Image Processing and Neural Network Optimization Methods for Automatic Visual Inspection

Kawther Aboalam, Christoph Neuswirth, Florian Pernau, Stefan Schiebel,
Fabian Spaethe, Manfred Strohrmann

Faculty of Electrical Engineering and Information Technology – Karlsruhe University of Applied Sciences
kawther.aboalam@h-ka.de
manfred.strohrmann@h-ka.de

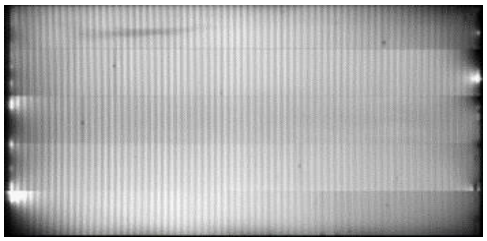
Abstract. In intelligent production lines, methods of automatic visual inspection are used to continuously record process parameters and production results. Using the quality control of thin film solar modules as an example, this paper shows how visual inspections can be automated by using neural networks. The starting point of this automation is an image of a manufactured solar module generated by the inverse operation of the solar cell and the associated electroluminescence.
It turns out that the amount of data generated with every image is very high despite truncation of irrelevant areas and a reduction of the resolution from 1024 x 1024 pixels to 256 x 128 pixels. Without further preprocessing of the data, A neural network would be built that would have 32768 input nodes due to the pixels acquired.
The Convolutional Neural Networks (CNNs) are usually considered for such image classification problems. However, their use increases the complexity of the architecture of the network and thus the number of parameters to be optimized. Therefore, in addition to automated visual inspection, this work addresses the question of how image processing methods can be used for high-quality and efficient implementation.
The Fast Fourier Transform is used for data preprocessing to enable the use of a multilayer perceptron (MLP) rather than a CNN. It is shown that the computation time can be reduced by a factor of 13 by image preprocessing and using the Fast Fourier Transform to compress the dataset. The reduced computation time is a prerequisite for optimizing the neural network hyperparameters. Particle-Swarm Optimization and Genetic Algorithms are implemented and compared to perform a Neural Architecture Search (NAS) for a MLP and to optimize the other hyperparameters. The methods lead to architectures with which an accuracy of more than 99 % is achieved.
The high accuracy of the presented method recommends it for further projects of automatic visual inspection. The method thus allows digitalization in statistical process control and thus contributes to the implementation of the Quality 4.0 vision.

Keywords: Genetic Algorithms, Image Processing, Particle-Swarm, Automatic Visual Inspection

1 Introduction

Many companies are currently adopting systematic approaches to data analysis with the aim of efficiently implementing comprehensive quality management. Following the term Industry 4.0, which refers to the intelligent networking of machines and processes in industry with the help of information and communication technology, the term Quality 4.0 has been introduced by some companies. The aim is to plan and implement intelligent production lines that monitor themselves and constantly provide information about the current production quality. In this context, the term predictive quality refers to measures that forecast decision-making bases for action measures on the basis of continuously logged process parameters and production results. Statistical evaluations of this logged data lead to statistical process control, with which manufacturing deviations can be immediately detected and eliminated.
An essential element of statistical process control is the continuous evaluation of production quality. In many industries, this evaluation is done by manual visual inspection. If this monotonous and tiring work is carried out over a longer period of time, the concentration of the employees decreases and the reliability of the inspection suffers. For some time, therefore, attempts have been made to automate manual visual inspections.
Using the quality control of thin-film solar modules as an example, this paper shows how visual inspections can be automated by using neural networks to classify the quality of the solar modules. The starting point for this automation is the image of a manufactured solar module, which is generated by the reverse operation of the solar cell and the associated electroluminescence. Employees decide on the basis of the image what quality the solar module has and classify it into the categories "OK", "PREMIUM" and "NOK". Bright areas in the image indicate high current densities, dark areas indicate interruptions. Fig.1 compares the images of a solar module of high quality and a solar module with significant manufacturing defects.

