Convolutional neural network applied for nanoparticle classification using coherent scatterometry data

Kolenov, D.; Davidse, D.; Le Cam, J.; Pereira, S. F.

DOI
10.1364/AO.399894

Publication date
2020

Document Version
Final published version

Published in
Applied Optics

applied optics

Convolutional neural network applied for nanoparticle classification using coherent scatterometry data

D. Kolenov,[1,*] D. Davidse,[1] J. Le Cam,[2] and S. F. Pereira[1]

[1]Optics Research Group, Imaging Physics Department, Faculty of Applied Sciences, Delft University of Technology, Lorentzweg 1, 2628 CJ Delft, The Netherlands
[2]Institut d'Optique Graduate School, 2 Avenue Augustin Fresnel, 91120 Palaiseau, France
*Corresponding author: d.kolenov@tudelft.nl

Received 24 June 2020; revised 18 August 2020; accepted 21 August 2020; posted 21 August 2020 (Doc. ID 399894); published 18 September 2020

The analysis of 2D scattering maps generated in scatterometry experiments for detection and classification of nanoparticles on surfaces is a cumbersome and slow process. Recently, deep learning techniques have been adopted to avoid manual feature extraction and classification in many research and application areas, including optics. In the present work, we collected experimental datasets of nanoparticles deposited on wafers for four different classes of polystyrene particles (with diameters of 40, 50, 60, and 80 nm) plus a background (no particles) class. We trained a convolutional neural network, including its architecture optimization, and achieved 95% accurate results. We compared the performance of this network to an existing method based on line-by-line search and thresholding, demonstrating up to a twofold enhanced performance in particle classification. The network is extended by a supervisor layer that can reject up to 80% of the fooling images at the cost of rejecting only 10% of original data. The developed Python and PyTorch codes, as well as dataset, are available online.

https://doi.org/10.1364/AO.399894

1. INTRODUCTION

With the rapid growth of integrated circuits (ICs) fabrication in the semiconductor industry and, accordingly, the dramatic decrease in the size of the components, the next generation of chips becomes more complex and compact [1,2]. As a consequence, fast, sensitive, and reliable quality inspection of masks as well as wafers utilized in lithography machines is essential [3]. In order to maintain the high yield and quality in semiconductor manufacturing, particle contamination in the range of 1 μm down to 20 nm (in diameter) should be detected and, if possible, removed. Coherent Fourier scatterometry (CFS) has been suggested to fulfil the need for noninvasive and sensitive inspection of nanoparticles on surfaces [4–7].

With the help of CFS, one can obtain precise information about the condition of the unpatterned wafer, which is used for all types of devices, such as those with III-V materials, analogue, logic, and memory [8]. Inspection of wafers revealing the diameters, density, and positions of killer-nanoparticles is vital for the nanofabrication production environment because it is related directly to wafer cleaning and has an essential relation with fabrication yield [9]. Some traditional search and thresholding algorithms for datasets as generated by CFS (scattered maps) have been proved to be the appropriate schemes to estimate the size distribution of the contamination particles accurately [10–12].

Recently, there is growing interest in deep learning, which has demonstrated its feasibility to significantly improve optical microscopy, enhancing its spatial resolution over a large field of view and depth of field [13], analysis of medical images [14], analyzing TSOM images of nanostructures [15], detecting and localizing holographic features [16], and many other application areas in optics and physics [17,18]. Deep learning algorithms are part of a broader family of machine learning algorithms, which can be considered as a network consisting of multiple neural layers with the idea to progressively extract higher level features from the raw input, otherwise known as learning on the representation of the data [19]. Examples include the deep neural network (DNN) [20], recurrent neural network (RNN) [21,22], long short-term memory (LTSM) [23], and convolutional neural network (CNN) [24,25]. The feasibility of CNN has been demonstrated by wafer map defect pattern classification using simulated wafer maps (synthetic data) [26], relying on SEM images for classification of defects and contamination [27] and recently defining the chemical composition of particle defects on semiconductor wafers by merging the SEM image data with EDX spectral data as input [28].

Recently, 2*D* scattered maps generated by the CFS technique have been studied with line-by-line search algorithms resulting in histograms that rely on the features of characteristic electronic signals that are generated when a particle is detected. The cumbersome search routines are associated with the hyperparameters defined by the user for each specific input dataset, e.g., an expected amplitude and width of the characteristic signal, density, and number of points in a cluster of isolated particles, and the zeroing parameters for different iterations of the search [29]. There is still a need for discussion on minimization of user–algorithm interaction [30,31]. However, it is true that considering a growing amount of data and pressure to inspect data more quickly, the line-by-line analysis of the dataset can become computationally slow. The additional challenge is the precise categorization of killer-particles using automated contamination or defect classification. In other words, the confusion between the different sizes of the particles on the sample should be minimal. Moreover, while there are solutions to all of these problems, there is a cost associated with each of them. For instance, the application of neural networks requires re-training the network when the physical parameters change, which will consume a lot of resources and time. On the other hand, when pre-trained and deployed, the classification runs almost at no time, and virtually no *a priori* parameters or additional tuning of the network is required.

