Journal of
Micro/Nanolithography,
MEMS, and MOEMS

Nanolithography.SPIEDigitalLibrary.org

Deep learning-based detection, classification, and localization of defects in semiconductor processes

Dhruv V. Patel
Ravi Bonam
Assad A. Oberai

SPIE.

Dhruv V. Patel, Ravi Bonam, Assad A. Oberai, "Deep learning-based detection, classification, and localization of defects in semiconductor processes," *J. Micro/Nanolith. MEMS MOEMS* **19**(2), 024801 (2020), doi: 10.1117/1.JMM.19.2.024801

Deep learning-based detection, classification, and localization of defects in semiconductor processes

Dhruv V. Patel,[a] Ravi Bonam,[b,*] and Assad A. Oberai[a,*]
[a]University of Southern California, Department of Aerospace and Mechanical Engineering,
Los Angeles, California, United States
[b]IBM Research Division, Albany, New York, United States

Abstract. Defects in semiconductor processes can limit yield, increase overall production cost, and also lead to time-dependent critical component failures. Current state-of-the-art optical and electron beam (EB) inspection systems rely on rule-based techniques for defect detection and classification, which are usually rigid in their comparative processes. This rigidity limits overall capability and increases relative engineering time to classify nuisance defects. This is further challenged due to shrinkage of pattern dimensions for advanced nodes. We propose a deep learning-based workflow that circumvents these challenges and enables accurate defect detection, classification, and localization in a unified framework. In particular, we train convolutional neural network-based models using high-resolution EB images of wafers patterned with various types of intentional defects and achieve robust defect detection and classification performance. Furthermore, we generate class activation maps to demonstrate defect localization capability of the model "without" explicitly training it with defect location information. To understand the underlying decision-making process of these deep models, we analyze the learned filters in pixel space and Fourier space and interpret the various operations at different layers. We achieve high sensitivity (97%) and specificity (100%) along with rapid and accurate defect localization. We also test performance of the proposed workflow on images from two distinct patterns and find that in order to retain high accuracy a modest level of retraining is necessary. © *2020 Society of Photo-Optical Instrumentation Engineers (SPIE)* [DOI: 10.1117/1.JMM.19.2.024801]

Keywords: lithography; photomask; defects; programmed defects; neural networks; convolutional neural network; generative adversarial network; electron beam images.

Paper 20001 received Jan. 14, 2020; accepted for publication Mar. 30, 2020; published online Apr. 20, 2020.

1 Introduction

Any semiconductor process used in high-volume manufacturing has a certain defectivity level specification that is required to be monitored to maintain the yield of the final product. Significant time and effort are spent in optimizing processes to maintain the desired level of defectivity, which is monitored through routine inspections that involve defect detection and classification. Automated defect classification algorithms use tags provided by engineers to perform classification and are relatively rigid when performing comparison and classification. In general, it is difficult to attain high values of sensitivity (number of defective patterns that are detected correctly out of total number of defective patterns) in defect detection and classification. The requirement of high sensitivity often leads to an increase in the number of defect-free patterns that are incorrectly classified as defects. This requires critical reviews to be performed by engineers to sanitize the inspections setups, which can consume substantial time and resources.

With the advent of faster graphics processing units (GPUs), better techniques for acquiring and storing data, and the development of innovative algorithms, deep learning techniques have seen an exponential growth in solving image-based detection and classification problems. Since most inspection and review tools in semiconductor manufacturing/processing produce grayscale images, this raises the possibility of using deep neural networks for these tasks to perform defect

*Address all correspondence to Ravi Bonam, E-mail: rkbonam@us.ibm.com; Assad A. Oberai, E-mail: aoberai@usc.edu

1932-5150/2020/$28.00 © 2020 SPIE

detection, classification, and localization with higher sensitivity and specificity. Further, they may be applicable to different patterns and technologies. With this as motivation, in this work, we apply convolutional neural network (CNN)-based models on images collected from electron beam (EB) inspection and quantify its performance in detecting, classifying, and locating defects, as well as develop an understanding on how it performs classification tasks.

1.1 *Defects in Lithography*

Defects in semiconductor lithography can range from being critical failures to yield limiters. Semiconductor manufacturers have stringent requirements to capture all defect types, and engineers spend significant time to review and debug inspection data. Figure 1 shows examples of defects commonly encountered in semiconductor processes. Figures 1(a) and 1(b) show line/space patterns with bridge type and partial line missing types defects, respectively. Figures 1(c) and 1(d) show pattern missing and bridge type defects for contact patterns, respectively.[1]

To assess sensitivity of defect inspection tools (minimum defect detection capability of an inspection tool), defects are intentionally placed in test structures and patterned on wafer. These wafers are inspected by various tools and capture efficiencies are monitored. These defects can be of various types and sizes, and may be located anywhere within a pattern. The printability of defects on wafer is a function of patterning process and detection capability.

Apart from detection of defects, classification is a major challenge as it can consume a lot of engineering resources and time.[1] Figure 2 shows examples of multiple types of intentional defects that were used as part of tool sensitivity assessment. Figure 2(a) shows multiple defect types for line space pattern and Fig. 2(b) shows defect types for contact patterns. These defect types are commonly found as a result of defects on photomasks or patterning processes and emphasize the challenge in classification. Classification is an important part in a defect inspection process and can be used to identify the root cause of defects and assess any modulation based on process fixes.

1.2 *Deep Learning and Convolutional Neural Networks*

The advancement of semiconductor technology and ever-increasing prevalence of personalized smart devices have enabled the development of powerful computational resources and collection of a large amount of data. These two ingredients have enabled powerful machine learning (ML)

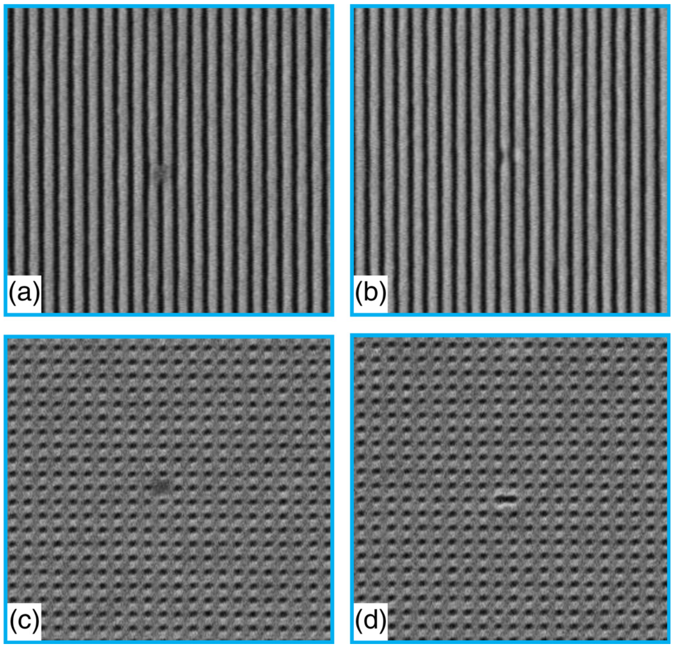


Fig. 1 Typical defects.

