition, weights can be used to penalize misclassifications of certain classes differently. While a similar approach would be beneficial also for this problem, where false ‘no defect’ classifications bear more serious consequences than other misclassifications, standard classification error is used in this work, since it is easier to interpret.

2.3. The roughness problem

The roughness problem is concerned with determining the mounting hole roughness from an image of the hole. Surface roughness can be defined by various kinds of parameters, such as amplitude parameters, spacing parameters, and hybrid parameters [6]. The parameter R_z is chosen in this study because it is used at the production plant to measure roughness in the current off-line quality control and was also used in [4]. R_z represents the difference between the maximum peak height and the maximum valley depth from a profile in the sampling length.

Commutators with the roughness parameter value $R_z \leq 16\,\mu\text{m}$ are considered acceptable, while the ones with $R_z > 16\,\mu\text{m}$ unacceptable. The roughness problem could therefore be treated either as a classification problem with two classes (‘acceptable’ and ‘unacceptable’) or a regression problem, where the exact value of R_z needs to be predicted. The latter is of special interest to the user because it could help detect when the tools used in the turning procedure start to wear out and need to be replaced. Consequently, this paper deals with the roughness problem as a *regression problem* of predicting the roughness parameter R_z.

Image preprocessing. The image preprocessing steps for the roughness problem are presented in Fig. 7. The original grayscale image of the commutator mounting hole has a resolution of 2592 × 1944 pixels. It is first transformed using a predefined threshold. The resulting binary image is then used to calculate the image centroid based on the pixel intensity. Since the mounting hole area always contains the highest proportion of high-intensity pixels, the calculated centroid is always positioned at about the same location of the mounting hole, regardless of its position in the image. In the next step, the coordinate system is assigned to the location of the calculated centroid to enable precise positioning of the image mask. Finally, the image extraction mask is applied and the ROI of 700 × 300 pixels is extracted.

Attribute extraction. Fig. 8 details the steps for attribute extraction, which require four parameters to be set. First, a box filter with size from $\{1, 2, \ldots, 200\}$ is applied to smooth the image. Next, the fast Fourier transform (FFT) with two parameters (filter sizes) from $\{1, 2, \ldots, 25\}$ is used to eliminate the noise that results from inhomogeneities in the material. Based on FFT results, a certain proportion of high frequencies is rejected, thus removing noise from the image. Afterward, the surface line profile is measured and certain attributes are extracted already at this step. To obtain the remaining attributes, the Niblack binarization algorithm [7] is applied. Based on its threshold from $\{0, 1, \ldots, 255\}$, it returns a binary image with surface roughness represented by stripes of two colors.

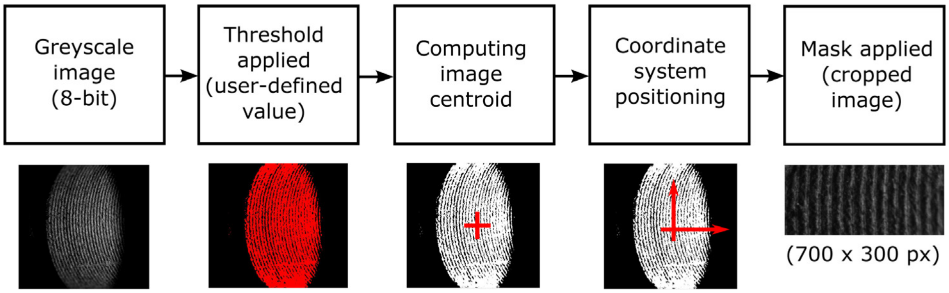

Greyscale image (8-bit)
Threshold applied (user-defined value)
Computing image centroid
Coordinate system positioning
Mask applied (cropped image)
(700 x 300 px)

Fig. 7. Image preprocessing steps for the roughness problem.

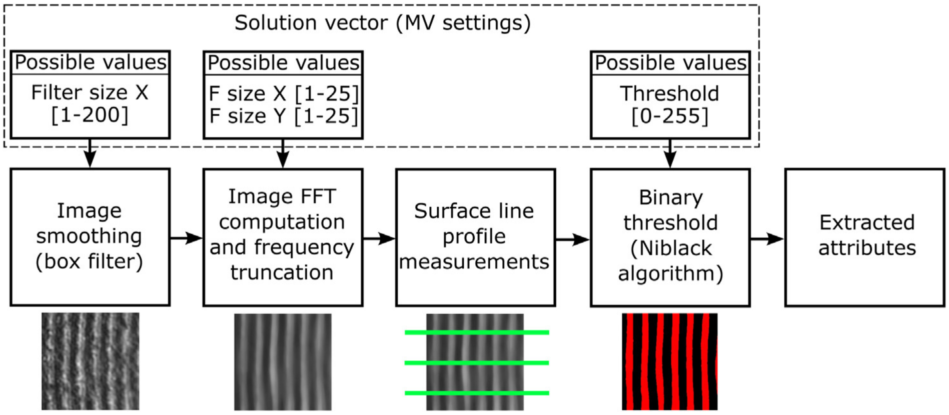

Solution vector (MV settings)
Possible values
Filter size X [1-200]
Possible values
F size X [1-25]
F size Y [1-25]
Possible values
Threshold [0-255]
Image smoothing (box filter)
Image FFT computation and frequency truncation
Surface line profile measurements
Binary threshold (Niblack algorithm)
Extracted attributes

Fig. 8. Attribute extraction steps for the roughness problem. (For interpretation of the references to color in text near the figure citation, the reader is referred to the web version of this article.)