The application and feasibility of deep learning for the datasets of CFS have not been studied yet. In this paper, we propose a method to classify the scattered maps of isolated nanoparticles using CNN. We utilize calibrated samples of polystyrene latex (PSL) nanospheres, spin-coated on the silicon wafer, with diameters ranging from 40 to 80 nm to collect the training data. Polystyrene particles are standard for the calibration of surface inspection tools because they have well-characterized optical properties (low index of refraction, thus most challenging to detect) and a very tight monodisperse size distribution [32]. Furthermore, for the classification, we study the areas of the wafer where the nanoparticle is absent, contributing to the "background" class. We also target a novelty detection by looking at ways for the network to separate the "unknown" class from the input data, i.e., classes that have been unseen in the training. In order to do that, we rely on a simple baseline approach and also a more sophisticated approach of introducing the OpenMax layer [33]. We realized an experiment by adding noise to the scattered maps (degrading thus the SNR), fooling images, as well as some unfamiliar (reversal) images to the network. As one of the main goals for the CNN is to accurately discriminate among classes, we study samples that contain multiple classes of PSL particles with diameters of 40 and 50 nm, 50 and 60 nm, and 60 and 80 nm. The results show that our model can successfully discriminate among the proposed five classes with an accuracy up to 95%. By providing the samples that were unseen during training, our results for the first time highlight the importance of the novelty detection to capture the confusing inputs in a contamination detection problem. The results show that the proposed method has superior capabilities compared to classification with the traditional search algorithm [29]. A vital issue for future research is to merge the proposed classification CNN with the network for automatic object detection. This will allow to speed up in characterizing large amounts of data with the potential to nearly real-time inspection. The dataset and the codes used to generate the typical results of this paper are available online [34,35].

2. METHOD

The proposed CNN algorithm takes a set of "images" (signal intensity maps) as the network input and outputs the class labels [see Fig. 1(B)]. The discretization of the data is due to the sampling speed of the NI 5922 acquisition board and selection of the scanning step between lines. The scale in Fig. 1(A) (150 × 150) is given in pixels with each pixel corresponding to 2 nm in x direction and 4 nm in y direction. The intensity maps are captured from illuminating the sample with a blue diode laser (405 nm by Power Technology, model: IQ1A25) that is focused by a non-commercial objective designed for mastering CDs in optical data storage of numerical aperture NA = 0.9. The focused spot of ≈1 μm illuminates the sample containing isolated nanoparticles deposited on a wafer and mounted on a 3D piezo-electric stage whose position can be controlled with sub-nm precision (P-629.2CD by Physik Instrumente). The sample is scanned in a raster fashion [see Fig. 1(A)]. The total scattered and reflected fields from the nanoparticle and sample surface are collected at the balanced detector (BD) via the beam splitter (bi-cell silicon photodiode from Advanced Photonix).

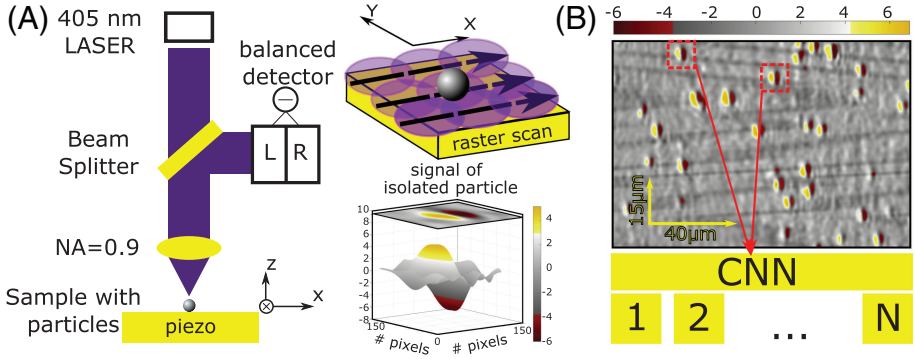

Fig. 1. (A) Differential detection principle: (left) schematic of setup, (right, top) raster scanning scheme, and (right, bottom) example of 2D map with the signal obtained when an isolated particle is recorded by the balanced detector. (B) Schematic process of CNN-based classification. The dotted red boxes indicate that the images are cut out. The inputs to the CNN are cut-out far-field maps containing the detection of nanoparticles; the output of the CNN is the label of the particle diameter 1, 2, … N.

For each scan position, the left and right sections of the detector are integrated and subtracted from each other, generating one photocurrent value per scan position, sequentially. In order to generate the 2D scan maps, the data points are arranged line per line, according to the raster scan pattern.

The raw data yield the detection of numerous isolated particles. The density of deposited particles has been chosen in order to have a considerable number of detected particles in a scan area of, e.g., $40 \times 15\ \mu m^2$ [Fig. 1(B)]. Other signals are also present, corresponding to clusters of particles, particle deposition residues, and possibly cross-contamination. Since directly using all particle-like detections from a sample is not always possible and would not give a high-quality dataset, we performed manual labeling. The type of particle signal present with the highest density inside the global scan area is representative of the nominal size. The bounding box is placed such that the particle signal is fully visible in the region of interest [dotted red boxes in Fig. 1(B)]. This square cut is centered about the position of the maximum amplitude of the differential signal. For the background class, the particle signal pattern (positive and negative amplitudes) of the particle should be absent. We have grouped the images into classes of 40, 50, 60, and 80 nm particles, and the "background" class that corresponds to the areas of the sample without particles [see Fig. 2(A)].