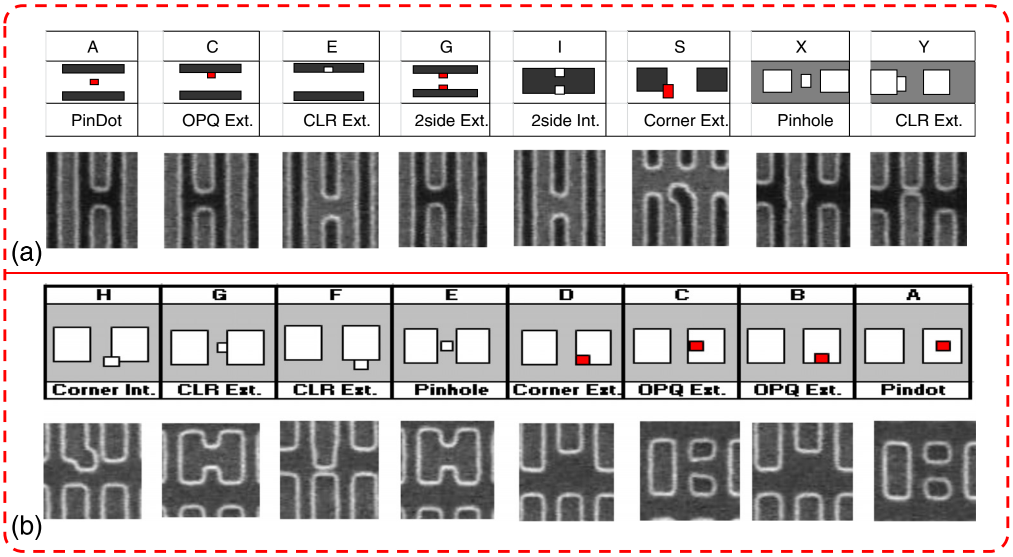


Fig. 2 Different types of intentionally introduced defects. (a) Line/space (1-D) patterns. A: Pindot—unwanted absorber in a clear area, C: opaque extrusion—absorber extrusion, E: clear extrusion, G: two-side clear extrusion, I: two-side clear intrusion, S: corner extrusion, X: pinhole—unwanted clear area in absorber, Y: clear extrusion—line end. (b) Line/Space (1-D) patterns. A: pindot—unwanted absorber in a clear area, C: opaque extrusion—absorber extrusion, E: clear extrusion, G: two-side clear extrusion, I: two-side clear intrusion, S: corner extrusion, X: pinhole—unwanted clear are in absorber, Y: clear extrusion—line end.

algorithms with applications ranging from recommendation systems and fraud detection in e-commerce websites to web searches and content filtering on social networks. Recently, deep learning has made substantial progress in a variety of different tasks, such as detecting objects in an image, translating a sentence from one language to other, generating images from sentences, and developing self-learning robots.[2–5] The final objective in these seemingly diverse applications is achieved by the same underlying principle, which is to take raw data as input, extract useful features directly from this data, and use these features to accomplish the given task.

Every deep learning model encodes this principle by performing a sequence of nonlinear transformations of input data. Depending on the type and the order of transformations, one can classify deep learning models into different categories. CNN is a type of deep learning model that is composed of sequence of convolution operations followed by nonlinear transformations.[6] It is particularly useful for data with hierarchical structure, such as images, as it allows the extraction of important local features in a stratified fashion. In a CNN, the convolution layers are usually followed by fully connected layers, which manipulate the extracted features in high-dimensional space, which is then passed to final output layer, which performs the output task (classification, regression, etc.).

1.3 *Semiconductor Defect Inspection Using Deep Learning*

In recent years, deep learning has shown remarkable capability in automatically extracting task-specific features from data and performing different tasks, such as image classification, object localization, semantic segmentation, etc.[6–8] Indeed, deep learning algorithms have now surpassed human performance on many benchmark datasets.[6] This has made them a tool of choice for various semiconductor defect inspection tasks. Many recent works[9–12] have explored the use of deep learning for hotspot detection. This includes the use of CNN-based models[9] and their modification to tackle data imbalance to achieve high accuracy and lower nuisance rate.[10,12]

In another line of work, researchers have explored the use of deep learning algorithms in defect pattern recognition (analysis of the distribution of defect patterns on the whole wafer), which provides valuable information about the root cause of defects and helps improve the overall yield and reliability. This includes the use of synthetic wafer map images to train a CNN for classification of defect pattern and image retrieval task[13] and the use of clustering-based algorithms for simultaneous detection of outliers and recognition of defect patterns.[14] More recent works have explored the use of stacked convolution sparse denoising autoencoder for robust defect pattern recognition on wafer map[15] and the use of generative adversarial network (GAN) to tackle the imbalanced data problem in defective pattern recognition task.[16]

In the above-mentioned applications, the critical features (hotspot or defect patterns) are at a relatively larger scale (at wafer level). However, there are many instances when the defects are at a small scale (same scale as the technology nodes—few nanometers). In these cases, the current state-of-the-art optical and EB inspection tools, which rely on grayscale differences between the background pattern and the defect pattern, require careful manual tuning of many process parameters, which is a time-consuming and expensive process. Furthermore, it is observed that optimal value of these parameters depends on defect types.[17] Moreover, the capture efficiency of this manual technique is found to be directly proportional to the nuisance rate.[17] The work presented in this paper tackles the defect detection and classification problem.

In our earlier work,[18] we proposed a deep learning-based model that could be trained directly using high-resolution EB images and eliminated the need of manual parameter tuning and accuracy-nuisance rate dependence. In this work, we build on that and investigate the performance of two CNN models on more challenging tasks. First, we consider the use of raw EB images that display more variability, but do not require any preprocessing and thus can be seamlessly integrated with existing automatic defect classification systems. Second, we consider a situation where a model trained on images of a given pattern is used to perform classification on images with a different pattern with access to a small number of training images from the new pattern. This situation is quite common in applications, such as self-driving cars,[19] medical imaging,[20] and robotics,[21] where real data are quite scarce and/or expensive, and some form of transfer learning or domain adaptation strategy is adopted.