In this way, 24 attributes are extracted. They describe the properties of the captured surface image, such as the grayscale value of pixels along the line profile, the highest grayscale pixel value, the lowest grayscale pixel value, the distance in pixels between the stripes, fast Fourier transform (FFT) values, etc. All attributes are numerical.

Solution evaluation. The quality of MV settings is assessed with the error of regression trees (or ensembles of regression trees) trained on the database of image attributes that correspond to these MV settings. Here, the task is to accurately predict the R_z value. For each commutator, the reference value R_z was computed as an average of three measurements by a contact profilometer to reduce noise from the measurements.

The chosen error metric is the Root Relative Squared Error (RRSE), which was also used in [4]. It measures the error of the ML model in comparison to the error of a simple rule that ignores all predictors and always selects the most frequent value.

3. The pitfalls of overfitting

When building a model, some of the data is used for its *training*, while the rest is used for *testing* its performance. Ideally, both sets should be fairly large, since a lot of data is needed to train a model well, and a lot of data is needed to truthfully assess how it will perform on unseen instances. However, in reality, the data is often scarce and certain compromises need to be made.

One of the most popular approaches to estimating model performance in cases of insufficient data is *k-fold cross-validation*, where the data is split into k sets (folds) of approximately equal cardinality. Next, $k-1$ of the sets are used for training the model, while the remaining set is used for testing (or validating) its performance. This is repeated k times in such a way that each set is utilized for testing exactly once. The average of all performance results is then used to estimate the accuracy of the model built on the entire data.

This and other cross-validation techniques (see [8] for a survey) were envisioned in order to avoid overfitting, i.e., constructing models that describe noise in the data instead of the underlying relationships, since a model that overfits the training data performs poorly on unseen instances. This happens if the model is too complex. For example, it is commonly known that pruned decision trees are less prone to overfitting than unpruned (or less pruned) ones. Overfitting can also be diminished by using tree ensembles, such as random forests [9].

Another source of overfitting exists [10]—if a high number of models is compared on a small set of instances, the best ones are usually those that overfit these instances. This means, for example, that if an optimization algorithm that can produce thousands or even millions of solutions is used to find the best model for a problem, this best found model almost surely overfits the given data. Note however, that this study does not compare multiple models on the same data. The optimization problem at hand resembles more that of attribute selection where a subset of attributes needs to be found so that models using these attributes will achieve good performance. Overfitting has been noticed in attribute selection problems as well [11].

This paper aims to explore whether and to what extent overfitting appears in the presented automated quality control procedure for each of the two problems. Prediction error within the optimization loop is estimated with 10-fold cross-validation (CV), which has been used also in the past studies [1–4]. Overfitting during optimization is assessed using two less optimistic approaches for error estimation: repeated cross-validation and validation on a *hold-out set* (a set of instances that have not taken part neither in building the model nor estimating its performance).

In repeated CV, the error of a ML model is assessed by executing 10-fold CV ten times, each one with a different split of the data to the folds. The resulting averaged error has lower variance than the error estimated using single CV. Comparison of error estimates from repeated and single CV can be used to assess whether single CV overfits to a particular split.

On the other hand, the estimate on a hold-out set of instances is usually less optimistic than the estimate using single CV, though its variance is higher. Comparison of error assessments using single CV and validation on a hold-out set can show whether the single CV is too optimistic.

4. Related work

4.1. Previous work on the soldering problem

Three previous studies [1–3] investigated different setups for the automated quality control procedure, while a recent study [5] tried to address overfitting on the soldering problem.

The first experiment [1] explored whether machine vision, machine learning and multiobjective evolutionary optimization techniques could be employed to find small and accurate classifiers for this problem. Attributes were extracted from the images with fixed MV settings. They were then used to train classification trees capable of predicting if a commutator segment has any of the four defects or is soldered well. The DEMO (Differential Evolution for Multiobjective Optimization) algorithm [12] was used to search for small and accurate trees by navigating through the space of parameter values of the decision tree learning method. The study found this setup to be beneficial, but urged to focus future research on more sophisticated extraction of attributes from the images as this seemed to hinder the search for more accurate classifiers.