We created a class-balanced dataset (see Table 1) with roughly 260 images per class. The total amount of 1302 images is fed to the network, and we use the $60 - 20 - 20$ split for training, validation, and testing. Here we ensure that all three sets contain representative examples by randomly splitting data from each class into three parts and then merging to form the unbiased sets. We use the holdout method for validation, meaning that after each epoch, the validation dataset is passed through the network. When the training is complete, we show the test set to the network for a single time. The amount of images required for the training of the network, contrary to expectations, turns out to be relatively small, presumably due to the simple pattern of the particle signal. We did not apply any geometric transformation to the experimental data; thus, the input data contain diverse examples due only to the inherent experimental conditions.

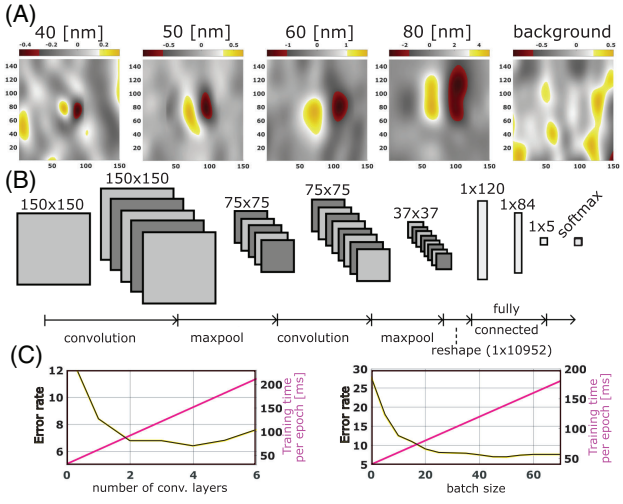

Fig. 2. (A) Examples of the five output classes. (B) The architecture consists primarily of convolutional layers capable of extracting relevant features of the input samples. Three fully connected layers at the end serve as a decision layer, mapping the automatically extracted features to the desired output class. (C) Error rate as a function of the number of convolution layers and batch sizes. From the plots, we see that two convolution layers and batch size of 15 are the optimal choice for the architecture, since they introduce a good balance of training time and low-enough error rate.

Table 1. Amount of Images Per Class (Original Dataset)

Class	40 nm	50 nm	60 nm	80 nm	Background	Total
# of images	254	253	276	272	247	1302

As shown in Fig. 2(B), we use the network architecture where no manual feature selection is necessary. The simple deep neural network is composed of repeated units of convolutional layers, whose number and sizes are chosen to have a balance between speed and low *error* (1–*Accuracy*) [Fig. 2(C)]. The final architecture includes an input size of 150×150 pixels and two convolution layers operating with filter (kernel) sizes of 5×5 pixels. The amount of filters in the first convolutional layer is five and in the second is eight; stride is one. In between and after the convolutional layers, we have inserted two max-pooling layers with a size of 2×2 pixels, effectively reducing the image resolution by a factor of two at each step. The purpose of these layers is to reduce computation for consecutive layers and to provide a form of translation invariance. All convolutional layers have rectified linear unit (ReLU) activation. Each ReLU in the network is followed with batch normalization [36]. The final max-pool layer is fed into three fully connected layers of sizes 120, 84, and five. Final layers are necessary to learn the relationship between the learned features and the sample classes, which in our case is five. Finally, the logits are converted to the probability scores by the SoftMax function. The network's output is used to compute the mean-square error between the true label and the predicted label, also known as the cross-entropy loss. We used the Adam optimization scheme with a global learning rate of 0.001 to minimize this loss function. The total number of weights in the network updated during the training process is 1,326,087. We built a Pytorch [37] implementation and moved it to the GPU (NVIDIA GTX1050 Ti) calculation with tensors. More information about the implementation can be found in Refs. [34,35].

3. RESULTS

A. Closed Set Classification

The best model has to be selected based on the accuracy metric calculated on the test data. For the closed set of five classes, as according to Table 1, we found that after approximately 12 epochs (12 times through all the training examples), the loss no longer decreased significantly (Fig. 3A). The top performing network ($Accuracy = 95\%$) was stored to be used in further tasks.

