Defect recognition in line-space patterns aided by deep learning with data augmentation

Jihun Ahn,[a] **Ye Chan Kim**,[b] **So Youn Kim**,[b] **Su-Mi Hur**,[a,c,*] **and Vikram Thapar**[a,c,*]

[a]Chonnam National University, Graduate School, Department of Polymer Engineering, Gwangju, Republic of Korea

[b]Seoul National University, School of Chemical and Biological Engineering, Seoul, Republic of Korea

[c]Chonnam National University, Alan G. MacDiarmid Energy Research Institute, School of Polymer Science and Engineering, Gwangju, Republic of Korea

Abstract

Background: Finding optimal processing conditions to reduce defectivity is a major challenge in high-resolution lithographic tools such as directed self-assembly and extreme ultraviolet lithography.

Aim: We aim to develop an efficient automated method that can detect defects and identify their types and locations, allowing for assessing the performance of lithographic processing conditions more easily.

Approach: We propose a deep learning approach using an object recognition neural network for the classification and detection of defects in scanning electron microscopy (SEM) images featuring line-space patterns. We optimized our network by exploring its design variables and applied it on a case with limited numbers of SEM images available for training the network.

Results: With an optimized network and data augmentation strategies (flipping and mixing images from simulations), it was possible to achieve a significant increase in the performance of the network. Transferability of the network was also proven when applied on a more diverse dataset of SEM images gathered from selected publications.

Conclusions: The outcome of our work shows that the amalgamation of an optimal network design and augmentation strategies performs satisfactorily for defectivity analysis and is generic for data not constrained to fixed settings.

 [DOI: 10.1117/1.JMM.20.4.041203]

Keywords: lithography; neural networks; defects; line-space patterns; data augmentation.

Paper 21038SS received Apr. 12, 2021; accepted for publication Aug. 11, 2021; published online Sep. 24, 2021.

1 Introduction

The progressive scaling of transistors in semiconductor manufacturing demands cost-effective lithographic techniques capable of smaller feature patterning with feature sizes below the 10-nm scale. As potentially viable next-generation lithography candidates for features with line-space (L/S) patterns, several techniques have been proposed; among these, extreme ultraviolet lithography (EUVL) and directed self-assembly (DSA) are two of the most promising. EUVL is an approach relying on patterning features using a short-wavelength light source of 13.5 nm,[1] whereas DSA exploits the self-assembly behavior of block copolymers that undergo microphase separation when coated on a wafer to create dense periodic features over large areas.[2,3] Compared with EUVL, DSA has a lower cost of ownership because of the reduced number

*Address all correspondence to Su-Mi Hur, shur@chonnam.ac.kr; Vikram Thapar, thapar.09@gmail.com

of processing steps, but it is less flexible in regards to print patterns with variable pitches. In addition, one of the major challenges in applying DSA to high-volume manufacturing is the observed defect densities, which are larger than the required defect density of 1 and 0.01 defect/cm^2 for memory and logic applications, respectively. The most commonly observed defects are bridges and dislocations. Even EUVL is not free of defectivity issues, as noted in previous work,[4,5] and is shown to make bridge defects.

To address the concern of large defect densities, especially in DSA, various process optimization steps are put into use to determine the important factors that can contribute to reductions in the overall defect density; optimization steps include varying annealing conditions, periodicity of the surface pattern, width of the guiding line, topography of a pattern, and background chemistry, among others. For every combination of the listed processing steps, it is necessary to perform defect inspections of scanning electron microscopy (SEM) images to evaluate the performance of the processing conditions. This involves collecting large enough numbers of SEM images for statistical purposes and performing defect detection either manually or using an image processing tool. The manual labeling of defects becomes inefficient as the number of different combinations of processing steps increases. One of the solutions is to use emerging deep learning algorithms to detect and classify defects of different types. In the field of material science, numerous algorithms have been applied to learn complex defective features from a given set of images. For example, (1) Xie et al.[6] used the multi-class support vector machine algorithm to detect most regularly observed defects both in printed circuit boards and wafers; these defects involve rings, semicircles, clusters, and scratches. (2) Zheng and Gu[7] adopted the machine learning algorithm to detect the unknown number of multiple vacancies in graphene with high accuracy. (3) Tabernik et al.[8] reported a study in which they applied segmentation-based deep learning architecture to detect surface anomalies in finished products from the perspective of certain industrial applications. The deep learning-assisted-identification of defects is not restricted to the field of material science and has been used in various other fields for purposes such as defect detection in sewer pipes[9,10] and fruit defect detection.[11] We believe that the use of such automated methods for counting different types of defects, as well as specifying their locations in the line and space (L/S) patterns, could assist process engineers in quickly collecting enough statistics and provide a more accurate and consistent method of evaluating each combination of processing conditions.