High quality solar panel



Solar module with significant manufacturing defects

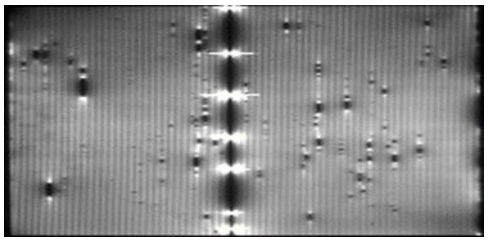


Fig. 1. Electroluminescence of two solar modules of different quality

Employees evaluate the images and classify the manufactured solar modules into classes. Over a longer manufacturing period, images were linked to a quality assessment and stored. The resulting dataset represents the knowledge about this quality assessment and is the starting point for its automation with neural networks.
The paper represents a Multi-Layer-Perceptron (MLP) structure that is optimized using three methods. First, space-filling design, a method of statistical experimental design, is considered. In addition, two methods from statistical optimization are employed, particle swarm optimization and genetic algorithms. The particle swarm optimization is explained in section 4.1
The space-filling design (SFD) - often also called Latin hypercube sampling - is a procedure from statistical experimental design, which is explained below for a two-dimensional experimental space with two hyperparameters (λ_1 ,λ_2). Each hyperparameter is uniformly distributed and has a distribution function F(λ_1) and F(λ_2). In this application, all distribution functions are uniform distributions. If the simulation is to be performed for j = 1 ... J different values, the range of values of the distribution function from 0 ... 1 is divided into J equal intervals for all variables. [1]
Genetic algorithms (GA), like PSO, are nature-inspired algorithms. They are based on Charles Robert Darwin's theory of evolution, which states that life on Earth is subject to constant change. These changes are a result of mutation and selection, with only individuals surviving that are best adapted to their environment. GAs have been developed by John Holland in 1975. [2]
Different representation schemes of GA exist, each leading to different performances in terms of accuracy and computation time. In GA, two common representation methods for numerical optimization problems are distinguished. One is the binary string representation method with a set of binary bits representing a single parameter. The second representation method is to use a vector of real numbers, where each real number represents a single parameter. The GA with real numbers is best suited for optimization in a continuous search space. [3] In this application, each chromosome consists of the hyperparameters described in Table 3.
The structure of the rest of the paper is as follows: After this introduction, Section 2 introduces the image processing and the preparation of the data. Section 3 describes the methodology that has been implemented. Section 4 illustrates the optimization methods for the network architecture. The results and discussions are reported in Section 5. Finally, Section 6 concludes this work.

2 Image Processing and Data preparation

The images of the solar modules generated by electroluminescence are the starting point for the quality assessment. The resolution of these images determines the dimension of the input data for the neural network. The resolution and the dimension of the input data have a significant influence on the quality assessment on the one hand, and on the size of a neural network on the other hand [4]. The aim of image processing and data preparation is to reduce the data to essential information by means of suitable image processing in order to provide the neural network with the required information with a small dimension of input variables.

2.1 Normalization, truncation of irrelevant areas and reduction of resolution

The captured images of the solar modules show strong differences in brightness, which impaired the generalization ability of the neural network. [5] For this purpose, the images are normalized into a number range of 0 ... 255 using a linear scaling.
The resolution of the images acquired is 1024 x 1024 pixels. Some regions in the images show the module frame. These irrelevant areas are selected and removed. The active areas consist of individual strips where individual interruptions or short circuits must be reliably detected. The images can be compressed especially in this direction of these strips. After this first step of preprocessing the resolution is to 256 x 128 pixels.

2.2 Data compression with the Fast Fourier Transform

The image recordings are composed of pixels of different brightness, which form a visible grid structure. Due to the periodicity of the structures, it is convenient to transform the image into the frequency domain using the Fast Fourier Transform. [6]

If the captured image has imperfections, the uniform grid structure of the image is disturbed. Small defects with steep light/dark changes cause high-frequency disturbances that lead to high-frequency spectral components. Larger defects with flat light/dark changes, on the other hand, lead to low-frequency spectral components. With the help of the Fast Fourier Transform, these spectral components can be determined with their respective frequencies. Since the image a(m,n) is a two-dimensional signal, the Fourier coefficients A(k,l) of the Fast Fourier Transform are also two-dimensional. If the image has a width of M and a height of N pixels, M x N Fourier coefficients result in the frequency domain.

$$A(k,l) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} a(m,n) \cdot e^{-j \cdot 2 \cdot \pi \cdot \left(\frac{m \cdot k}{M} + \frac{n \cdot l}{N}\right)} \tag{2.1}$$

The Fast Fourier Transform results in complex Fourier coefficients A(k,l) with magnitude and phase. The magnitude describes the intensity and the phase refers to the spatial orientation of the corresponding harmonic oscillation. The reconstruction of the image is calculated with the inverse Fast Fourier Transform.

$$a(m,n) = \sum_{k=0}^{M-1}\sum_{l=0}^{N-1} A(k,l) \cdot e^{j \cdot 2 \cdot \pi \cdot \left(\frac{m \cdot k}{M} + \frac{n \cdot l}{N}\right)} \tag{2.2}$$

Up to M x N complex amplitudes can be used for this purpose. In the present case, however, it is shown as in Fig. 2 that the structures to be identified are already reproduced sufficiently well with 64 x 32 complex coefficients.