In order to see which classes the network struggles to distinguish and to what degree, we built the confusion matrix [Fig. 3(B)]. The horizontal axis represents the particle classes predicted by our model, and the vertical axis represents the true input image labels. For example, the 80 nm row (fourth row

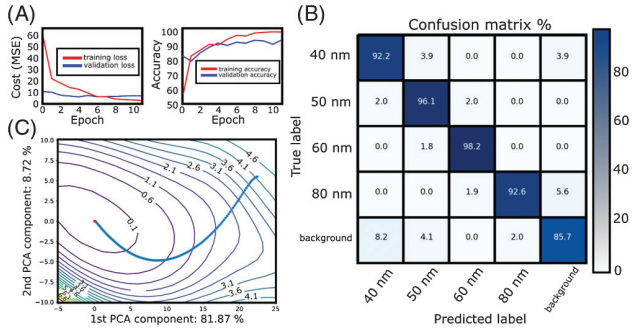

Fig. 3. (A) Training and validation loss (left) and accuracy (right) are evaluated across different numbers of epochs based on the optimal parameters [Fig. 2(B)] architecture of the CNN model. (B) Accuracy for the test set in the confusion matrix. (C) 2D visualization of the loss surface of the CNN model with the projected learning trajectories using normalized PCA direction (batch size of 20, Adam optimizer, and 15 epochs of training).

in the matrix) indicates that 92.6% of the images labeled with 80 nm are correctly predicted as 80 nm; 1.9% are incorrectly predicted as 60 nm; 5.6% are predicted as background. Our experiment shows that 50 nm is often confused with 40 nm. The reason is that there is a small difference between the scattering cross sections generated by these two particle sizes (the scattering varies by the sixth power of the diameter [32]). It is clearly visible that most misclassification involves the background class. Finally, we built the landscape [38], where we demonstrate the convergence to minima, as our learning procedure follows the loss in a gradual manner. The projected learning trajectory is estimated using normalized principal component analysis (PCA) directions. The squared nature of the loss function leads to a mostly convex loss landscape [Fig. 3(C)].

B. Comparison with Thresholding Classification Method

We compared the performance of our CNN classifier, pre-trained on the five classes (Section 3.A) with a method that has been recently implemented by some of the authors of this paper [29]. We did this on new test sets of separately 40, 50, 60, and 80 nm particles, with roughly 40 cut-out images per class. The thresholding classification method can be summarized as:

- Search line by line for signals that have characteristic shape (positive-negative pulses) and that are close to the expected amplitude and time-width of the particle in question;
- Use the density-based spatial clustering of applications with noise (DBSCAN) algorithm to define the group of signals attributed to a single scatterer and return the estimate of the time-width from centroid. By group of signals, we mean that the signal should repeat itself at the same x position in a few consecutive scan lines (in y direction);
- Use a calibration curve based on the time-width of the signal as a function of the particle size to return a class label for the particle.

This method operates on the reference positions of the cut-out images from the corresponding raw scan maps. We keep the number of output classes equal to five to provide a fair comparison; hence, in the thresholding method, instead of the background, the class of 100 nm particles is present.

Table 2. Comparison of Accuracy Per Class between the Proposed CNN and Method Based on Thresholding and Search

	40 nm (35 Images)	50 nm (37 Images)	60 nm (37 Images)	80 nm (46 Images)
Thresholding	0.37	0.43	0.63	0.82
CNN	0.97	0.94	1	1

In Table 2, we present the classification performance of thresholding and CNN approaches on the four test sets. From the results, we can see that the classifier based on the neural network achieves better performance as compared to the classical search routine. Both approaches perform very accurate on the data of 80 nm particle class, but when reducing the size of a particle, the accuracy drops much faster in the case of the thresholding method.

It is critical to note that both approaches can consider the 2D local information inherent to our measured data. In the case of the thresholding approach, positive-negative signals that are present in consecutive scan lines at the same x positions are clustered and considered as a single particle (see Fig. 1). In the case of CNN, the convolution filter can extract the spatially connected information by walking over the image. It is unlikely thus that improved classification accuracy is due to the 2D nature of convolutional kernels. The essential difference is the ability of CNN to extract the higher-level representation by cascading the filters and learning on these representations. On the contrary, classification based on the calibration curve (signal feature as a function of particle diameter) always relies on the representation of the data that are manually engineered.

To address further the classification tendencies as they appear in methods under comparison, we demonstrate the confusion among the classes. The clustering method performed a lot worse, where the overall accuracy was 56%, with relatively accurate results for the 60 and 80 nm classes yet with a lot of confusion on the 40 and 50 nm classes. Evidently, in both approaches, the confusion between the neighboring classes is present (see Fig. 4). For the thresholding method, it is clear that the steeper the calibration curve becomes, the less confusion is present. Inherently, this reference method relies on the time-width, which has shown to be a sensitive parameter for the case of particles $\geq$100 nm [39]. We point out that classification of the single particle image takes 0.89 [s] for the case of the thresholding algorithm and 0.02 [s] for the pre-trained network in this paper. Generally speaking, other features can be selected in order to allow for better discrimination, which addresses the topic of feature engineering; however, this is beyond the scope of this paper. At the same time, not so deep CNN is enough to pick up on patterns in many different features of an input. Features extracted by the network extend far beyond those that make sense to the human eye, such as maxima, minima, or the pattern size. A CNN trained to recognize particles might find other features, such as patches of color, topography, or background, which can become even stronger predictors.