The rest of this paper is organized as follows. Section 2 is focused on the method employed in this study and it describes both the experimental setup used to generate the training data and computational setup—CNN architecture and training details. Here, we consider two different CNN architectures that differ in the final few layers. One uses fully connected layers and the other uses the global average pooling (GAP). Section 3 is focused on results and analysis. Specifically, we quantify the accuracy of CNN-based models in classifying different types of defects and analyze the activation maps and convolution filters of each layers to better understand the learning process. We also generate the heatmap showing class activation maps (CAMs) for each class. Section 4 demonstrates the performance of two different CNN models described in Sec. 2 on more challenging image detection tasks. We end with concluding remarks in Sec. 5.

2 Methods

2.1 *Generation of Data*

In the first study, a line space pattern with 30 different types and sizes of intentionally placed defects was fabricated. Figure 2(a) shows sample defect types used as part of this study. These defect types are commonly found in photomasks or patterning processes. The pattern with these intentionally placed defects was transferred to wafer, which was imaged using EB inspection system. In the second study, a square contact pattern with 30 different types and sizes of synthetic defects was considered. This dataset is described in more detail in Sec. 4.2.

Figure 3 shows the data generation workflow, which begins with the design with different intentional defects marked on it. This is used to pattern the wafer, which is imaged using an EB inspection setup. A grid is placed on this “EB image” to help create a small subimage of appropriate size, which is then used to train the CNN. Each subimage is then appropriately labeled.

In this study, images are categorized into three different classes. As shown in Fig. 4, all the images containing defects that cover a single line (open, short, etc.) are placed within the single line break defect class (in the figure a brown-rectangle is drawn around the defect for the reader to locate it); whereas all the images that contain defects spanning multiple line instances or

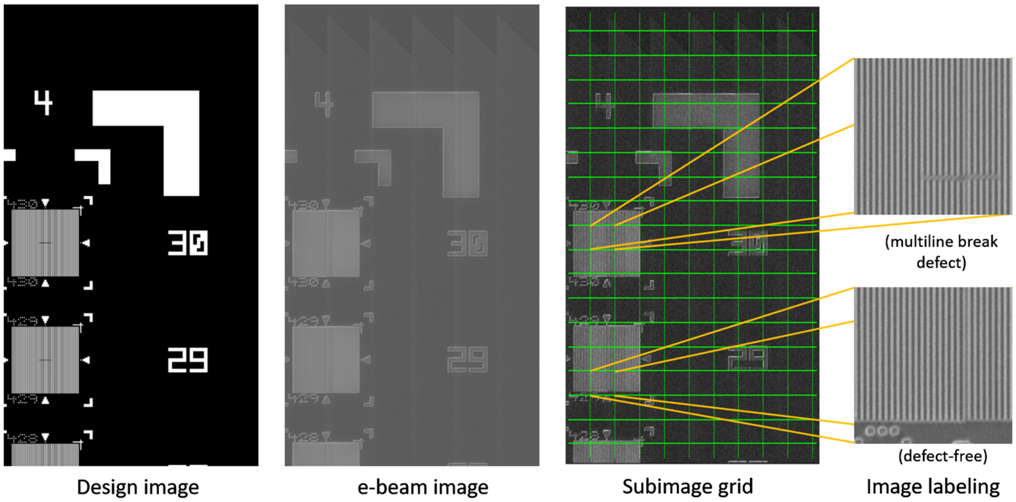


Fig. 3 Data generation.

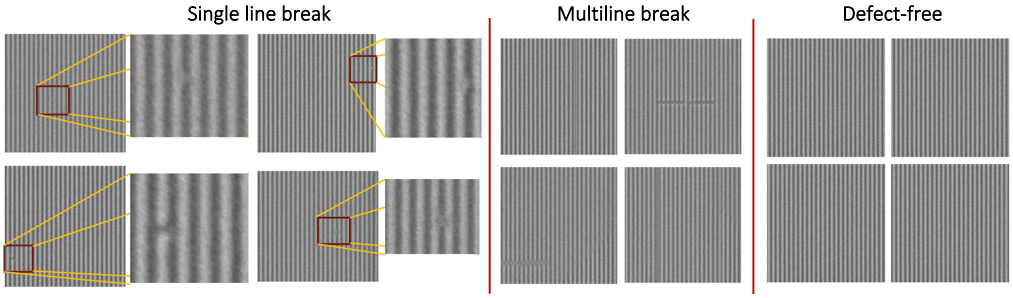


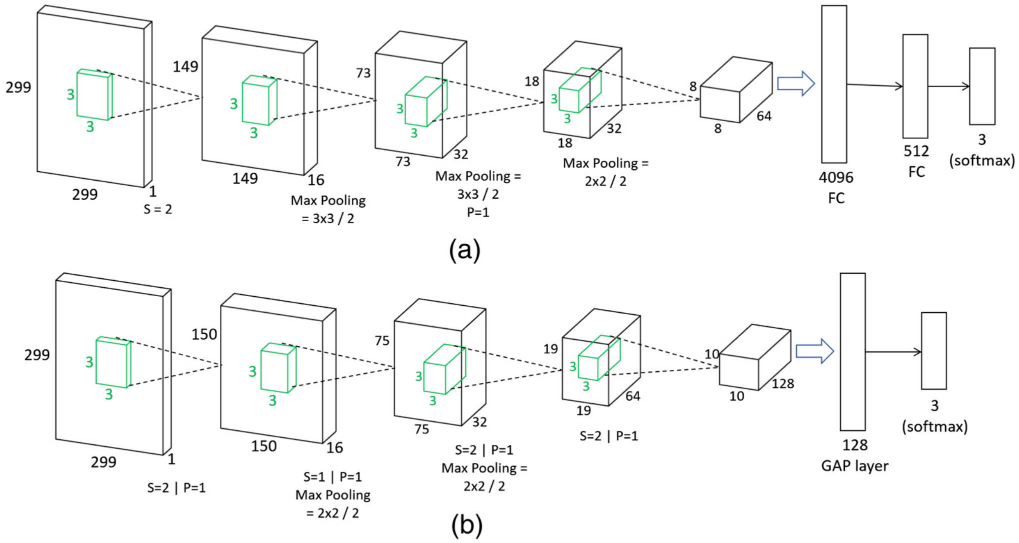
Fig. 4 Typical images of each class.

images containing a missing vertical line are placed within the multiline break defect class. Finally, all images that do not contain any defect are placed in the defect-free image class.