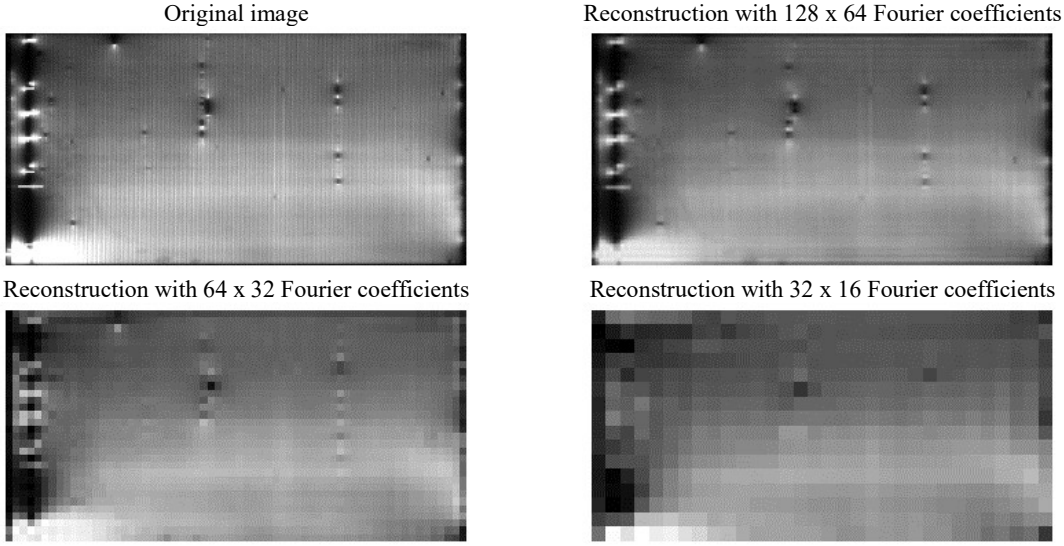


Fig. 2. Original image and reconstructed image with Fourier transforms of different resolution

For quality assessment, the location of defects is of subordinate interest. Therefore, the magnitude of the Fourier coefficients is used for quality assessment only. The number of input nodes is reduced by the transition from 256 x 128 pixels to 64 x 32 real Fourier coefficients.

2.3 Data augmentation

Each image in the dataset was assigned to one of the categories NOK (not okay), OK (okay) and PREMIUM during a manual visual inspection by a staff member. The result was a classified dataset. The classification problem belongs to the category of so-called supervised machine learning. Here, the neural network is repeatedly trained with training data in order to determine a meaningful prediction model for future. If the existing dataset is composed of few, hardly distinguishable images, this impairs the generalization ability of the neural network and leads to so-called overfitting [7]. In overfitting, the network memorizes certain features in the data, making it more intolerant of image data that is not included in the training set. Therefore, the dataset

for a neural network should be composed of many different images. In addition, a balanced dataset should be used. By having the same number of image data per category, a uniform recognition rate is achieved for all categories during training. [5]
Another factor for determining the required data volume is the complexity of the neural network used, in particular the number of parameters to be determined. The training of the neural network serves to optimally represent the dataset to be examined by a suitable selection of parameters. For the determination of each parameter, at least one sample is required. The dataset on which the training is based must therefore be correspondingly extensive.
In order to extend the dataset, the existing image data were modified by random but realistic operations and simultaneously duplicated without degrading the informative reliability in the subsequent classification. The original dataset is unbalanced, images with "OK" category are dominant compared to the categories "PREMIUM" and "NOK". Therefore, image augmentations techniques are not applied equally to the three categories. Fewer operations are applied to the category "OK", because it has much more images than the other two categories. Table 1 gives an overview of the concrete operations used.

Table 1. Overview of the operations used for data augmentation

Quality	Operation
NOK Quality insufficient	Shifting the image by two pixels to the right / left and up / down
	Rotation of the image by 3° to the right / left
	Horizontal reflection
	Combination of individual measures
OK Quality sufficient	Shifting the image by two pixels to the right / left
PREMIUM Premium quality	Shifting the image by two pixels to the right / left
	Rotation of the image by 3° to the right / left
	Horizontal reflection
	Combination of individual measures

There is some similarity between the original images and the generated images, but the neural network perceives them as new input data. The described procedure is called data augmentation and allowed an extension of the dataset from originally 83647 images to 490295 images.

3 Methodology

For the visual pattern recognition of images, so-called convolutional nets are usually used. These types of networks use the discrete convolution operations known from signal processing for feature extraction. This reveals patterns in the images, which are used to classify the data [8], their use increases the complexity of the network structure and thus the number of parameters to be determined. Furthermore, due to the complex convolution operations, the training and testing phases are time and computationally intensive. For this reason, this project uses the Fast Fourier Transform for data preprocessing explained in Section 2.2 to enable the use of a MLP network.