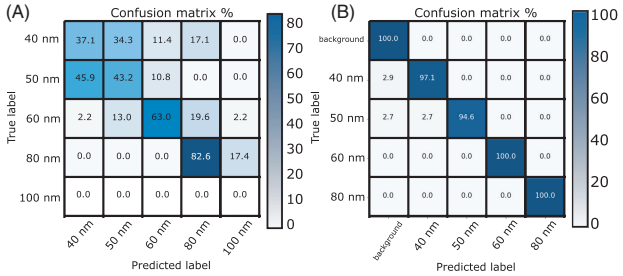

Fig. 4. Confusion matrices comparing classification tendencies between predicted and true labels by (A) thresholding and (B) CNN approach. Squares are colored based on the value of the cell, with darker colors indicating more matches. Values along the diagonal of each confusion matrix represent the images classified correctly, while values in off-diagonal regions represent blurring between types of classes.

C. Towards Multi-Class Open-Set Classification

Making alterations to the regular input data even in the form of tiny changes that are typically invisible to humans can mislead the best neural networks. These problems are not easy to solve because CNN's are fundamentally fragile. As shown in the previous two sections, the accuracy in classification is very high, but this is possible until networks are taken into unfamiliar territory where they can break in unpredictable ways. To bring a spotlight on the problem of confusion by the so-called "adversarial examples," [40] scientists have evolved images that look like an abstract pattern but that the DNNs see as familiar objects [41,42]. In the context of CNN applied for the classification of the different particle size contamination, we should envision that distorted measurement data or other types of untrained particles could also be spotted by the proposed network.

The output layer of the original architecture in Fig. 2(B) is a SoftMax layer that contains the vector of probabilities

$$P(y=j|\mathbf{x})=\frac{\exp(\mathbf{v}_j(x))}{\sum_{i=1}^{N}\exp(\mathbf{v}_i(x))}, \tag{1}$$

where x is the sample image, and $\mathbf{v}(x)$ is the corresponding activation vector. The number of classes is $j=1,\ldots,N$, and i is the index that goes over the classes. Due to the summation in the denominator, the probabilities are normalized and sum up to one. We want to build a $(N+1)$-classifier $f(x)$ with the classes $C=\{d_1, d_2, \ldots, b, \text{rejection}\}$. The most straightforward approach of a novelty detection is to introduce the baseline value for the scores of the SoftMax layer. If the probability of the output classes is not high enough, the input image is assigned with the unknown label. The novelty score NS is defined as

$$\text{NS}=1-\max(P(y=j|\mathbf{x})). \tag{2}$$

The procedure of computing the OpenMax probabilities includes four steps:

1. For each class $C=[c_{j,\_,N}]$, the mean activation vector is computed $\text{MAV}=[\mu_{j,\_,N}]$, where $\mu_j=\text{mean}(v_j(x_{i,j}))$, and $x_{i,j}$ represents the correctly classified sample.
2. Per class, fit the Weibull model with parameters $p_{c_j}=(t_{c_j}, \lambda_{c_j}, k_{c_j})$ to the distance between the input sample and the mean of the set of η number of outlier examples of class j. t_{c_j} is used for shifting the data, λ_{c_j} and k_{c_j} are, respectively, the scale and shape parameters derived from the training data of the class c_j and control the cumulative density function (CDF). For more details on Weibull distribution and extreme value theory, see Ref. [43].
3. Estimate the Weibull CDF probability on the distance between sample x_i and the known class's mean activation vector: $\text{MAV}[\mu_{j,\_,N}]$ defined as $w(\mathbf{x})$. Recalibrate the activation vector by $\hat{v}(x)=\mathbf{v}(\mathbf{x})\circ w(\mathbf{x})$. To allow the novelty detection, augment output to $N+1$ classes by $\hat{v}_{N+1}(x)=\sum_i \hat{v}_i(x)(1-w_i(x))$.
4. To support explicit rejection, the pseudo-probability of an unknown class is estimated from the known class's activation scores:

$$\hat{P}(y=j|\mathbf{x})=\frac{\exp(\hat{\mathbf{v}}_j(x))}{\sum_{i=0}^{N}\exp(\hat{\mathbf{v}}_i(x))}, \quad j=1,\ldots,N+1. \tag{3}$$

The third way of dealing with the unknown input is similar to OpenMax; however, it is much simpler and essentially relies on the MAV:

1. Calculate the MAV for the correct classifications of each class.
2. For each image x in the train and validation sets, obtain the activation vector $v(x)$ and predicted class $c(x)$. Then, calculate the distance to the MAV with $d=||v(x)-\text{MAV}_{c(x)}||$. Save values of d separately for correct and incorrect classifications.
3. For each image in the test set, calculate d in the same way, and if it is above some threshold, reject the classification (thus classifying it as unknown).

Thus, we utilize and compare three approaches of baseline, OpenMax, and distance to MAV in order to catch open-set examples, each time showing the unseen images to the network without additional training. We introduce an input sample with high noise, where Gaussian and $1/f$ noises were added to every image. For instance, samples that correspond to 80 nm [see Fig. 5(A)] were modified such that the SNR decreased by -9.7 dB (1.5 times), fooling images of an elephant from the *Animal*–10 dataset [44], and finally, the mirrored image along x axis of the 2D scattered maps with detected particles, which is representative of the image of a defect such as a small pit.