The second study [2] presented a different setup for the automated quality control procedure to address the issues from the first study. Instead of optimizing decision tree parameter values, Differential Evolution (DE) [13] was used to search for the best settings of MV parameters such as filter thresholds. The single classification problem with five classes was split into four binary classification subproblems, where each subproblem was dedicated to detecting one of the four defects and used data only from the corresponding ROI. In addition, instead of classification accuracy, the measure to be optimized was set to a function penalizing the portion of false 'no defect' predictions 100 times harder than the portion of false 'defect' predictions. The study found that the new combination of machine vision, machine learning and optimization techniques was powerful and achieved some good results. While optimization with DE always found better parameter settings for MV methods than those defined by domain experts, some subproblems proved to be harder than others. For example, detection of solder excess achieved a satisfactory accuracy, while the detection of metalization defects did not.

The third study [3] investigated the correctness of the implicit assumption from [2] that only attributes of the subproblem-specific ROI would influence the outcome of the classifier for that subproblem. The study found that attributes from other ROIs can be important as well, suggesting that it might be better to address all subproblems together.

Finally, a recent study [5] revealed that the ML model used to estimate prediction error is being overfitted during the optimization of the automated quality control procedure for the soldering problem. Overfitting was detected using repeated cross-validation on classification trees with a small or large degree of pruning (no tree ensembles were employed). No hold-out set was used and the results were not compared against those achieved without optimization (expert-defined MV settings). The focus of [5] was mostly on how to guide the optimization process to diminish cross-validation variance and not questioning whether optimization should be used in such procedures, which is the aim of this work.

4.2. Previous work on the roughness problem

The roughness problem with a very similar setup to the one presented in this paper was previously tackled in [4]. The DE algorithm was used to search for good MV settings in both variants of the problem—classification and regression. While the classification problem was easy to solve (a classification accuracy of 100% was typically achieved in less then 100 examined solutions), the regression problem proved to be more challenging. The paper called for more commutator samples in order to improve on the results.

This is the first time overfitting is being studied on the roughness problem.

4.3. Other related work

Much research and development has been devoted to MV systems for automated quality control in manufacturing. Golnabi and Asadpour [14] provide an overview discussing a design methodology for these systems based on a generic MV model. They analyze the components and functions of MV systems, as well as the key points in designing and applying MV in industry.

Many MV applications specialize in controlling the quality of soldering of various materials. Jiang et al. [15], for example, proposed a MV and background remover for inspection of solder defects on printed circuit boards. Their method first identifies the solder location, thus reducing the amount of information for further processing. Values of predefined features are then extracted from both binary and gray-level images. Finally, the defects are classified based on box plots of the feature values. The resulting classification accuracy is over 97%.

Similarly, Wu et al. [16] presented a MV method for automated inspection of solder bumps on integrated circuit boards. The key steps of the method are image processing, feature extraction, defect detection and their classification into five classes. After the training stage, the method achieves the classification accuracy of nearly 98%.

Evolutionary algorithms have often been deployed to enhance the performance of image processing. In a review of the field, Shimodaira [17] identifies the steps of image processing suitable for involving genetic and evolutionary computation, and reports the tasks on which this was confirmed beneficial. Many of them, such as edge and shape detection, segmentation, and object recognition are essential in manufacturing quality control.

Zheng et al. [18] developed an experimental system to automate the inspection of structural defects on bumpy metallic surfaces. It takes the surface images and uses a genetic algorithm to tune the image processing parameters such as structuring elements and defect segmentation threshold. Experimental results indicate the system can accurately identify the defects and represents a suitable base for a commercial visual inspection system.

Zhu and Chen [19] developed a MV-based system to automate roundness measurement in spring clamps. In an attempt to improve the speed and precision of clamp diameter detection, they first transform the diameter detection problem into the circle detection problem. This is then solved utilizing an enhanced particle swarm algorithm that searches for the points defining the circle. In a comparative study on several industrial test examples the authors demonstrated improved accuracy and efficiency of the proposed system.

As a step beyond the prevailing expert arrangement of image processing operators and evolutionary tuning of their settings, Ebner [20] proposed automatic generation of MV algorithms by

Table 1
The soldering problem domain.

Class	Training set	Hold-out set	Total
No defect	192	20	212
Metalization defect	31	4	35
Deficit of solder	44	5	49
Excess of solder	31	4	35
Disorientation	29	3	32
Total	327	36	363

means of genetic programming where a user shows which object to detect in a sequence of images, while genetic programming generates and tests alternative MV algorithms and iteratively improves them. Acceptable efficiency of this approach is achieved with graphic processing unit (GPU) accelerated image processing.

5. Experimental study

5.1. Setup

The experiments on the soldering problem were performed on the same dataset as in previous studies, which contains 363 instances with an uneven distribution of classes. However, in this work 36 instances were retained in a hold-out set and were not used until the end of the experiments. They were selected manually to account for differences within classes that are visible from the input images (e.g., the sizes of areas with metalization defects can vary substantially within the metalization defect class). The hold-out set was chosen to have a similar distribution of classes (and the differences within classes) as the entire dataset (see Table 1).