Table 1 represents the data distribution for this study. As shown in the table, there are 527 images in total that are split into training, validation, and testing sets. The training set is used to find the parameters of the model, the validation set is used to tune the hyperparameters of the model, and the testing set is used to evaluate and report the performance of the model. Out of the 216 images of single line break class, only 36 are unique images. These unique images are augmented (by a factor of 6) by reflection and translation to reduce the class imbalance problem. Similarly, the multiline break class contains only 52 unique images, which are then augmented by a factor of 2. This type of data augmentation is often employed in data-sparse applications to avoid overfitting and tackle class imbalance-related problems.

Table 1 Data distribution.

	Single line break	Multiline break	Defect-free	Total
Train	167	77	151	395 (~75.0%)
Validation	28	16	22	66 (~12.5%)
Test	21	11	34	66 (~12.5%)
Total	216	104	207	527



(a)

(b)

Fig. 5 CNN models. (a) CNN-FC: convolutional neural network with fully connected layers and (b) CNN-GAP:[22] convolution layers followed by GAP layer (S = stride; P = padding).

2.2 *CNN Models*

Figure 5 shows two different models used in this study. Both models take input image of size 299 × 299 and both have four convolutional stages, where each convolutional stage is comprised of several convolution filters followed by rectified linear units (ReLUs). The filter size, padding, stride, and number of filters for each layer are selected to ensure having rich feature maps of reasonable size at the end of last convolution stage.

In the CNN-FC model, as shown in Fig. 5(a), these feature maps are reshaped into a fully connected layer (of size 4096). This fully connected layer is then connected to a second fully connected layer of size 512 with a dropout. Dropout is a special type of regularizing technique[23] that helps to prevent overfitting and improves generalization ability. In the dropout layer, the activation of neurons is randomly set to zero with probability p in each training step. The value of p is usually tuned using the validation set. In this study, we observed that $p = 0.5$ attains highest validation accuracy. This second fully connected layer is connected to a softmax classifier of size 3 (number of classes).

Next, we consider CNN-GAP model. The architecture of this model is shown in Fig. 5(b). In addition of performing the standard classification task, CNN-GAP model can also generate CAMs highlighting the region most influential in the final prediction. This makes this deep model more interpretable by providing additional insights into the decision-making process of the model, making it a preferred choice of deep model in applications where it is critical to have some understanding of the internal working of a classifier. Furthermore, as we showcase in Sec. 3.3, for this particular application, CAM also helps to identify the exact location of defects in a weakly supervised fashion. In other words, we are able to identify the exact location of a defect inside an image without explicitly training the model with defect location information.

In CNN-GAP model, as shown in Fig. 6, the extracted feature maps after fourth convolution stage are connected to a GAP layer.[24] Specifically, we take the feature maps of the last convolution stage, perform a spatial average, and feed this average values to the GAP layer. This GAP layer is then connected to the final output layer for classification. Given this simple structure, we can think of the learned weights of this final layer as importance weights of the corresponding feature maps, as they indicate how important a particular feature map is in performing the final prediction. Hence, we can take the linear combination of the feature maps of the final convolution stage to generate CAMs.[22] Later, we describe this more formally for the three-class classification problem considered in this paper.

For a given image, let $a^k(x, y)$ represents the activation of k'th channel in the last convolutional stage at spatial location (x, y). Then, the value of unit k in the GAP layer is obtained by

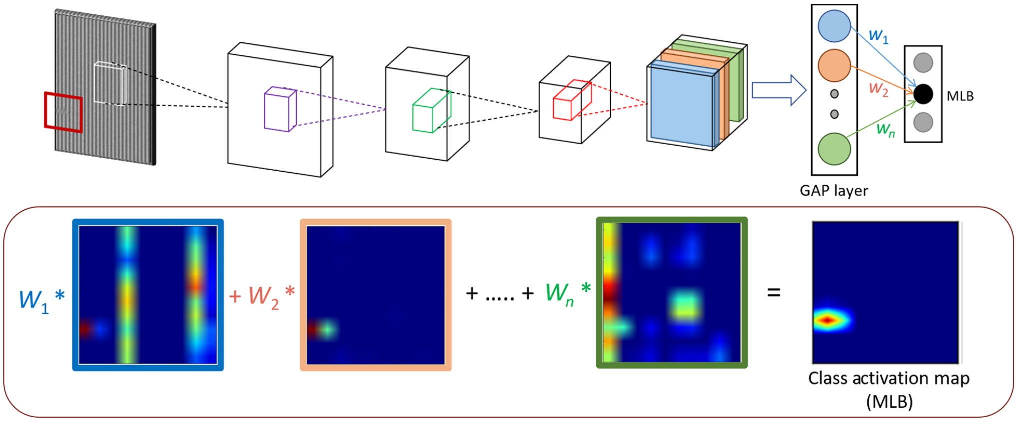


Fig. 6 CAM highlighting the location of defect is generated by taking linear combination of activations of last convolutional layer and upscaling the resulting map to original image size. For training this CNN-GAP model, only image-level labels (defect-free/single line break/multiline break) are provided and no information about exact location of defect is required; however, CAM is still capable of identifying the exact location of defect. (Visualization inspired by Ref. 22.)

performing global average pooling: $f^k = \sum_{x,y} a^k(x,y)$. Thus for a given class c (where c = single line break, multiline break, or defect-free), the input to final softmax classification layer, S_c, is $\sum_k w_c^k f^k$, where w_c^k is the weight corresponding to class c for unit k indicating importance of f^k for class c. Finally, the classification output for class c is given by softmax function $\frac{\exp(S_c)}{\sum_c \exp(S_c)}$. This calculation is schematically shown in top row of Fig. 6.

Let A_c be the CAM for class c, then $A_c(x,y) = \sum_k w_c^k a^k(x,y)$. This process is graphically depicted in second row of Fig. 6 as an example for multiline break. Note that the dimension of $A_c(x,y)$ is much smaller than input image; hence, we upscale it using bilinear interpolation to input image dimension. In Sec. 3.3, we provide more examples of CAMs generated by this CNN-GAP model for all three classes of images.

2.3 *CNN Training*

We use the training data (with distribution described in Table 1) to train these models for classification task and use validation set to tune the hyperparameters. Both models were implemented in Tensorflow[25] on a desktop with a single Nvidia 1080Ti GPU. The Adam optimizer[26] with appropriately tuned learning rate (1e-4) and momentum parameters ($\beta_1 = 0.9$, $\beta_2 = 0.99$) was used to train these models.