It is not rare that studies on detecting fooling/adversarial images have a narrow focus on optimizing the output or penultimate layer of the network, such that the probability for an unknown image is low. This is something of a pitfall because it is possible to optimize the score of rejecting the unknown class with the cost of losing a significant number of correct images. With this intention, when making the comparison, we apply the network with different output layers for both the original and fooling dataset. In different scenarios, we observe similar behavior. The vast majority of the original images lie in the shallow uncertainty region, which is an indication of a highly accurate network. However, the incorrect ones span nearly the same range as the correct ones, meaning we cannot completely

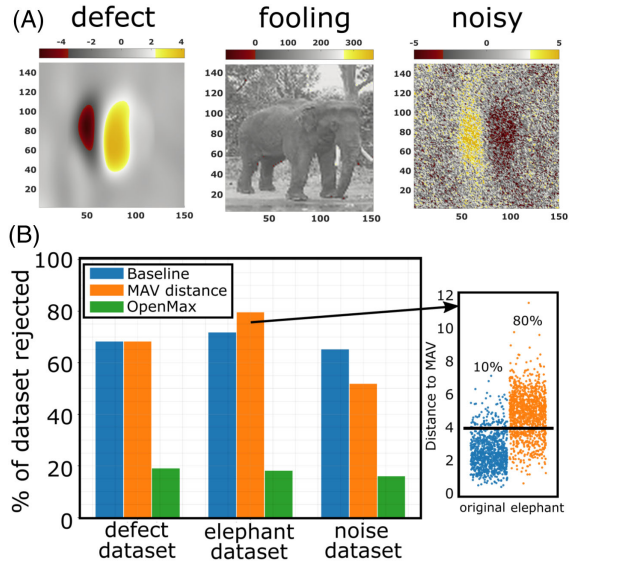

Fig. 5. (A) Three types of open-set examples: image of defect, fooling image of the elephant, and noisy particle image. (B) Summary of comparison among three unknown detection methods as applied to the open-set examples, ≈1090 images per type. Inset: example of the best performing case, where the "distance to MAV" method is applied to the fooling set of elephants. Blue points represent the complete original set passed through the network, and the orange points represent the fooling set. If the threshold (in red) is set such that only 10% of the "good" images are dropped, then the same network can capture 80% of the unknown images.

get rid of incorrect classifications by thresholding uncertainty. No matter what threshold we would set, we would always have some incorrect classification. Further, we chose this threshold such that we reject approximately 10% of standard data. This is an arbitrary value, since the acceptable maximum rejection of standard data would depend on the application, and on how frequently images appear in the data that should be rejected. As a result, we compare the amount of rejection in open-set examples by the three supervisor approaches in Fig. 5(B). In particular (inset figure), for the fooling dataset, the best performing algorithm is the one based on the distance to MAV, where we can see how it is possible to separate 80% of the fooling images. We pass the entire original dataset (blue points) and the whole fooling elephant dataset (orange points) through a trained network and set the threshold based on the information from the MAVs. The additional information gained from using the entire vector of outputs rather than just the maximum (novelty score) helps with rejecting unknown inputs there. For the defect set, it performs no better than the baseline approach, and on the noise set, it performs significantly worse. Finally, the OpenMax gave us poor results on all sets.

4. DISCUSSION

The Weibull model is the core of the OpenMax approach. As an essential part of the algorithm, the method selects the m highest activations per activation vector. Our data have only five classes, meaning our activation vectors have only five entries. There is not much selection possible in this case. The original paper of OpenMax [33] visualizes activation vectors for a 450 class system and provides some intuition about where the information resides that is used in OpenMax. It is thus clear that, with a low amount of classes, the CDF distribution would be very discrete, and a lot less information could be gained from it.

Instead of having the supervisor in the network, such as baseline or MAV, one can include the fooling images as a part of the training data, in particular, to expose the network to problematic cases regularly. In this form, the output layer would explicitly contain the desired class. However, training a network to withstand one kind of "unknown" images could weaken it against others [45].

In case the network performs poorly, inspecting the images that contribute to the off-diagonal elements of the confusion matrix allows us to study the data better and, if required, remove inputs from the dataset to get to higher accuracy. There are also existing approaches where the subsets of outlier images are removed from the training or test data. These are the interactive learning-based methods for curating datasets using user-defined criteria [46,47].

To deploy the network in the in-line fab inspection scenario, more output classes should be provided, approaching the real-world situation with a variety of particle sizes on the surface. The synthetic data could replace the types of particles not available experimentally for calibration.

5. CONCLUSION

In this paper, we have applied CNN to 2D maps obtained using CFS for nanoparticle detection and classification. We trained a CNN to recognize four classes of nanoparticles and a surface background class. Based on a total of 1302 experimental images rather than synthetic data, with a simple CNN with two convolutional layers and batch normalization, we demonstrated 95% accuracy on the test data. The proposed approach outperforms the existing algorithm for analysis of scattered maps, which is based on thresholding and search [29]. For relatively small particles, with diameters (classes) of 40 and 50 nm, the accuracy has been improved by a factor of two. Also, when studying the amount of misclassification presented by both methods, we see that the CNN can cope better in separating nanoparticles that produce very similar scattering cross sections (such as particles with diameters of 40 and 50 nm). The demonstrated increase in accuracy and minimized confusion could be attributed to the fact that CNN automatically extracts the features from the proposed data, while the search approach looks only at manually engineered features. Further, we experimented with selecting the best approach to capture the images unseen during the training of the network. The output layer of the proposed CNN can be augmented with a method based on MAVs, OpenMAX, or simple baseline approach. Depending on the need, the threshold value for the uncertainty of the unknown images can be introduced as we show experimentally, e.g., 80% of the fooling images of an elephant can be neglected at the cost of dropping only 10% of the particle-type dataset. We believe that the proposed CNN is an essential addition to nanoparticle detection and classification.