Generally, a large number of training samples are required to ensure high accuracy of the network. Unfortunately, as mentioned earlier, the time-consuming process of labeling the defects present in SEM images is demanding because of the required load of human effort and expertise. This creates an impediment to collecting sufficient data for the desired precision of the deep learning network. Data augmentation is one viable option to inflate the training dataset by exploiting more information from the original dataset. As discussed in the review paper by Shorten and Khoshgoftaar,[12] augmentation strategies include geometric and color transformations, random erasing, and feature space augmentation. Flipping images, one of the easiest and computationally cheapest strategies, combined with other geometric transformations of cropping, rotating, and scaling is shown to improve the accuracy of the deep learning algorithm.[13,14] Another data augmentation method is to expand the dataset by performing simulations. Such a strategy is explored in the classification of astronomical events in Carrasco-Davis et al.,[15] in which the authors rely on a physics-based model to generate the simulated dataset. In Ref. 16, when a simulated dataset generated with a point-scattering model for radar image simulation as described by Holtzman et al.[17] was mixed with a real dataset, there was a boost in improvement to the accuracy of target recognition in synthetic aperture radar images in ships.

In this work, with the use of a minimal SEM dataset for training [O(100) images], we use an object classification and detection network inspired by the well-established version 3 of You Only Look Once (YOLOv3)[18] to predict the location and type of defects present in images. The SEM dataset was collected after carrying out experiments using cylinder forming block copolymers under shear-solvo annealing conditions.[19] The numbers of convolutional layers and filters in a network were optimized for the network's accuracy. Further examination of various activation functions and different loss functions was implemented. The initial dataset with a limited number of SEM images was inflated using two strategies: (1) geometric transformation

through flipping images horizontally, vertically, and horizontally followed by flipping vertically and (2) augmentation with simulated images. The simulated images were obtained using a physics-driven theoretically informed coarse-grained (TICG) model[20–22] that has been successfully applied to describe experimental observations in block copolymer systems. Changes in the network's accuracy on expanding the initial SEM dataset of various sizes using the two aforementioned strategies were systematically investigated. Given the flexibility of customizing the type and size of defects in simulated images, the importance of the distribution of the simulated dataset was probed. Finally, the generalizability of our network was demonstrated by testing it on a more diverse DSA dataset generated with various polymers, processing conditions, or substrate's guiding patterns and on L/S patterns from photo/EUV lithography collected from publications.[23–58]

In Sec. 2, we briefly describe the type of detected defects studied in this work. For background, in Sec. 3, we discuss how the YOLO algorithm works and further describe the network architecture used in this work. In Sec. 4, we provide details on the preparation of both experimental and simulation datasets. Section 5 shows the results obtained in this work. In Sec. 6, we provide concluding remarks, highlighting some future directions.

2 Types of Detected Defects

A deep learning network was developed to detect the most commonly observed defects in line and space patterns. The defects include an edge dislocation (ED), dislocation pole (DP), dislocation dipole (DD), and bridge. Postprocessed SEM images containing each of these defects highlighted in dark yellow boxes are shown in Fig. 1 (details on the processing of SEM images are given in Sec. 4.1). We refer to the black- and white-colored regions as regions filled with polymeric species A and B, respectively, in the following explanation without the generality loss. The DP consists of either the A or B domain terminated in the middle of the regular lamellar domain, with distorted nearby planes of the internal AB interfaces. ED is similar to DP in that it also has the A/B terminated domain; however, in ED one of the terminated ends connects with the adjacent layer without distorting nearby planes as shown in Fig. 1. The pairs of DPs with opposite Burger vectors constitute a DD. Note that individual DPs that make a dipole can be single/multiple lamellae layers apart from one another. In the image shown in Fig. 1, the DD defects are five lamellae layers apart. Bridge (B) defects, as shown in Fig. 1, occur when chains of one BCP block propagate across a domain of the opposite block and form a "bridge" between two nearby domains of the same block type.