3 Results and Discussion

In this section, we quantify the performance of both models described in Sec. 2. We also analyze the filters of CNN-FC model and examine the convolution activation maps/heatmaps generated by CNN-GAP model to better understand the learning process.

3.1 *Defect Detection and Classification Task*

Figure 7 shows training and validation accuracy as a function of epochs for the CNN-FC and CNN-GAP models. In ML jargon, an epoch is completed when every example in training dataset has passed through one training cycle. As can be observed from Fig. 7, both training and validation accuracies improve almost monotonically as training progresses and there is no sign of overfitting.

Table 2 shows the performance of the fully trained CNN-FC and GAP models on the testing data in the form of a confusion matrix. As can be observed from this table for the CNN-FC

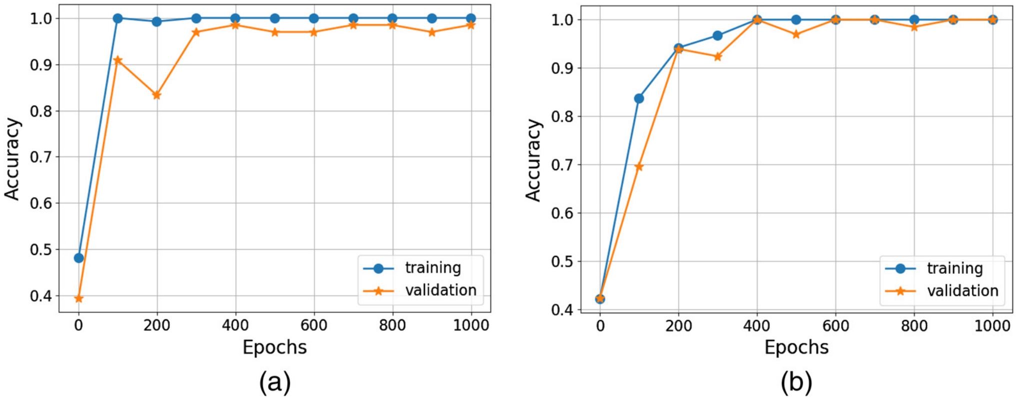


Fig. 7 Evolution of training and validation accuracy with epochs for (a) CNN-FC and (b) CNN-GAP models.

Table 2 Confusion matrix for CNN-FC and GAP architectures.

		Actual		
Test set ($N = 66$)		Defect-free	Multiline break	Single line break
Predicted (FC)	Defect-free	33	0	1
	Multiline break	0	11	0
	Single line break	1	0	20
Predicted (GAP)	Defect-free	34	0	1
	Multiline break	0	11	0
	Single line break	0	0	20

model, there is only one false negative and one false positive, indicating robust performance of this model. The overall accuracy is around 97% with a sensitivity of 97% and a specificity of 96.9%. We remind the readers that sensitivity is the ratio of defective images that are being classified correctly to the total number of defective images and specificity is the ratio of defect-free images that are being classified correctly to the total number of defect-free images. From this table, we also observe that the GAP model is even more accurate as it is able to correctly classify all but one image, resulting in an accuracy of 98.5%, sensitivity of 96.9%, and specificity of 100%.

We note that the false-negative image for CNN and GAP models was the same image containing a very tiny defect close to the image boundary. It is our suspicion that the models misclassified this image due to the defect size (one of the smallest defect in the dataset) and expect that by including more such images in the training dataset, our model might be able to classify this image correctly.

3.2 *Analysis of Convolution Layers*

In situations where important decisions are made based on the outputs of ML models, it is desirable to have some understanding of the underlying decision-making process of these learning models. One simple way of doing this is to analyze the activation maps and the corresponding convolution filters of a trained CNN model.

Figure 8 shows the typical activation maps at each layer for three different classes of images. The left most column represents three typical images from the three different classes. From top to bottom, these images represent the single line break, the multiline break, and the defect-free

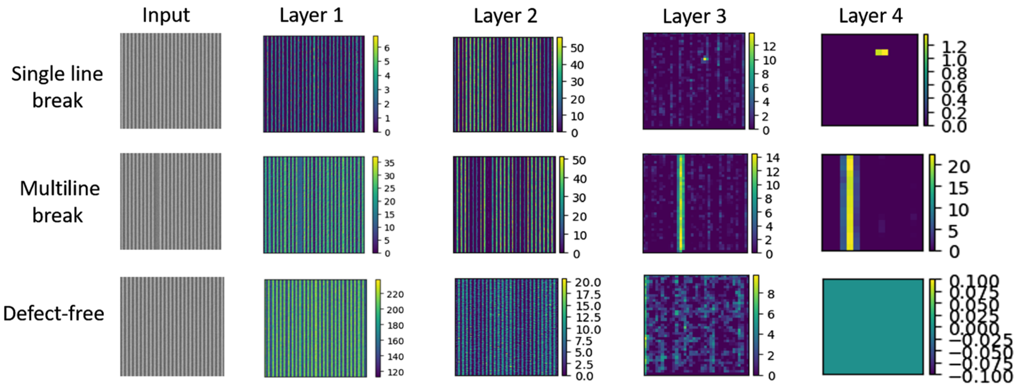


Fig. 8 Typical ReLU activation maps at each layer for three different class of images.

classes, respectively. Next, we pass each of these three images through a trained CNN-FC model and visualize the activation map at each layer starting from the first layer in column 2 to the fourth layer in column 5. Since each convolution stage is comprised of multiple filters, we obtain multiple activation maps at each layer. Due to space constraint, we only present a typical activation map image from each layer. As can be seen from these ReLU activation maps, the trained CNN-FC model is able to differentiate between defect and the background. It does so by highlighting the defects in layers 1 and 2, and then deleting the background in layers 3 and 4.

Next, we analyze the learned convolution filters. Figure 9 shows some typical active convolution filters (top row) of first three layers of the trained CNN-FC model and their corresponding Fourier transform (bottom row). In the Fourier map, the $k_x = k_y = 0$ component is at the center of the image and the horizontal and vertical axes represent wavenumbers (k_x and k_y, respectively) along those directions.

By analyzing these filters, we can qualitatively understand the operations at each layer. In layers 1 and 2, the filter on the left is a weighted average operator. This can be concluded by observing that all the convolution weights are greater than zero and the spectrum of the convolution in the Fourier domain is peaked at the origin. In contrast, the filter to the right appears to be close to a first-order derivative. This can be concluded by observing the left-right asymmetry in its weights, which translates to a dip in the Fourier spectrum at the origin followed by peaks along the horizontal direction. Thus, one may conclude that these layers are mainly concerned with smoothing and enhancing edges in the images.