Funding. High Tech Systems and Materials Research Program, Nederlandse Organisatie voor Wetenschappelijk Onderzoek, Applied and Technical Sciences Division (TTW) (Project no. 14660).

Acknowledgment. We gratefully acknowledge the help provided by TNO in the fabrication of samples. Dmytro Kolenov acknowledges the High Tech Systems and Materials Research Program with Project no. 14660, financed by the Netherlands Organisation for Scientific Research (NWO), Applied and Technical Sciences division (TTW), for funding this research. Justine Le Cam acknowledges Erasmus+ organization for granting her a scholarship during her internship at TU Delft.

Disclosures. The authors declare no conflicts of interest.

REFERENCES

1. J. Carballo, W. J. Chan, P. A. Gargini, A. B. Kahng, and S. Nath, "International technology roadmap for semiconductors 2.0," Executive Report, 2015, http://tinyurl.com/yxrwy5oo.
2. J. Ahopelto, G. Ardila, L. Baldi, F. Balestra, D. Belot, G. Fagas, S. D. Gendt, D. Demarchi, M. Fernandez-Bolaños, D. Holden, A. Ionescu, G. Meneghesso, A. Mocuta, M. Pfeffer, R. Popp, E. Sangiorgi, and C. S. Torres, "Nanoelectronics roadmap for Europe: from nanodevices and innovative materials to system integration," Solid State Electron. **155**, 7–19 (2019) (Selected Papers from the Future Trends in Microelectronics (FTM-2018) Workshop).
3. W. Broadbent, Jr., S. Watson, P.-C. Chiang, R.-F. Shi, J.-R. Wang, and P. Lim, "1X HP EUV reticle inspection with a 193nm inspection system," Proc. SPIE **10451**, 149–157 (2018).
4. D. Kolenov, R. C. Horsten, and S. F. Pereira, "Heterodyne detection system for nanoparticle detection using coherent Fourier scatterometry," Proc. SPIE **11056**, 336–342 (2019).
5. O. E. Gawhary and S. J. Petra, "Method and apparatus for determining structure parameters of microstructures," U.S. Patent No. 9,175,951 (November 3, 2015).
6. S. Roy, S. F. Pereira, H. P. Urbach, X. Wei, and O. El Gawhary, "Exploiting evanescent-wave amplification for subwavelength low-contrast particle detection," Phys. Rev. A **96**, 013814 (2017).
7. S. Roy, A. C. Assafrao, S. F. Pereira, and H. P. Urbach, "Coherent Fourier scatterometry for detection of nanometer-sized particles on a planar substrate surface," Opt. Express **22**, 13250–13262 (2014).
8. M. Lapedus, "Inspecting unpatterned wafers," 2018, https://tinyurl.com/qlrrqsj.
9. W. Kern, "Chapter 1—overview and evolution of silicon wafer cleaning technology," in *Handbook of Silicon Wafer Cleaning Technology*, K. A. Reinhardt and W. Kern, eds. (William Andrew Publishing, 2018), pp. 3–85.
10. L. Dou, D. Kesler, W. Bruno, C. Monjak, and J. Hunt, "One step automated unpatterned wafer defect detection and classification," AIP Conf. Proc. **449**, 824–828 (1998).
11. T. Hattori, A. Okamoto, and H. Kuniyasu, "Challenges of finer particle detection on unpatterned silicon wafers," AIP Conf. Proc. **683**, 271–277 (2003).
12. K. A. Reinhardt and W. Kern, "Chapter 12—detection and measurement of particulate contaminants," in *Handbook of Silicon Wafer Cleaning Technology*, 3rd ed. (William Andrew Publishing, 2018), pp. 659–699.
13. Y. Rivenson, Z. Göröcs, H. Günaydin, Y. Zhang, H. Wang, and A. Ozcan, "Deep learning microscopy," Optica **4**, 1437–1443 (2017).
14. S. Zhou, H. Greenspan, and D. Shen, *Deep Learning for Medical Image Analysis* (Elsevier, 2017).
15. Y. Qu, J. Hao, and R. Peng, "Machine-learning models for analyzing TSOM images of nanostructures," Opt. Express **27**, 33978–33998 (2019).
16. M. D. Hannel, A. Abdulali, M. O'Brien, and D. G. Grier, "Machine-learning techniques for fast and accurate feature localization in holograms of colloidal particles," Opt. Express **26**, 15221–15231 (2018).
17. G. Carleo, I. Cirac, K. Cranmer, L. Daudet, M. Schuld, N. Tishby, L. Vogt-Maranto, and L. Zdeborová, "Machine learning and the physical sciences," Rev. Mod. Phys. **91**, 045002 (2019).
18. K. Kuppala, S. Banda, and T. R. Barige, "An overview of deep learning methods for image registration with focus on feature-based approaches," Int. J. Image Data Fusion **0**, 1–23 (2020).
19. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature **521**, 436–444 (2015).
20. G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," Science **313**, 504–507 (2006).
21. M. I. Jordan, "Attractor dynamics and parallelism in a connectionist sequential machine," in *Eighth Annual Conference of the Cognitive Science Society* (Erlbaum, 1986), pp. 531–546.
22. B. A. Pearlmutter, "Learning state space trajectories in recurrent neural networks," Neural Comput. **1**, 263–269 (1989).
23. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput. **9**, 1735–1780 (1997).
24. K. Fukushima, "Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," Biol. Cybern. **36**, 193–202 (1980).
25. Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," Neural Comput. **1**, 541–551 (1989).
26. T. Nakazawa and D. Kulkarni, "Wafer map defect pattern classification and image retrieval using convolutional neural network," IEEE Trans. Semicond. Manuf. **31**, 309–314 (2018).
27. S. Monno, Y. Kamada, H. Miwa, K. Ashida, and T. Kaneko, "Detection of defects on SiC substrate by SEM and classification using deep learning," in *Advances in Intelligent Networking and Collaborative Systems*, F. Xhafa, L. Barolli, and M. Greguš, eds. (Springer, 2019), pp. 47–58.
28. J. OrLeary, K. Sawlani, and A. Mesbah, "Deep learning for classification of the chemical composition of particle defects on semiconductor wafers," IEEE Trans. Semicond. Manuf. **33**, 72–85 (2020).
29. D. Kolenov and S. F. Pereira, "Machine learning techniques applied for the detection of nanoparticles on surfaces using coherent Fourier scatterometry," Opt. Express **28**, 19163–19186 (2020).
30. F. Hutter, J. Lücke, and L. Schmidt-Thieme, "Beyond manual tuning of hyperparameters," KI—Kunstl. Intell. **29**, 329–337 (2015).
31. Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "Completely automated CNN architecture design based on blocks," IEEE Trans. Neural Netw. Learning Syst. **31**, 1242–1254 (2020).
32. H. R. Huff, R. K. Goodall, E. Williams, K. S. Woo, B. Y. Liu, T. Warner, D. Hirleman, K. Gildersleeve, W. M. Bullis, B. W. Scheer, and J. Stover, "Measurement of silicon particles by laser surface scanning and angle–resolved light scattering," Electrochem. Soc. **144**, 243–250 (1997).
33. A. Bendale and T. E. Boult, "Towards open set deep networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 1563–1572.
34. D. Davidse and D. Kolenov, "CNN for nanoparticle," 2020, https://github.com/ddavidse/CNN_for_nanoparticle.
35. D. Kolenov and D. Davidse, "Training and test data for the preparation of the article: convolutional neural network applied for nanoparticle classification using coherent scaterometry data," 2020, https://doi.org/10.4121/uuid:516ab2fa-4c47-42f8-b614-5e283889b218.
36. S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," in *Proc. of the 32nd International Conference on International Conference on Machine Learning*, 2015, Vol. **37**, 448–456.
37. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: an imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A.