3.1 Network architecture and resulting hyperparameters

The dimension of the input data (input) is given by the number of magnitudes of the Fourier coefficients passed as a 1 x 2048 vector of the network input. Since the neural network is to divide the images of the solar modules into the three categories NOK, OK and PREMIUM, three output nodes (output) are required. The number and size of the hidden layers in between (HL: Hidden-Layer) determine the quality of the classification and have to be chosen depending on the task. Based on empirical investigations, a network structure with 5 hidden layers is determined. This results in the network structure shown schematically in Fig. 3. [9]

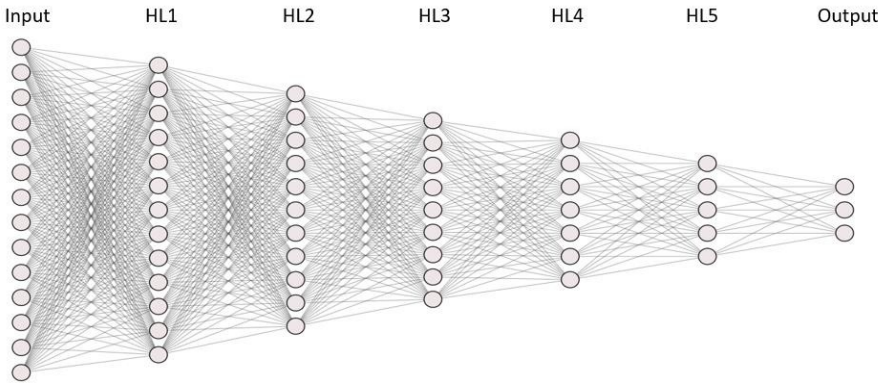


Fig. 3. MLP network structure used with 5 hidden layers

The number of neurons in each hidden layer (HL1 - HL5) has a decisive influence on the quality of the classification [10]. Therefore, they are included as hyperparameters for the optimization of the network.
The Softmax activation function is used for the output layer and the Rectified Linear Unit activation function (ReLU) is used for the hidden layers. The function does not adapt the weights for negative score values because of the vanishing gradient in the training. On the other hand, negative scores are also multiplied by zero and thus excluded from further processing. In [11] it is shown that this implicitly involves regularization. Trials with different activation functions have confirmed that the ReLU function is the best choice for the hidden layers of the network used in this project.
Due to the change of the network parameters, a shift of the output distributions in each layer can occur during the training. As a result, individual weights w_n are no longer updated because they lie outside the functional range of the activation function. This can lead to a worse performance of the neural network. To overcome this problem, so-called batch normalization layers are used. These measure the mean and variance at the output of all hidden layers during the training of the network in each batch. Using adaptive scaling and shifting parameters, the distributions are centered, thus ensuring updating during training. [9]
The selection of a suitable cost function depends on the respective classification problem. In this project, it is a multi-class classification problem with more than two classes, which is why the so-called categorical cross entropy function (Keras) is used. [12]
The optimizer is used to adjust the weights w_n during training so that the loss function reaches its minimum. Depending on the choice of optimizer, different parameters are available for adjustment. Each optimizer has a learning rate α as a hyperparameter, which has a decisive influence on the training process. The learning rate can be optimized during training in order to achieve a faster convergence of the loss function. In preparation for the optimization, optimizers with this adaptive function were tested in advance and the so-called Adadelta optimizer was selected based on the results. In addition to the adaptation of the learning rate, this optimizer has another important hyperparameter, the decay factor ρ. The Decay Factor is used to define which fraction of the previously computed gradients is used in the current computation to update the weights. The information from previous directional derivatives increases the probability of achieving a faster convergence of the error function in the direction of the steepest descent. Learning rate α and decay factor ρ of the chosen Adadelta optimizer have a crucial impact on the performance of the neural network, which is why they are included as hyperparameters to be optimized.

3.2 Training and testing of the neural network

The training of the neural network is divided into epochs. In an epoch, all training images are analyzed once. To keep memory usage and computational capacity efficient during training, all image data in an epoch are divided into batches. The batch size determines after which number of images the weights are updated. After each epoch, the validation dataset is used to check the accuracy of the classification with the currently computed weights and to continue the training if necessary. Finally, in order to evaluate the performance of the network, the network is evaluated with the test dataset at the end of the training. To do this, a dataset must be used that the neural network has not yet run through during training. For these steps, the dataset is divided into training, validation and test datasets (Table 2). Both the number of training epochs and the batch size influence the convergence behavior of the network. For this project, after validation runs, the number of epochs has been set to 100 and the batch size to 128.