For the third layer, the filter shown on the left is once again an averaging/smoothing filter. The filter on the right is a shift operator. This may be concluded by observing that it has nonzero entries on the left. Therefore, the operation of this filter would shift an image to right. It is likely that this filter was used by the network in conjunction with an averaging filter to delete the background as shown in Fig. 8.

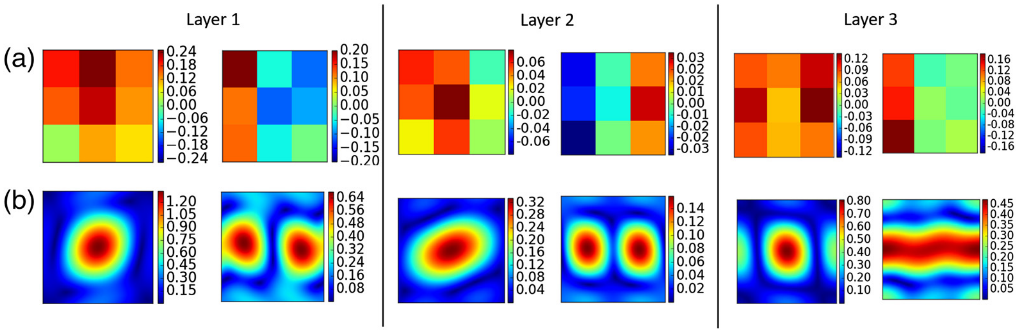


Fig. 9 (a) Convolution filters and (b) corresponding Fourier transform of some typical active filters of first three layers for CNN-FC model.

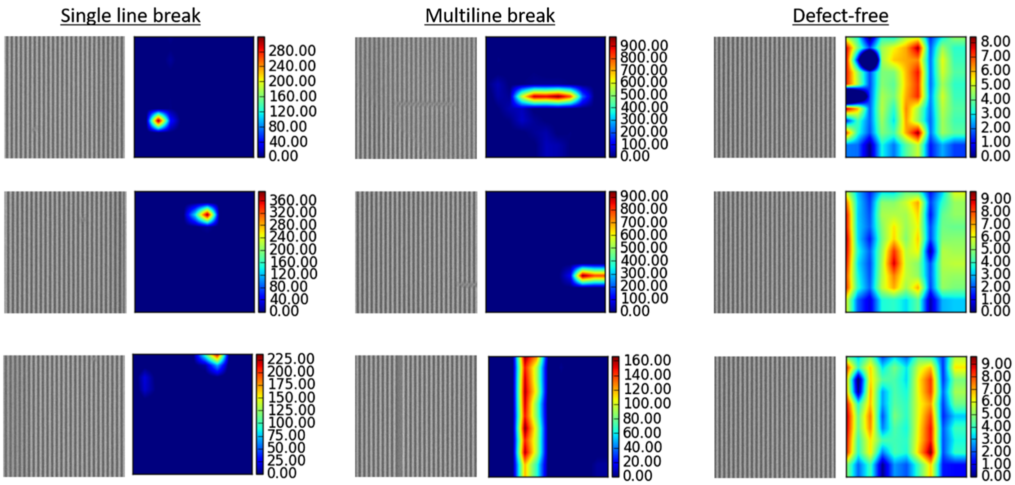


Fig. 10 CAMs obtained from CNN-GAP architecture.

3.3 *Defect Localization using Class Activation Maps*

As explained in Sec. 2, a slight modification of the standard CNN model with a GAP layer enables generation of CAMs/heatmaps that point to the regions that are most influential in making the final classification.

Figure 10 shows three typical images of each class and their corresponding CAM obtained through trained CNN-GAP model. As shown in the figure, the trained model has focused on the location that contains the defect to make the final prediction. This map can be used to localize the appropriate defect. This localization is obtained in an unsupervised mode, because when training the network, we only provide the label for a given image and not the location of a defect. The location is automatically learned by the net and can be visualized using the heatmap. It is also worth noting that for the image with no defect, the heatmap is much more spread out with overall low intensity values.

4 Toward More Challenging Classification Problems

In this section, we consider two variants of classification problem that are guided by the goal of extending the applicability of the ML algorithm to more challenging problems commonly encountered in semiconductor defect detection workflow, thereby reducing the burden on the human ally. In doing so, we use the same CNN architecture and hyperparameters that were used in the previous section.

4.1 *Electron Beam Images with More Variability*

First, we consider the use of EB images that are derived from the raw EB images with no preprocessing, which in turn introduces more variability. In particular, we waive the requirement of working only with images where the entire image is covered by the pattern. Instead, we allow images that are entirely or partially covered by a pattern, or have no pattern at all (see Fig. 11). This gives the human ally complete freedom in selecting any subregion of the larger EB image and passing it through the classifier, without needing to ensure that the entire image is filled by the pattern.

We train and test both the CNN-FC and CNN-GAP models using these types of images, where the breakup of the training/test data is described in Table 3. The confusion matrices obtained from the trained architectures are presented in Table 4 for FC and GAP architectures. We note that the networks are able to perform accurately even with inclusion of images with more variability (no patterns, partial patterns, and full patterns). The performance metrics for the FC network are: sensitivity = 94.8%, specificity = 100%, and accuracy = 97.6%, while those for the GAP network are: sensitivity = 98.3%, specificity = 91.9%, and accuracy = 94.9%.

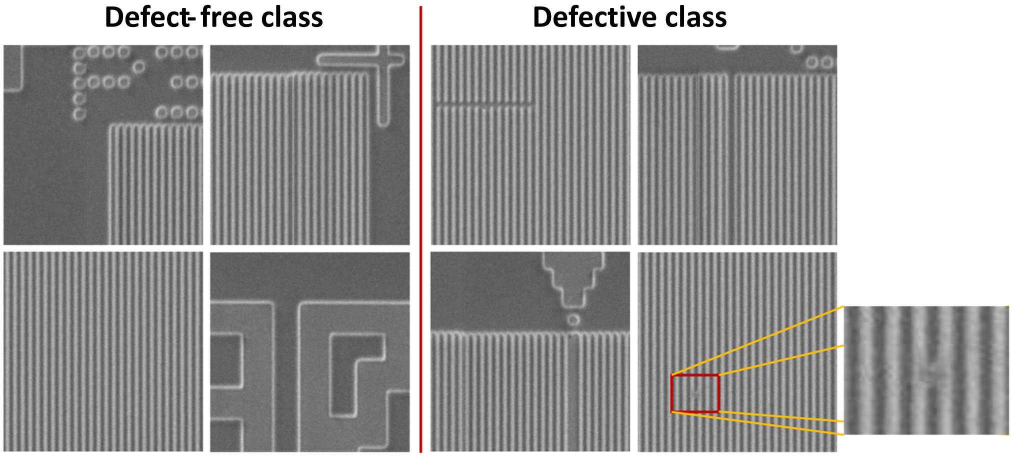


Fig. 11 Typical images for the vertical line pattern containing both patterned nonpatterned regions.