3 Deep Learning Method for Detecting Defects

For defect detection, we used a deep learning framework influenced from the YOLOv3 developed by Redmon and Farhadi.[18] YOLOv3 is an updated version of the YOLO algorithm that has been successfully applied in autonomous driving, real-time detection of traffic participants,[59] real-time unmanned aerial vehicles detection,[60] breast mass detection,[61] fruit detection,[11] underwater fish detection,[62] and sewer pipe defect detection.[9] It is an object detection algorithm that predicts not only the type of object labels but also its location by drawing a bounding box

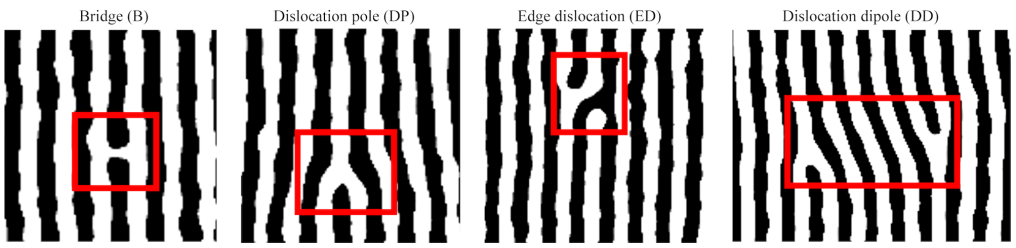

Fig. 1 Postprocessed SEM image for each type of defect: bridge (B), DP, ED, and DD. Red box shows the location of the corresponding defect in the image.

enclosing a given object. The algorithm is also capable of detecting multiple objects within an image. In comparison with the other well-known object detection algorithms such as fast R-CNN and faster R-CNN, in which the object detection is performed through extracting regions of interest for the classification of objects, a predefined grid cell is used in YOLO to carry out the object prediction, resulting in faster execution of the algorithm.

We begin with a high-level overview of the network/architecture used in this work. Similar to YOLOv3, the entire network is branched into two major segments: multi-scaled feature extractor and detector. As illustrated in Fig. 2, an input image is fed into the feature extractor first to realize the feature embedding at three different scales. These embeddings are delivered to three sections of the detector to get the classes to which the objects belong to and the object bounding boxes.

The feature extractor is marked with a red rectangular box in Fig. 2. A 416×416 sized input feature vector is fed into the feature extractor composed of a series of convolutional layers with the task of selecting features from prior layers by performing a convolution of input arrays. Each convolutional layer is followed by batch normalization and a Leaky ReLu layer. The number of filters for each convolutional layer is expressed in the units of F, where F is defined as the number of filters for the first convolutional layer. The value of F is 32 in the original implementation of YOLOv3 and the filter dimensions are either 3×3 or 1×1. The stride value in the convolutional layers is set to be one except when the feature vector is downsampled; the vector's dimension is reduced by half using a stride value of 2. The detailed layout of residual blocks introduced in the feature extractor unit is shown in the top right of Fig. 2. Inside a residual block, a 1×1 convolutional layer is followed by a 3×3 convolutional layer plus a skip connection. Residual blocks are added to solve the gradient disappearance or gradient explosion problems in the network, allowing for easier control of the gradient propagation and improved network training. To ensure that the important features are retained during the operation of downsampling/condensing the feature vector, the number of residual blocks are increased as we go deeper into the network. In this work, the number of residual blocks at the initial stage of the network is set to its minimum value of 1 (the same as YOLOv3), and in the latter stages, it is expressed in R units; the R value of 2 is used in YOLOv3. In Sec. 5, we find the optimal F and R values through an exploratory study in which we methodically vary F and R values and compare the accuracy of the network for defect detection.

Just like YOLOv3, the network implemented in this work is designed as a multi-scale detector. Three 52×52, 26×26, and 13×13 dimensional feature vectors are obtained as an output