For the roughness problem, the entire dataset contained 701 instances, which is more than double of the amount available in [4]. Of these, 71 were selected using random stratified sampling and kept in the hold-out set. Fig. 9 shows the distribution of instances for this problem w.r.t. the value of the roughness parameter R_z. Recall that values $R_z \leq 16\,\mu\text{m}$ are acceptable, while values $R_z > 16\,\mu\text{m}$ are not. The dataset contains no instances with $R_z \in [16, 19]$ because all unacceptable samples were retrieved among commutators returned to the plant due to a reclamation. Among them, there were no commutators with R_z in this range, possibly because the R_z values below 19 were not 'bad enough' to require reclamation.

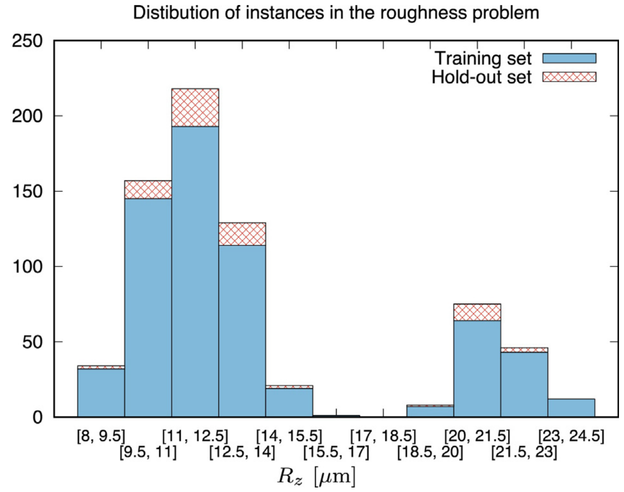


Fig. 9. Distribution of instances for the roughness problem w.r.t. the value of R_z.

Table 2
Optimization parameters for the two problems.

Problem	Population size	Number of generations	Total evaluations
Soldering	80	1000	80 000
Roughness	40	100	4000

All MV methods were implemented using the Open Computing Language (OpenCL) [21], or more precisely, the OCL programming package [22], an implementation of OpenCL functions in the Open Computer Vision (OpenCV) library [23].

All ML models were trained within the Weka machine learning environment [24]. Classification trees were constructed using the J48 algorithm, a Java implementation of the C4.5 algorithm [25]. The 'regular' classification trees were constructed with default J48 parameter values, while additional pruning was obtained by increasing the *m* parameter of the J48 algorithm that defines the minimal number of instances in any tree leaf from 2 to 5. For the regression problem, regression trees were built using the M5P algorithm [26] (an improvement of the M5 tree building algorithm [27]). Again, increased pruning was achieved by changing the *m* parameter from its default value of 4 to 30. Random forests for classification as well as regression were built using the RandomForest algorithm with its default Weka parameters [9].[1]