Table 3 Data distribution for images with higher variability.

	Defective	Defect-free	Total
Train	692	858	1550
Test	172	198	370
Total	864	1056	1920

Table 4 Confusion matrix for images with high variability.

		Actual	
Test set ($N = 370$)		Defect-free	Defective
Predicted (FC)	Defect-free	198	9
	Defective	0	163
Predicted (GAP)	Defect-free	182	3
	Defective	16	169

4.2 *Performance on Multiple Patterns*

Next we consider the following question: given a CNN-based classifier that is trained to work for a given pattern (source pattern), to what extent must it be retrained in order to classify similar defects in another pattern? We measure the extent of “retraining” of the network by the fraction of images from the new pattern (target pattern) in the training set. That is, the ratio of the number of images from target pattern to the total number of images from both (source + target) patterns. If this fraction is zero, no retraining is required. On the other hand, when this fraction approaches $1/2$, a significant amount of retraining is required. In the context of the results presented in the previous section, the original (source) pattern is the vertical line (line/space) pattern, and the new (target) pattern is a square (contact) pattern, that is considered for the first time in this section. Typical images selected from the square pattern are shown in Fig. 12.

First, we consider the two models from the previous section that were trained on images with a vertical line pattern with the data distribution described in Table 3. The performance of these models on images with vertical line patterns is presented in Table 4. We then use these models to

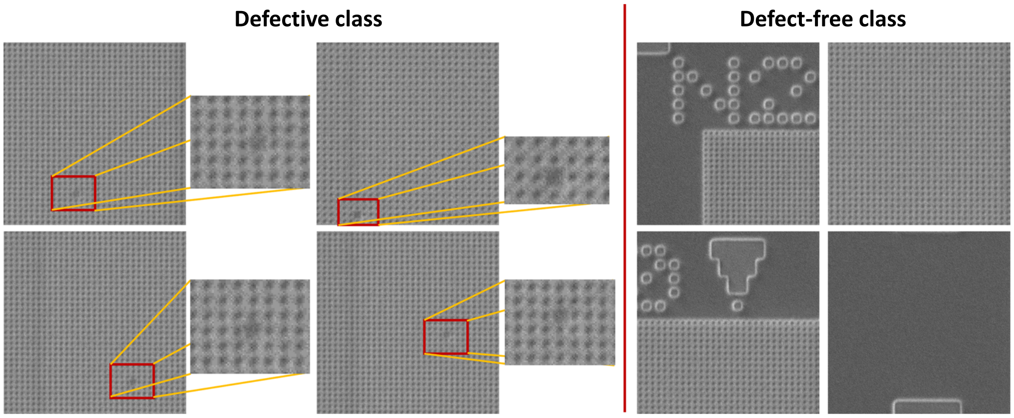


Fig. 12 Typical images for the square pattern containing patterned and nonpatterned regions.

Table 5 Confusion matrix for images with square contact pattern for the FC and GAP models.

		Actual	
Test set ($N = 328$)		Defect-free	Defective
Predicted (FC)	Defect-free	269	54
	Defective	5	0
Predicted (GAP)	Defect-free	274	54
	Defective	0	0

classify images obtained from a square contact pattern and report their performance in Table 5. We note that both models appear to classify most images as defect-free and demonstrate almost no ability to discriminate between images with and without defects, thus indicating that some retraining with the new pattern is necessary.

Next, we consider the effect of adding a small proportion of the square pattern images to the training set (see Table 6 for the distribution). In this table, the number within the rectangular parenthesis is the number of images from the square pattern set. The performance of the networks trained on this data is shown in Table 7. Even in this table the numbers within the rectangular parenthesis refer to images from square contact patterns. We observe that both models perform well for the entire data [vertical (line/space) and square (contact) patterns] and for data specifically from the square patterns. For the FC network, we report a sensitivity of 94.9% and a specificity of 90.9% for all images, and a sensitivity of 93.2% and specificity of 81.5% for images restricted to the square contact pattern, whereas for the GAP network, we report a sensitivity of 94% and a specificity of 93.6% for all images, and a sensitivity of 81.8% and

Table 6 Data distribution for the hybrid dataset with ~20% images from the square (contact) pattern. The number of images from square (contact) pattern is shown in rectangular parenthesis and from vertical (line/space) pattern is shown outside.

	Defective	Defect-free	Total
Train	864 [172]	1065 [220]	1929 (~80.0%)
Test	216 [44]	265 [54]	481 (~20%)
Total	1080 [216]	1330 [274]	2410

Table 7 Confusion matrix for hybrid dataset for the CNN models.

Value set ($N = 481$)		Actual Defect-free	 Defective
Predicted (FC)	Defect-free	241 [44]	11 [3]
	Defective	24 [10]	205 [41]
Predicted (GAP)	Defect-free	248 [53]	13 [8]
	Defective	17 [1]	203 [36]

specificity of 98.2% for images restricted to the square contact pattern. This leads us to conclude that even adding a small amount of data from the target pattern can be useful in getting the models to perform well with the new patterns.

5 Conclusions and Future Work

In this work we have demonstrated the effectiveness of optimized deep learning-based models in identifying, localizing, and classifying different types of wafer defects with high degree of accuracy. We have achieved this by training two CNN-based models on high-resolution EB images of patterns on a wafer with different types of intentionally placed defects. We have further analyzed the convolution filters and corresponding activation maps to better understand the learning process of CNN models. We have trained a CNN model with a GAP layer to generate CAMs and demonstrated that these maps can be used to localize defects. We have also examined the effect of introducing more variability in the input image dataset, which helps in reducing the amount of preprocessing required. Finally, we have considered how these models might be retrained so that they can be applied to a new pattern to detect and classify similar types of defects.

As the technology nodes further shrink making traditional defect inspection systems less efficient and more expensive, we hope that this work will demonstrate and motivate the usefulness of deep learning-based models for effective defect detection, classification, and localization.