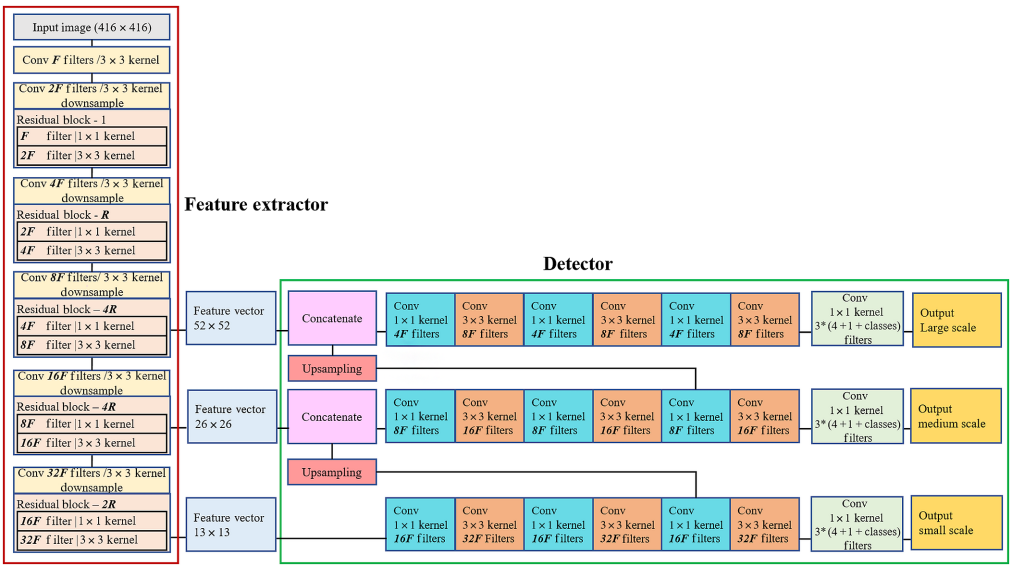

Fig. 2 The architecture of our network. Similar to YOLOv3, the entire network is branched into two major segments: the multi-scaled feature extractor enclosed in the red box and the detector enclosed in the green box. The feature extractor realizes the feature embedding at three different scales (13×13, 26×26, and 52×52); these embeddings are then delivered to three sections of the detector to get the classes to which the objects belong and the object bounding boxes.

from the feature extractor and are supplied to the detector. The detector, which is shown inside a green box in Fig. 2, has multiple 1×1 and 3×3 convolutional layers and a final 1×1 convolutional layer. The feature vectors at medium and small scales are concatenated with the previous scale feature vectors as an upsampling operation, allowing small-scale detection to benefit from the result of large-scale detection. The final output of the detector is a tensor augmented with the outputs of three different scales in the shape of $[(52,52,\ 3,\ (4+1+N_c))$, $(26,26,\ 3,\ (4+1+N_c))$, and $(13,13,\ 3,\ (4+1+N_c))]$, where N_c is the number of object classes. (In this work, it is the number of different types of defects with a value of 4.) Before we explain the values embedded in each of three entries in a detector tensor, it is essential to introduce the concept of the anchor box in Fig. 3.

The object detection algorithm aims to serve the dual purpose of correctly predicting a bounding box of a given object and its class. The notion of grid cells is now introduced, and these are constructed by dividing the image into 13×13, 26×26, or 52×52 square matrices. For each grid cell, our network predicts three bounding boxes. A bounding box is represented by four variables defined as $x_{\min}$, $x_{\max}$, $y_{\min}$, and $y_{\max}$; all four values are normalized with respect to the image's size (416). Due to the large variance in scale and aspect ratio of ground truth boxes, learning the bounding box variables through random initialization is highly inefficient. Therefore, in Ref. 18, an anchor box is used instead of performing bounding box detection from the random initial guessing. Anchor boxes with different aspect ratios are predefined by k-means clustering on the entire dataset. During the training, our network bounding box is searched by predicting offsets against the anchor boxes. The formulas to obtain the normalized real coordinates of the predicted bounding boxes from the location offsets are given in Fig. 3. As three scales of grids are used, we have a total of $52 \times 52 \times 3$, $26 \times 26 \times 3$, and $13 \times 13 \times 3$ predicted bounding boxes. For each predicted bounding box, three essential attributes are trained as follows.

(1) Four values for the location offset against the anchor box are defined as t_x, t_y, t_w, and t_h.
(2) Objectness score is the probability that a given grid cell does or does not contain an object.
(3) Class probabilities of the object belong to each class out of a total N_c classes.

In total, $4 + 1 + N_c$ values are obtained for each of the three predicted bounding boxes at three different scales, which explains the shape of the output tensor from the detector.