Table 2. Division of data into training, test and validation data, total 490 295 images

	Training	Test	Validation
Distribution	83.3 % of the total	16.7 % of the total	10 % of the test patterns
Number of images	408563	81732	8173

4 Optimization methods for an adequate network architecture

The hyperparameters of a neural network can be freely selected. Different hyperparameters lead to different results for the same dataset. The different hyperparameters can be combined in a vector λ. Each element of the vector affects the cost function.
Only vague hints can be found in the literature about which hyperparameters are suitable for the task. Therefore, the hyperparameters in this project are determined using various optimization techniques. The goal is to find a solution vector of hyperparameters λ that gives an optimal result of the cost function. Table 3 gives an overview of the hyperparameters that are determined in the following optimization step. The parameters HL1 ... HL5 refer to the neural network architecture, while the learning rate hyperparameter α and the decay factor ρ refer to its training. The definition of the limits in Table 3 is based on the validation runs with the neural network represented in Section 3.1.

Table 3. Limitation of the experimental space for the optimization procedures

Hyperparameter	Lower limit	Upper limit
Number of neurons in the 1st hidden layer, HL1	100	9000
Number of neurons in the 2nd hidden layer, HL2	50	5000
Number of neurons in the 3rd hidden layer, HL3	30	3000
Number of neurons in the 4th hidden layer, HL4	20	2000
Number of neurons in the 5th hidden layer, HL5	10	1000
Learning rate, α	0.0001	1.0
Decay factor, ρ	0.0001	0.9999

Three different methods are used for optimization. On the one hand, the space-filling design, a method of statistical experimental design, is explored. In addition, two methods from statistical optimization are used, the Particle-Swarm-Optimization and the Genetic Algorithms. The Particle-Swarm Optimization (PSO) algorithm is illustrated below.
The Particle-Swarm Optimization (PSO) algorithm is one of the most widely considered and applied algorithms in the literature for stochastic optimization problems. Mathematically, its objective is to find the global minimum of a multidimensional and nonlinear cost function. It is based on the flocking behavior of birds. It uses three main interactions of the bird flock: separation, alignment and cohesion. Due to the flocking behavior and the associated large number of test points, there is a high probability that the algorithm will find a global optimum. [13]
The first step in the PSO is the random initialization of the particles. Each individual particle with index j represents a random selection of a combination of all hyperparameters as a numerical vector λ_j. In addition, the initial velocity v_j of each particle is randomly determined. For each particle, the cost function is calculated.
Based on the function values of all particles at time k, new particle positions are determined for all particles. For this purpose, the individual velocity $\underline{v}_j$ at time k+1 is determined for each particle.

$$\underline{v}_j[k+1] = w \cdot \underline{v}_j[k] + c_1 \cdot r_1 \cdot \left(\underline{p}_{j,BEST}[k] - \underline{\lambda}_j[k]\right) + c_2 \cdot r_2 \cdot \left(\underline{g}_{BEST}[k] - \underline{\lambda}_j[k]\right) \tag{4.1}$$

The first summand consists of the weighted velocity at the last time point k. The weighting factor w describes which part of the velocity is retained. The term is also called the inertia term because of the analogy to

mechanical mass inertia. The larger the weighting factor w is chosen, the higher is the velocity and thus the distance the particle travels in the experimental space.
The second summand stands for the individual cognitive component and evaluates the previous best position $\underline{P}_{j,BEST}$ of particle j (Particle Best) as well as its current position $\underline{\lambda}_j[k]$. The difference moves the particle in a new direction and is weighted by the factors c_1 and r_1.
The third summand represents the social swarm component. It consists of the weighted difference of the best result of all particles of the swarm g_{BEST} (Global Best) and the current position $\underline{\lambda}_j[k]$. The difference moves the particle in a new direction and is weighted by the factors c_2 and r_2.
The weighting of the components is done using the factors c_1 and c_2, the so-called cognitive and social speedup coefficients. To add a stochastic component to the search algorithm, the factors r_1 and r_2 are used, which are random and uniformly distributed in the range between 0 and 1. This ensures that a particle can also move randomly into new areas.
Using the individual velocity $\underline{v}_j[k+1]$, the future position $\underline{\lambda}_j[k+1]$ of the individual particle j is determined.

$$\underline{\lambda}_j[k+1] = \underline{\lambda}_j[k] + \underline{v}_j[k+1] \tag{4.2}$$

For graphical visualization, this update is shown in Fig. 4 for one particle.

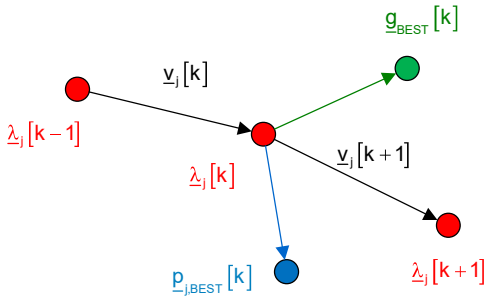


Fig. 4. Visualization of Particle-Swarm-Optimization

The optimization ends when the search function reaches a defined accuracy with the determined hyperparameters or the maximum number of iterations is reached.