There are a couple of interesting directions for further research. One extension would be to develop a pattern-invariant model that does not require any images from the target pattern. Another interesting and practically useful avenue would be to perform detection and classification in an unsupervised or semisupervised fashion. One possible solution for this is the use of deep generative models, such as GAN, and pose the defect detection problem as an anomaly detection problem.

Acknowledgments

This work was performed at various IBM Research and Development sites. The authors would like to acknowledge Derren Dunn for program support.

References

1. R. Bonam et al., “E-beam inspection of EUV mask defects: to etch or not to etch?” *Proc. SPIE* **9048**, 904812 (2014).
2. S. Ren et al., “Faster R-CNN: towards real-time object detection with region proposal networks,” in *Adv. Neural Inf. Process. Syst. 28*, C. Cortes et al., Eds., Curran Associates, Inc., pp. 91–99 (2015).
3. I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Proc. Adv. Neural Inf. Process. Syst. 27*, pp. 3104–3112 (2014).

4. S. Reed et al., "*Parallel multiscale autoregressive density estimation*," Technical Report.
5. D. Silver et al., "Mastering the game of Go without human knowledge," *Nature* **550**, 354–359 (2017).
6. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Adv. Neural Inf. Process. Syst.*, pp. 1–9 (2012).
7. S. Ren et al., "Faster R-CNN: towards real-time object detection with region proposal networks," *IEEE Trans. Pattern Anal. Mach. Intell.* **39**, 1137–1149 (2017).
8. K. He et al., "Deep residual learning for image recognition," in *IEEE Conf. Comput. Vision and Pattern Recognit.* (2015).
9. M. Shin and J.-H. Lee, "Accurate lithography hotspot detection using deep convolutional neural networks," *J. Micro/Nanolithogr. MEMS MOEMS* **15**, 043507 (2016).
10. H. Yang et al., "Layout hotspot detection with feature tensor generation and deep biased learning," in *Proc. 54th Annu. Des. Autom. Conf.*, ACM Press, New York, pp. 1–6 (2017).
11. T. Matsunawa et al., "A new lithography hotspot detection framework based on AdaBoost classifier and simplified feature extraction," *Proc. SPIE* **9427**, 94270S (2015).
12. H. Yang et al., "Imbalance aware lithography hotspot detection: a deep learning approach," *Proc. SPIE* **10148**, 1014807 (2017).
13. T. Nakazawa and D. V. Kulkarni, "Wafer map defect pattern classification and image retrieval using convolutional neural network," *IEEE Trans. Semicond. Manuf.* **31**, 309–314 (2018).
14. C. H. Jin et al., "A novel DBSCAN-based defect pattern detection and classification framework for wafer bin map," *IEEE Trans. Semicond. Manuf.* **32**, 286–292 (2019).
15. J. Yu, X. Zheng, and J. Liu, "Stacked convolutional sparse denoising auto-encoder for identification of defect patterns in semiconductor wafer map," *Comput. Ind.* **109**, 121–133 (2019).
16. J. Wang et al., "AdaBalGAN: an improved generative adversarial network with imbalanced learning for wafer defective pattern recognition," *IEEE Trans. Semicond. Manuf.* **32**, 310–319 (2019).
17. R. Bonam et al., "E-beam inspection of EUV programmed defect wafers for printability analysis," in *ASMC SEMI Adv. Semicond. Manuf. Conf.*, IEEE, pp. 310–314 (2013).
18. D. Patel, R. K. Bonam, and A. Oberai, "Engineering neural networks for improved defect detection and classification," *Proc. SPIE* **10959**, 1095910 (2019).
19. J. Tremblay et al., "Training deep networks with synthetic data: bridging the reality gap by domain randomization" (2018).
20. D. Patel et al., "Circumventing the solution of inverse problems in mechanics through deep learning: application to elasticity imaging," *Comput. Methods Appl. Mech. Eng.* **353**, 448–466 (2019).
21. X. B. Peng et al., "Sim-to-real transfer of robotic control with dynamics randomization," in *IEEE Int. Conf. Robotics Automation (ICRA)*, pp. 1–8 (2018).
22. B. Zhou et al., "*Learning deep features for discriminative localization*," in *IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, pp. 2921–2929 (2016).
23. N. Srivastava et al., "Dropout: a simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.* **15** (2014).
24. M. Lin, Q. Chen, and S. Yan, "Network in network," in *2nd Int. Conf. Learn. Represent. Conf. Track Proc.* (2014).
25. M. Abadi et al., "Tensorflow, large-scale machine learning on heterogeneous distributed systems," https://www.tensorflow.org/ (2015).
26. D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization," arXiv:1412.6980 (2014).

Dhruv V. Patel is a PhD student in the Aerospace and Mechanical Engineering Department at the University of Southern California. He received his bachelor's degree in mechanical engineering from L.D. College of Engineering, Ahmedabad, India, in 2013 and master's degree in applied mechanics from Indian Institute of Technology, Delhi, in 2016. His current research interests include physics-based data-driven modeling, deep learning, and Bayesian inference. He is a student member of SPIE.

Ravi Bonam is a senior member of research staff at IBM Semiconductor Technology Research Division and currently leads design, integration, and enablement for heterogeneous integration applications. He received his bachelor's degree in computer science and systems engineering from Andhra University in 2006, an MS degree in electrical and computer engineering from the Missouri S&T (formerly University of Missouri, Rolla) 2008, and a PhD in material science from SUNY-Albany in 2012. His current research interests include heterogeneous integration, Artificial Intelligence, microarchitecture, and chip design. He is a member of the IEEE.

Assad A. Oberai is the Hughes professor of engineering in the Aerospace and Mechanical Engineering Department at the Viterbi School at the University of Southern California. Prior to joining USC, he was a professor in the Department of Mechanical Aerospace and Nuclear Engineering at Rensselaer Polytechnic Institute, the associate dean for research and graduate studies in the School of Engineering, and the associate director of the Scientific Computation Research Center. He was an assistant professor of aerospace and mechanical engineering at Boston University from 2001 to 2005, and joined the Rensselaer faculty in 2006. He received a PhD from Stanford University in 1998, an MS from the University of Colorado in 1994, and a bachelor's degree from Osmania University in 1992, all in mechanical engineering. He heads the Computation and Data Driven Discovery group that designs, implements, and applies data- and physics-based models and algorithms to solve interesting problems in engineering and science. He is a fellow of ASME, USACM, and AIMBE.