The total loss function, which calculates the loss of the output from the detection unit against the ground truth labels, is composed of three terms: bounding box, objectness, and the

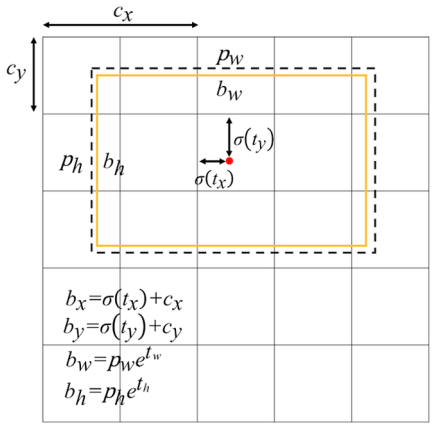

Fig. 3 Prediction of the bounding box attributes. x and y center coordinates of the predicted bounding box are given by b_x and b_y, respectively, and width and height are given by b_w and b_h, respectively. The values c_x and c_y are the top left corner coordinates of a grid cell containing the center of the object. The relations between predicted bounding box and predefined anchor box width and heights (p_w and p_h are anchor box width and height, respectively) are shown in orange. In the equation for the center of coordinates for the bounding box, σ represents the sigmoid function.

classification loss. For a complete mathematical description, we refer readers to Redmon and Farhadi.[18] In this work, generalized intersection over union (GIoU) loss is used for bounding box predictions, and it is shown to have a faster convergence and better accuracy than a simple intersection over union (IoU) loss.[63] Here IoU is the ratio of the area of intersection between two boxes over that of the union of those boxes. The focal binary cross entropy (FCE)[64] is used for objectness loss, and cross entropy (CE) loss[18] is used for classification loss. During the prediction, we keep the bounding boxes with high objectness scores (>0.5). To eliminate the predicted bounding box duplicates, an algorithm named non-maximum suppression, which is explained in detail by Hosang et al,[65] is applied.

4 Data Collection

This section provides the details on how we acquired data used for training and testing the network. Two types of datasets, one from experimental SEM images and the other from performing Monte Carlo simulations, were collected.

4.1 *Experimental Dataset*

The experimental dataset of 575 L/S pattern SEM images was acquired by performing experiments on cylinder-forming block copolymers. The experimental protocol details are published in a previous paper by Kim et al.[19] In summary, once block copolymers are spin coated, the dual processing steps of shear alignment and solvent vapor annealing (SVA) in sequence were performed. The shear aligned BCP thin film obtained by applying shear stress with a cured polydimethylsiloxane pad undergoes SVA treatment in a glass chamber at room temperature. The treated BCP thin films were characterized using SEM (Hitachi S-4800) with an operation energy of 5 keV and a working distance of 3 mm. For successful defect detection, samples obtained under experimental conditions that left several defects were used for SEM image collection.

Raw SEM images are processed to remove the noise and blurriness by applying two of the most commonly used image processing steps: digital unsharp mask filtering and Gaussian blur filtering (filters loaded from the PIL python library). The flowchart for refining raw SEM images is described in Fig. 4. The digital unsharp mask filtering is characterized by three variables: (1) blur radius, which serves the purpose of blurring an image by setting each pixel to the average value of the pixels in a square box extending the radius pixels in each direction, (2) unsharp