5 Results and discussions

The optimization methods are applied to a 7-dimensional test space according to Table 3. The parameterization of the methods is largely independent of each other. Only for the particles and chromosomes identical parameter configurations are used as starting points in order to achieve a better comparability of the results between PSO and GA. Furthermore, the number of particles and chromosomes is chosen to be the same with J = 40. In order to keep the scope of the experiment approximately the same for all methods, the number of iterations for PSO and GA is limited to 14. This results in a total of 560 parameter configurations to be examined. For the space-filling design, there are 500 parameter configurations. Based on the selected initialization for each method, the following results are obtained, which will be discussed in more detail.

5.1 Space-Filling Design

The space-filling design uses the entire range of values of each hyperparameter to achieve the best possible coverage of the experimental space. This idea can be well understood in Fig. 5. As an example, the accuracies are shown here as a function of two of the seven hyperparameters visualized in three-dimensional space. For the remaining hyperparameters, which are not listed, this arrangement applies analogously. The resulting accuracies lie between 96.99 % and 99.08 %. Low accuracies occur except for a few outliers for learning rates of $\alpha < 0.2$.

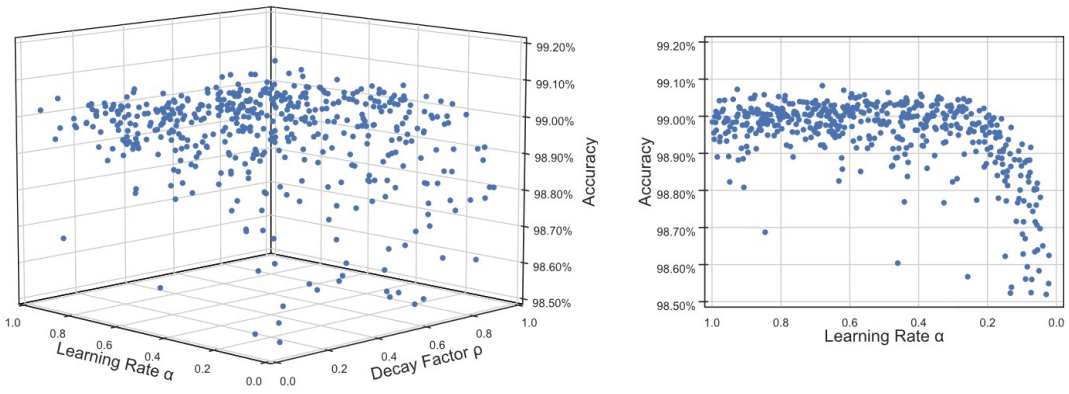


Fig. 5. Space-filling design - Accuracy as a function of the parameters learning rate and decay factor

5.2 Particle-Swarm-Optimization

The results from PSO are shown in Fig. 6. As with the SFD, the PSO delivers poor accuracies for learning rates $\alpha < 0.2$. Due to the selected starting points, the resulting accuracies lie in a larger scatter range between 80.71 % and 99.11 % compared to the SFD.

The vertical gradients in sections in the two-dimensional view characterize the swarm behavior during the optimization. Likewise, the denser arrangement of the samples describes the convergence of the swarm towards the identified optimum, which is here located at $\alpha = 0.3686$. Based on Fig. 6, it is also clear that the PSO algorithm does not include the complete range of values of the learning rate in the optimization. Instead of an exploratory search in all levels of the experimental space, the determination of the optimum is locally limited. The reasons for this lie in the selected parameterization of the PSO algorithm and the position of the starting points in the experimental space.

Although the maximum accuracies differ only marginally, the PSO achieves a slightly higher accuracy of 99.11 % than the SFD with 99.08 %, despite incomplete coverage of the entire test space.

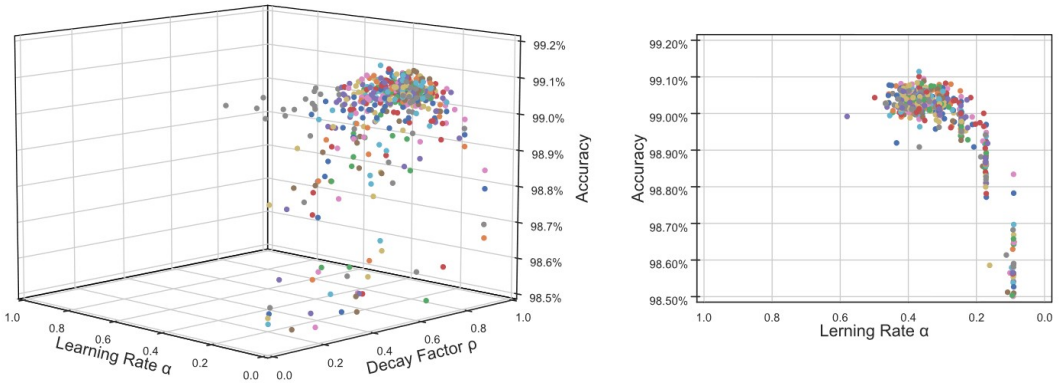


Fig. 6. Particle-Swarm-Optimization – Accuracy as a function of the parameters learning rate and decay factor