Beygelzimer, F. d' Alché-Buc, E. Fox, and R. Garnett, eds. (Curran Associates, Inc., 2019), Vol. **32**, pp. 8024–8035.
38. H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," in *Neural Information Processing Systems* (2018), pp. 6389–6399.
39. S. Roy, "Sub-wavelength metrology using coherent Fourier scatterometry," Ph.D. Thesis (TU Delft, 2016).
40. C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *International Conference on Learning Representations* (2014).
41. J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," IEEE Transactions on Evolutionary Computation, **23**, 8601309 (2019).
42. D. Heaven, "Why deep-learning ais are so easy to fool," Nature **574**, 163–166 (2019).
43. W. J. Scheirer, A. Rocha, R. J. Micheals, and T. E. Boult, "Meta-recognition: the theory and practice of recognition score analysis," IEEE Trans. Pattern Anal. Mach. Intell. **33**, 1689–1695 (2011).
44. H. Song, M. Kim, and J.-G. Lee, "SELFIE: refurbishing unclean samples for robust deep learning," in *International Conference on Machine Learning (ICML)* (2019), pp. 5907–5915.
45. D. Kang, Y. Sun, D. Hendrycks, T. Brown, and J. Steinhardt, "Testing robustness against unforeseen adversaries," arXiv:1908.08016 (2019).
46. R. Tous, O. Wust, M. Gomez, J. Poveda, M. Elena, J. Torres, M. Makni, and E. Ayguadé, "User-generated content curation with deep convolutional neural networks," in *IEEE International Conference on Big Data (Big Data)* (2016), pp. 2535–2540.
47. W. Ye, Y. Dong, and P. Peers, "Interactive curation of datasets for training and refining generative models," Computer Graph. Forum **38**, 369–380 (2019).