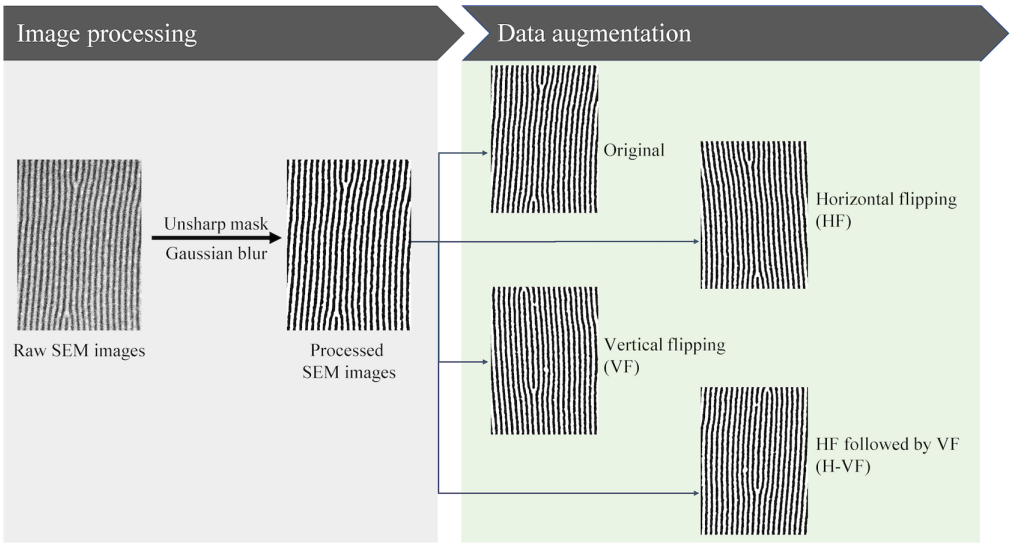

Fig. 4 Flowchart for the processing of raw SEM images. The raw images are first processed to remove noise and blurriness by applying digital unsharp mask filtering and Gaussian blur filtering. To expand the number of images during network training, the three operations of HF, VF, and H-VF are performed as shown under the data augmentation block.

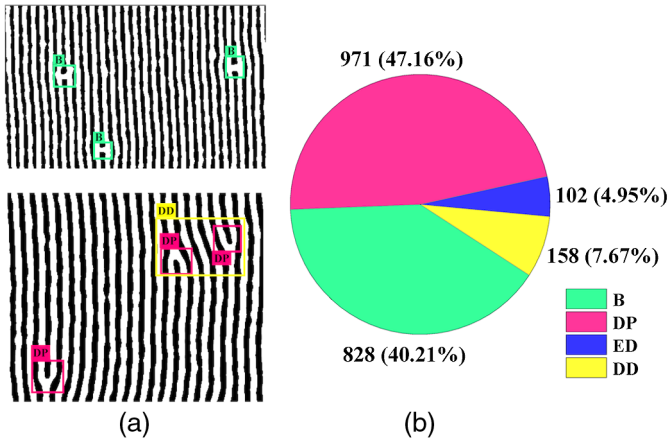

Fig. 5 (a) Top and bottom are the sample representative processed SEM images with annotated defects. Bridge (B), DP, ED, and DD are marked in green, dark pink, blue, and yellow, respectively. (b) Distribution of each type of defect in the whole SEM dataset.

strength in percent, which controls the magnitude of how much darker or brighter the pixels will be made, and (3) threshold parameter, which prevents the filter from sharpening the image unless the difference between adjacent pixels is large enough. Gaussian blur uses a blur radius defined above as its only parameter. The values of blur radius, unsharp strength, and threshold parameter in this work are 2, 10, and 500, respectively. After exploring the different sequences of application of the two filters, a filtering sequence of (1) Gaussian blur, (2) unsharp mask, (3) Gaussian blur, (4) unsharp mask, and (5) unsharp mask gives the non-blurred image without losing important features present in it. Figure 4 compares the original SEM image and the processed SEM image.

The number of images in the training dataset is expanded through performing the three operations of horizontal flipping (HF), vertical flipping (VF), and horizontal followed by vertical flipping (H-VF) as shown in Fig. 4. The defects present in the refined SEM image dataset are manually annotated by classifying them into four different categories of DP, ED, DD, and B as described in Sec. 2 [see Fig. 5(a), which shows drawn bounding boxes enclosing defects]. SEM images are rescaled to an input layer size of 416×416 pixels. By padding with the pixel value of 128 (in gray), the aspect ratio of a given image is unperturbed during the image rescaling. Our SEM image dataset contains 2059 defects with the percentage of each type of defect shown in Fig. 5(b); the minimum and maximum numbers of defects per image are 0 and 19, respectively. Note that the number of ED and DD defects in our dataset is an order of magnitude smaller than that of the B and DP defects. The prepared dataset has a vast range in the spacing between adjacent black/white lines, which is also defined as the pitch in L/S patterns. The pitch size ranges from a minimum value of 6 to a maximum value of 72 pixels, with an average value of 18; this prevents the network from overfitting.

4.2 *Simulation Dataset*

To prepare a simulation dataset, we use TICG combined with Monte Carlo simulations.[20–22] This model has been rigorously studied and validated with available experimental data for copolymer thin films. The TICG model is a particle-based coarse-grained model that adopts a representation in which each polymer chain is represented by a number of coarse-grained beads N. Here we only recall the model's main characteristics to simulate our system of interest. All n A-b-B block copolymer chains in our system are in a constant volume and constant temperature environment and are discretized into N beads. The bonded interactions of polymer chains adapting a Gaussian random-walk configuration of coarse-grained polymer beads is represented by harmonic springs attached between adjacent beads in a given chain. The total harmonic potential at a given temperature is then defined as