5.3 Genetic algorithms

The focus on samples in the optimal region is even more pronounced when genetic algorithms are used. The results obtained with this optimization method are shown in Fig. 7. The individual chromosomes quickly concentrate to values close to $\alpha = 0.38$ and $\rho = 0.75$, despite identical initial conditions as in particle swarm optimization. A larger mutation rate could be used to parameterize the algorithm so that it also covers the experimental space further.

Using the Genetic Algorithms, a maximum accuracy of 99.10 % is achieved. It is between the accuracy of 99.11 % achieved with the PSO and the accuracy of 99.08 % achieved with the SFD.

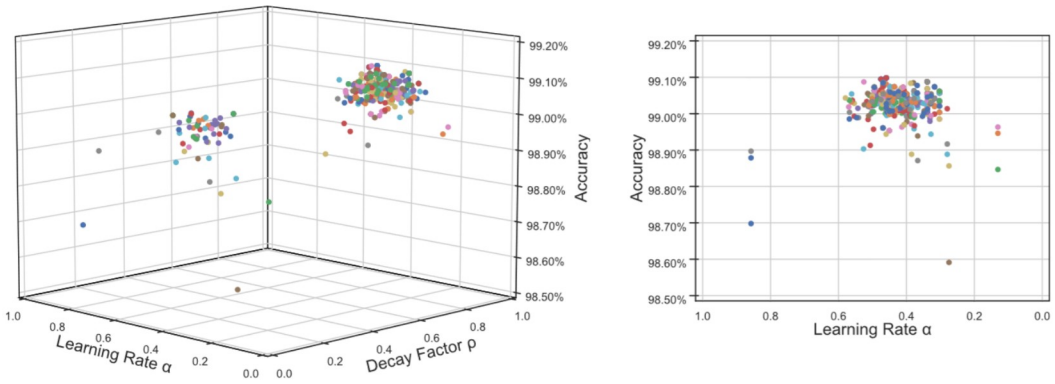


Fig. 7. Genetic algorithms - accuracy as a function of the parameters learning rate and decay factor

5.4 Comparison of the optimization methods

Due to the different optimization methods, a direct comparison of the results between the methods used is not possible. The PSO and the GA use an iterative and population-based approach to optimization, respectively. The SFD, on the other hand, is based on covering the experimental space as uniformly as possible.
In order to evaluate the performance of the methods, the values of the highest accuracy and the mean value of the accuracies per iteration or generation are used. They are shown in the two diagrams in Fig. 8. The results from the SFD are listed as a constant value.

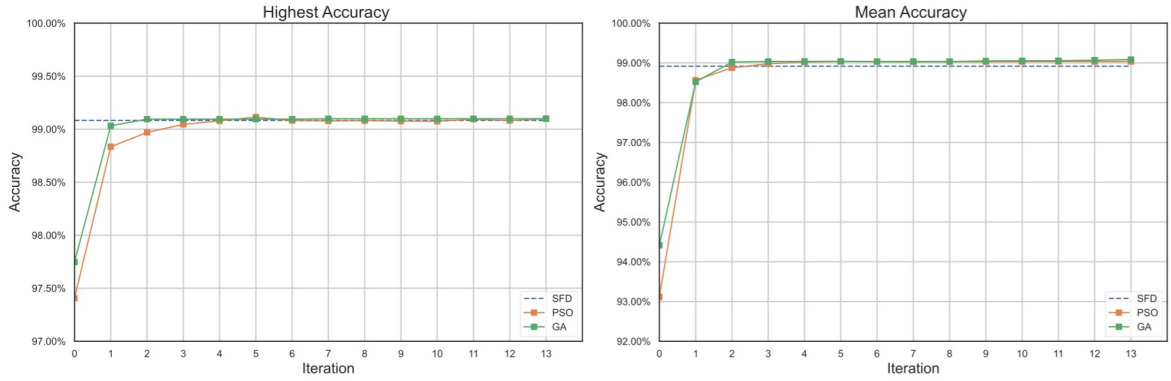


Fig. 8. Comparison of results

Already after the first iteration the accuracies show very high values of 98.83 % (PSO) and 99.03 % (GA). It becomes clear that the optimization with the GA converges faster than that with the PSO. The best values in each case are achieved with the parameter combinations given in Table 4.