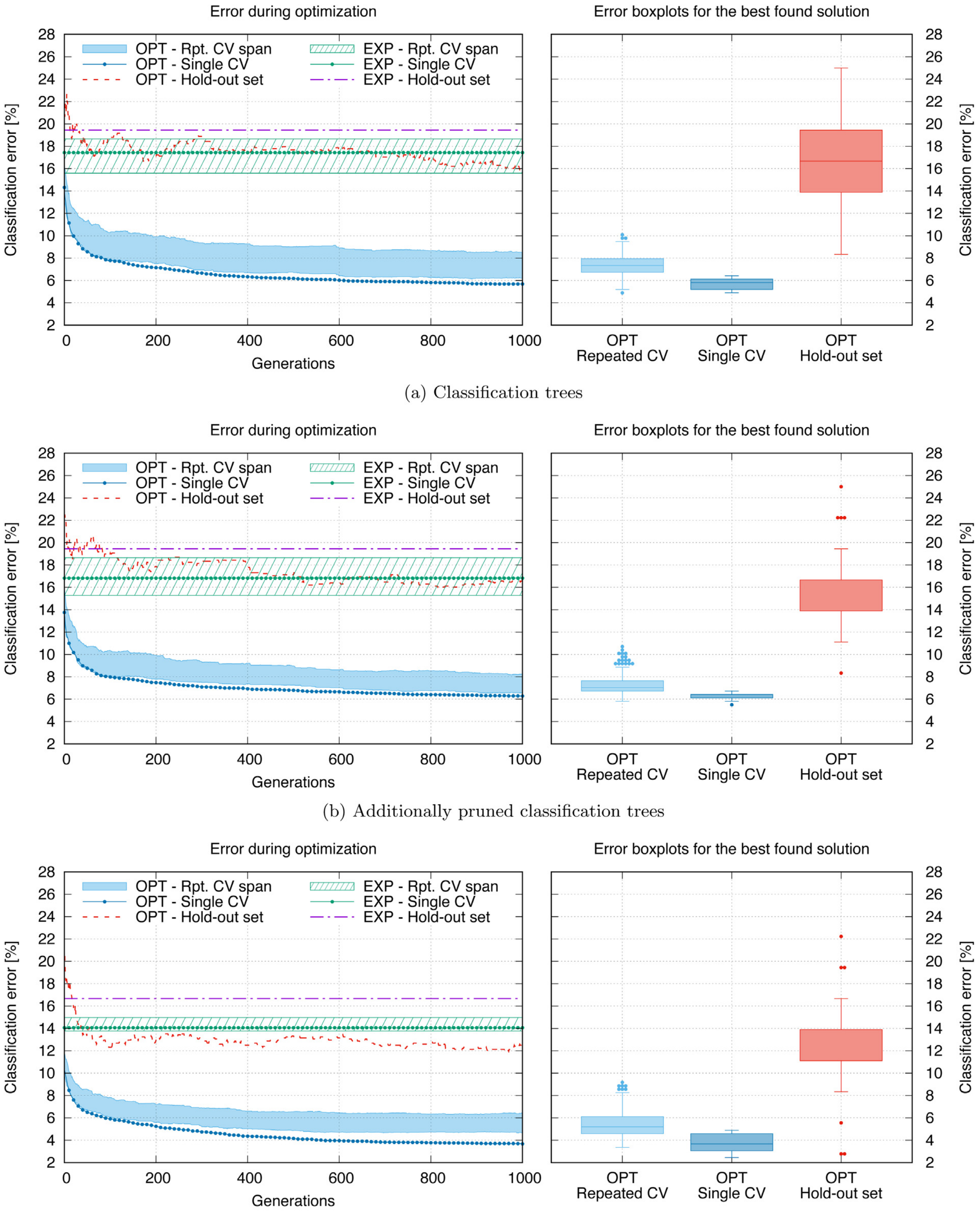
The jDE [28] evolutionary algorithm was employed to search in the space of MV settings. It is a state-of-the-art self-adaptive variant of DE [13] for which only a single parameter (the population size) needs to be set. In addition, the stopping criterion for the optimization procedure is defined as the number of generations. Table 2 presents the setup for these two optimization parameters. The total number of evaluated solutions was larger for the soldering problem because its decision space is larger ($\sim 3.5 \times 10^{23}$ different MV settings for the soldering problem vs. 3.2×10^{7} settings for the roughness problem).

5.2. Experiments

For each of the two problems, three different experiments were performed—with trees, additionally pruned trees and random forests. Each optimization run was repeated 30 times. To ease explanation, a single experiment on the soldering problem using the classification tree algorithm will be described in detail (see the top two plots in Fig. 10). An analogous description could be done also for the other ML algorithms and the roughness problem. There, RRSE is used instead of classification error and the models are regression trees, not classification trees (see Fig. 11).

For each optimization solution, i.e., particular MV settings, a classification tree is constructed on the dataset with attributes retrieved using these MV settings. The classification error for this tree is estimated using a single 10-fold CV. This estimation is used as the optimization objective to be minimized and is denoted with 'OPT – Single CV' in the left-hand side plot. At each generation, the classification error of the tree corresponding to the best found solution so far is additionally assessed in two ways. First, by performing 10-fold CV ten times with different random splits of instances to folds. The resulting span of repeated CV error assessments is marked as 'OPT – Rpt. CV span' in the left-hand side plot. Second, the same classification tree is used to predict the classes of the instances in the hold-out set. The resulting classification error estimate is indicated by 'OPT – Hold-out set' in the left-hand side plot.

[1] It is reasonable to expect that tuning the parameters of ML algorithms would improve results. In fact, this was found in [1] on an experimental setting similar to the one in this paper. However, such tuning could also represent an additional source of overfitting (as explained in Section 3), which would bring another layer of complexity to the study. For this reason, we have decided to use the described fixed ML parameter settings.



(a) Classification trees

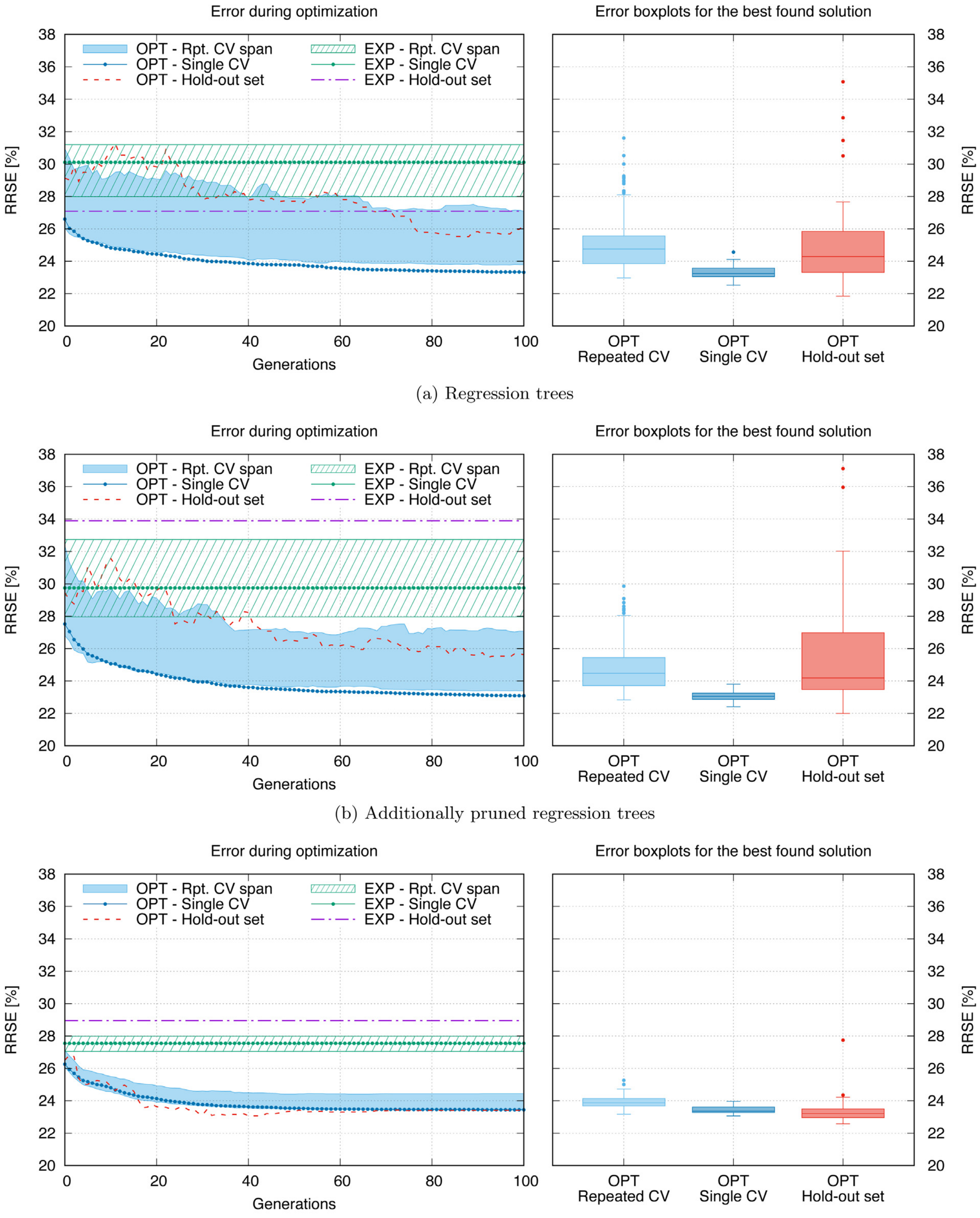
(b) Additionally pruned classification trees

(c) Random forests

Fig. 10. Classification error estimates for (a) classification trees, (b) additionally pruned classification trees and (c) random forests on the soldering problem. The plots on the left-hand side depict the error during optimization (averaged over 30 runs), while the boxplots on the right-hand side show the error distribution for the best found solution. 'OPT' and 'EXP' denote results by optimization and expert-defined settings, respectively. The error is assessed using single CV, repeated CV or the hold-out set. See Section 5.2 for details.

The same three error assessments (with single CV, repeated CV and on the hold-out set) are made also for the classification tree constructed on attributes retrieved from expert-defined settings for the MV parameters. Since a single tree is constructed this way, the outcomes are represented with straight lines. They are prefixed by 'EXP' in the left-hand side plot.

Additionally, we collect all error estimates for the best found solution in each optimization run and present their distribution



(a) Regression trees

(b) Additionally pruned regression trees

(c) Random forests

Fig. 11. RRSE estimates for (a) regression trees, (b) additionally pruned regression trees and (c) random forests on the roughness problem. The plots on the left-hand side depict the error during optimization (averaged over 30 runs), while the boxplots on the right-hand side show the error distribution for the best found solution. 'OPT' and 'EXP' denote results by optimization and expert-defined settings, respectively. The error is assessed using single CV, repeated CV or the hold-out set. See Section 5.2 for details.

using boxplots. While the boxplots of error estimates using single CV and the hold-out set contain 30 data points each (one for each run), the boxplots for repeated CV summarize 300 data points (ten for each run).

5.3. *Results and discussion*

Figs. 10 and 11 show the results for the soldering and the roughness problem, respectively, from which several observations can be made.