Table 4. Tabular comparison of the optimum settings for the different methods and the achieved accuracies

Hyperparameter	SFD	PSO	GA
Number of neurons in the 1st hidden layer, HL1	7073	4728	5562
Number of neurons in the 2nd hidden layer, HL2	3513	3719	5000
Number of neurons in the 3rd hidden layer, HL3	1561	2274	2808
Number of neurons in the 4th hidden layer, HL4	866	1800	827
Number of neurons in the 5th hidden layer, HL5	866	1000	888
Learning rate, α	0.6795	0.3686	0.4593
decay factor, ρ	0.8084	0.8426	0.9272
accuracy	99.08 %	99.11 %	99.10 %

The values show that the best results were produced with different settings. This suggests that achieving minimum requirements results in high accuracy and that the maximum accuracies are local maxima that are achieved by chance. The results depicted in Fig. 8 have always been calculated with the same seed to improve comparability, thus the learning process has always started with the same initial situation. In order to test whether the differences between the optimization methods can be generalized, the neural networks were trained for 40 different seed values. The results were reproducible, all accuracies lie in the narrow range of 99 - 99.1%. The median of the Genetic Algorithms (99.05 %) is larger than the median of the Particle Swarm Optimization (99.04 %) and the median of the Space Filling design (99.03 %).

6 Conclusion

This paper presents a method for automatic visual inspection of solar modules. It uses a supervised learning method with a labeled dataset to build a neural network that assigns the labels NOK, OK and PREMIUM to the modules with an accuracy of more than 99 %.
A prerequisite for the application of neural networks was the preprocessing of the data, which consisted of normalization, truncation of irrelevant areas and reduction of resolution. Fast Fourier Transform was used to compress the data. These measures reduced the original input data shape form 1024 x 1024 down to the applied data shape of 64 x 32, which results in a reduction of computation time by a factor of 13. Data augmentation was used to expand and balance the dataset.
The shortened computation time was a prerequisite for optimizing the hyperparameters of the neural network. Space-filling design, particle warm optimization and genetic algorithms were used and compared for optimization. All optimizations resulted in architectures that achieved over 99% accuracy. The Genetic Algorithms showed slight advantages in terms of the achieved accuracy
The high accuracy of the presented method and the use of genetic algorithms demonstrate the performance of the method and recommend it for further projects of automatic visual inspection. The method thus allows digitalization in statistical process control and thus contributes to the implementation of the Quality 4.0 vision.

References

1. Siebertz, K.: Statistical Design of Experiments, Berlin: Springer- Verlag GmbH, 2017.
2. Holland, J.: Adaptation in natural and artificial systems, The University of Michigan Press, 1975.
3. Jong-Wook K., Sang Woo K., PooGyeon P. and Tae Joon P.: On the similarities between binary-coded GA and real-coded GA in wide search space, Proceedings of the Congress on Evolutionary Computation. CEC'02, Honolulu, HI, USA, 2002, pp. 681-686 vol.1.
4. Sabottke C. and Spieler B.: The Effect of Image Resolution on Deep Learning in Radiography, Radiology: Artificial Intelligence, Vol. 2, No. 1, 2020.
5. Schiebel, S., Spaethe, F.: Solar cell classification, Project work, Karlsruhe University of Applied Sciences, Karlsruhe, 2019.
6. Strohrmann, M.: Systems Theory Online: www.hs-karlsruhe.de/mesysto, accessed on 07.04.2021.
7. YANG, S., et al. Image Data Augmentation for Deep Learning: A Survey. arXiv preprint arXiv:2204.08610, 2022.
8. Keiron O., Ryan N., An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458, 2015.
9. Pernau, F.: Visual Quality Assurance with Neural Networks Master's thesis, Karlsruhe University of Applied Sciences, 2020.
10. Brownlee J., Machine Learning Mastery, Available: https://machinelearningmastery.com/how-to-configure-the-number-of-layers-and-nodes-in-a-neural-network/. [Zugriff am 14 März 2020].
11. Vardi, G., Shamir, O.: implicit Regularization in ReLU Networks with the Square Loss, Weizmann Institute of Science, arXiv:2012.05156v2 [cs.LG] on 15.12.2020.
12. Keras: Keras API Refrence - Probabilistic losses, https://keras.io/api/losses/probabilistic_losses/, accessed on 16.04.2021.
13. Neuswirth, C.: Optimization of Hyperparameters of Neural Networks with a Particle Swarm Optimization, Project Thesis, Karlsruhe University of Applied Sciences, 2020.