First, most of the optimization results show that the average error estimation with single CV is lower than the one with repeated CV. The difference is especially pronounced on the soldering problem with random forests and the roughness problem with regression trees. Even in the cases where the difference is small or nonexistent (the roughness problem with random forests and the soldering problem with additionally pruned classification trees), it is obvious that the error estimation with single CV is more optimistic than the one with repeated CV. This is further confirmed by the detailed investigation of the error distributions of the final solutions shown with boxplots. This suggests overfitting to the particular CV split, since the same split is always used in optimization. Additional evidence for overfitting in this case can be seen by looking at the error estimates of the expert-defined solution. Namely, the single CV error is always found within the span of repeated ones—the expected outcome when no optimization is used.

Next, the comparison between error estimates with single CV and the hold-out set shows that in all but one experiment (the roughness problem with random forests), the single CV produces a much lower error estimate than the hold-out set. This indicates that the single CV estimation is overly optimistic. We have to note, however, that the hold-out set evaluation is very unstable due to its small size. This can also be seen in the boxplots, where the hold-out set evaluation has much larger variance than the other two.

These results are strong evidence that overfitting indeed takes place during optimization. Moreover, on the soldering problem (but not on the roughness problem), the difference among different error estimates is much larger for the optimized solutions than for the expert-defined ones. However, this does not imply that optimization does not provide better solutions. On the contrary, the results show that generally, minimizing the error estimated with single CV also minimizes the other error estimates resulting in better solutions than the expert-defined ones for all six experiments.

These results can be compared also from the perspective of the different ML models. Surprisingly, additional pruning does not necessarily help to reduce the error rate of the decision trees. On the other hand, random forests outperform decision trees, which could be expected since they are known to be robust and relatively resistant to overfitting.

Finally, a note on the practical value of these results. Based on the classification error on the hold-out set, the error on the soldering problem (around 13% for classification trees and random forests) is too large for the requirements of the automotive industry. Additional features should be extracted from the images to help improve on this issue. Better results are achieved on the roughness problem. Consider the predictions by the optimized solution from the median run of the 30 optimizations with random forests (see Fig. 12). Its RRSE on the hold-out set is 22.8% and its mean absolute error on the same set is 0.70 μm, which is satisfactory.

6. Conclusions

This paper investigated overfitting in an automated quality control procedure on two challenging real-world problems in commutator manufacturing. The first is the soldering problem, where a distinction needs to be made among images of joints soldered well and those that have one of the four possible defects. The second is the roughness problem, where the roughness of the commutator mounting hole needs to be predicted from its image. The quality control procedure combined machine vision, machine learning and evolutionary optimization methods with the goal of finding parameter settings for MV methods that yield a ML model with low error.

In this procedure, optimization was used to minimize the model error estimated using single CV. Comparisons of this error assessment to repeated CV and the error estimated on a hold-out set of unseen instances has shown that optimization amplifies overfitting, i.e., the single CV error estimates for the optimized models were found to be overly optimistic. Nevertheless, minimization of the error estimate by single CV in general resulted in minimization of the other error estimates as well, showing that optimization is indeed beneficial in this context.

The obtained results were satisfactory for the roughness problem, but not for the soldering problem, urging for additional work on extracting meaningful attributes from commutator images using MV methods as well as tuning the parameters of ML algorithms.

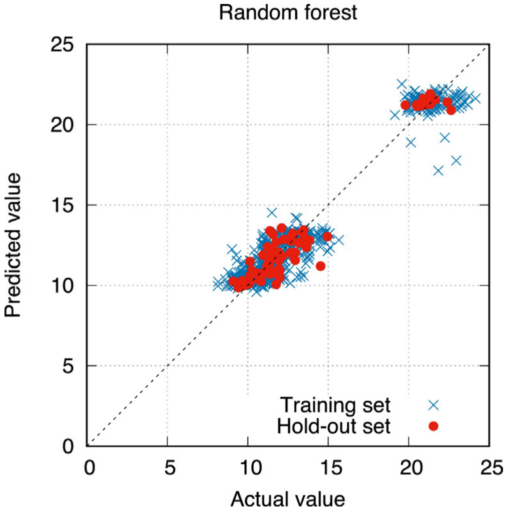


Fig. 12. Predictions on the training and hold-out sets by the optimized solution from the median run for random forests on the roughness problem.

Acknowledgments

This work is part of a project that has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement no. 692286. It was also partially funded by the ARTEMIS Joint Undertaking and the Slovenian Ministry of Economic Development and Technology as part of the COPCAMS project (http://copcams.eu) under Grant Agreement no. 332913, and by the Slovenian Research Agency under research program P2-0209.

References

[1] V. Koblar, B. Filipič, Designing a quality-control procedure for commutator manufacturing, in: Proceedings of the 16th International Multiconference Information Society, IS 2013, vol. A, 2013, pp. 55–58.

[2] V. Koblar, E. Dovgan, B. Filipič, Automating the Design of a Quality Control Procedure for Automotive Components, Technical Report IJS-DP 12323, Jožef Stefan Institute, 2014.

[3] E. Dovgan, K. Gantar, V. Koblar, B. Filipič, Detection of irregularities on automotive semiproducts, in: Proceedings of the 17th International Multiconference Information Society, IS 2014, vol. A, 2014, pp. 22–25.

[4] V. Koblar, M. Pečar, K. Gantar, T. Tušar, B. Filipič, Determining surface roughness of semifinished products using computer vision and machine learning, in: Proceedings of the 18th International Multiconference Information Society, IS 2015, vol. A, 2015, pp. 51–54.

[5] T. Tušar, K. Gantar, B. Filipič, The pitfalls of overfitting in optimization of a manufacturing quality control procedure, in: Proceedings of the 7th International Conference on Bioinspired Optimization Methods and their Applications, BIOMA 2016, 2016, pp. 241–253.

[6] E. Gadelmawla, M. Koura, T. Maksoud, I. Elewa, H. Soliman, Roughness parameters, J. Mater. Process. Technol. 123 (1) (2002) 133–145.

[7] J. He, Q.D.M. Do, A.C. Downton, J.H. Kim, A comparison of binarization methods for historical archive documents, in: Proceedings of the Eighth International Conference on Document Analysis and Recognition, ICDAR 2005, vol. 1, 2005, pp. 538–542.

[8] S. Arlot, A. Celisse, A survey of cross-validation procedures for model selection, Stat. Surv. 4 (2010) 40–79.

[9] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.

[10] A.Y. Ng, Preventing “overfitting” of cross-validation data, in: Proceedings of the Fourteenth International Conference on Machine Learning, ICML'97, 1997, pp. 245–253.

[11] R.B. Rao, G. Fung, R. Rosales, On the dangers of cross-validation. An experimental evaluation, in: Proceedings of 2008 SIAM Conference on Data Mining, SDM 2008, 2008, pp. 588–596.

[12] T. Robič, B. Filipič, DEMO: Differential evolution for multiobjective optimization, in: Proceedings of the Third International Conference on Evolutionary Multi-criterion Optimization, EMO 2005, Vol. 3410 of Lecture Notes in Computer Science, Springer, 2005, pp. 520–533.

[13] R. Storn, K.V. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.

[14] H. Golnabi, A. Asadpour, Design and application of industrial machine vision systems, Robot. Comput. Integr. Manuf. 23 (6) (2007) 630–637.

[15] B.C. Jiang, C.C. Wang, Y.N. Hsu, Machine vision and background remover-based approach for PCB solder joints inspection, Int. J. Prod. Res. 45 (2) (2007) 451–464.

[16] W.-Y. Wu, C.-W. Hung, W.-B. Yu, The development of automated solder bump inspection using machine vision techniques, Int. J. Adv. Manuf. Technol. 69 (1) (2013) 509–523.

[17] H. Shimodaira, Optimization of image processing by genetic and evolutionary computation: how to realize still better performance, in: Proceedings of the IAPR Conference on Machine Vision Applications, 2000, pp. 582–587.

[18] H. Zheng, L.X. Kong, S. Nahavandi, Automatic inspection of metallic surface defects using genetic algorithms, J. Mater. Process. Technol. 125-126 (2002) 427–433.

[19] X. Zhu, R. Chen, Roundness measurement in spring clamps based on a novel particle swarm optimization, Int. J. Mach. Learn. Comput. 5 (3) (2015) 219–224.

[20] M. Ebner, Engineering of computer vision algorithms using evolutionary algorithms, in: Proceedings of 11th International Conference on Advanced Concepts for Intelligent Vision Systems, ACIVS 2009, Vol. 5807 of Lecture Notes in Computer Science, Springer, 2009, pp. 367–378.

[21] OpenCL. The Open Standard for Parallel Programming of Heterogeneous Systems. http://www.khronos.org/opencl/ (retrieved April 10, 2017).

[22] OpenCL Module Within the OpenCV Library. http://docs.opencv.org/modules/ocl/doc/introduction.html (retrieved April 10, 2017).

[23] OpenCV. Open Source Computer Vision. http://opencv.org/ (retrieved April 10, 2017).

[24] Weka Machine Learning Project. http://www.cs.waikato.ac.nz/ml/weka/index.html (retrieved April 10, 2017).

[25] J.R. Quinlan, Induction of decision trees, Mach. Learn. 1 (1) (1986) 81–106.

[26] Y. Wang, I.H. Witten, Inducing model trees for continuous classes, in: Poster Papers, European Conference on Machine Learning, ECML 1997, 1997, pp. 128–137.

[27] R.J. Quinlan, Learning with continuous classes, in: Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, 1992, pp. 343–348.

[28] J. Brest, S. Greiner, B. Bosković, M. Mernik, V. Žumer, Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems, IEEE Trans. Evol. Comput. 10 (6) (2006) 646–657.