$$H_b = \frac{3k_bT}{2}\frac{N-1}{Re^2}\sum_{k=1}^{n}\sum_{i=1}^{N-1}\boldsymbol{b}_k^2(i), \tag{1}$$

where $\boldsymbol{b}_k(i)$ is a vector connecting the i'th and $(i+1)$'th beads in a chain k, R_e is the mean-squared end-to-end distance for an isolated non-interacting chain, and k_b is the Boltzmann constant. The non-bonded interactions are functional of local densities $\phi_A(\mathbf{r})$ and $\phi_B(\mathbf{r})$ and are given by

$$\frac{H_{nb}[\phi_A,\phi_B]}{k_BT} = \sqrt{\overline{N}}\int_V \frac{dr}{R_e^3}\left[\chi N\phi_A\phi_B + \frac{\kappa N}{2}(1-\phi_A-\phi_B)^2\right], \tag{2}$$

where the first term represents the incompatibility between A and B beads and is a function of the Flory–Huggins parameter χ. The second term, which is derived from the Helfand quadratic approximation, is the energy penalty due to the deviation of local bead densities away from its average value in a nearly incompressible dense polymer melt, and it is a function of the incompressibility parameter κ. The term $\sqrt{\overline{N}}$ is an interdigitation parameter that provides an estimate of the number of chains with which a given chain interacts. The values of N, κN, χN, and $\sqrt{\overline{N}}$ used are 32, 50, 23, and 128, respectively.

To calculate the H_{nb}, the local densities must be inferred from the beads' positions. A commonly used "particle-to-mesh" technique is applied: we split the simulation box into M number of cubic grid cells and estimate the densities of the species in these grids. The grid discretization length is defined as ΔL, and its value is fixed as $0.16Re$ in this work. The implementation details are discussed by Detcheverry et al.[22] Density fields of α species in a grid cell p are defined as

$$\phi_\alpha(p) = \frac{R_e^3}{\Delta L^3 N\sqrt{\overline{N}}}\sum_{i=1}^{nN}\delta(r_i - r(p))\delta_{\alpha t(i),} \tag{3}$$

where summation runs over all beads and $\alpha \in \{A,B\}$. $t(i)$ denotes the species of bead i. The delta function here represents that each bead is assigned to its nearest grid cell and contributes to density $\phi_\alpha(p)$.

Using the TICG model, Monte Carlo (MC) simulations are performed under the NVT ensemble. Two types of MC moves are used: single-bead displacement and reptation of chain. The maximum bead displacement size is set as $0.8b$, where b is the mean squared bond length of an ideal chain of N beads. The maximum value used of reptated bead in a chain is 5. In this work, an MC cycle is defined with $nN+2n$ number of MC moves, where on average nN bead displacement moves and $2n$ reputation moves are performed. The dimension of the simulation box in the x and y directions is chosen to be $10L_0$, where L_0 is the natural periodicity of lamellae in the AB diblock system (L_0 is $1.6Re$). The box dimension in the z direction is $1L_0$. Periodic boundary conditions are applied in the x and y directions, whereas a hard wall condition is used in the z direction.

Defects are generated by introducing a spatially varying external field in the simulations. The field applies interactions to beads of type α at a given position $\mathbf{r}$, characterized by $\lambda_\alpha N(\mathbf{r})$, such that it can generate the desired defects. Figure 6(a) shows the illustration of an applied field for two-layer apart DD; the black and white represent the attractive field for A and B beads, respectively. We simplified the geometric factors in applied fields by implementing two-tone 2D fields varying in the x and y directions only. In the A-rich (white) region, $\lambda_A N(\mathbf{r})$ and $\lambda_B N(\mathbf{r})$ are set to be -5 (attractive) or 5 (repulsive), respectively, and vice versa for the B-rich (black) region. As shown later, following relaxation simulations leads to the formation of metastable defective morphology with three-dimensional variation in density fields. Hence, for DD defects, four variables dd_i are used to define the geometry of DD and to introduce a randomness in the shape. Keeping the external field for defects generation on, AB diblock copolymers are self-assembled over 1000 MC cycles, which are enough to generate the required defect. We then switch off the external field and let the system equilibrate for an additional 500 MC cycles; this relaxation step is chosen to be long enough to reach metastable defective structure while not too long as to annihilate these kinetically trapped defects. A total of 1000 dislocation defects with random center locations was

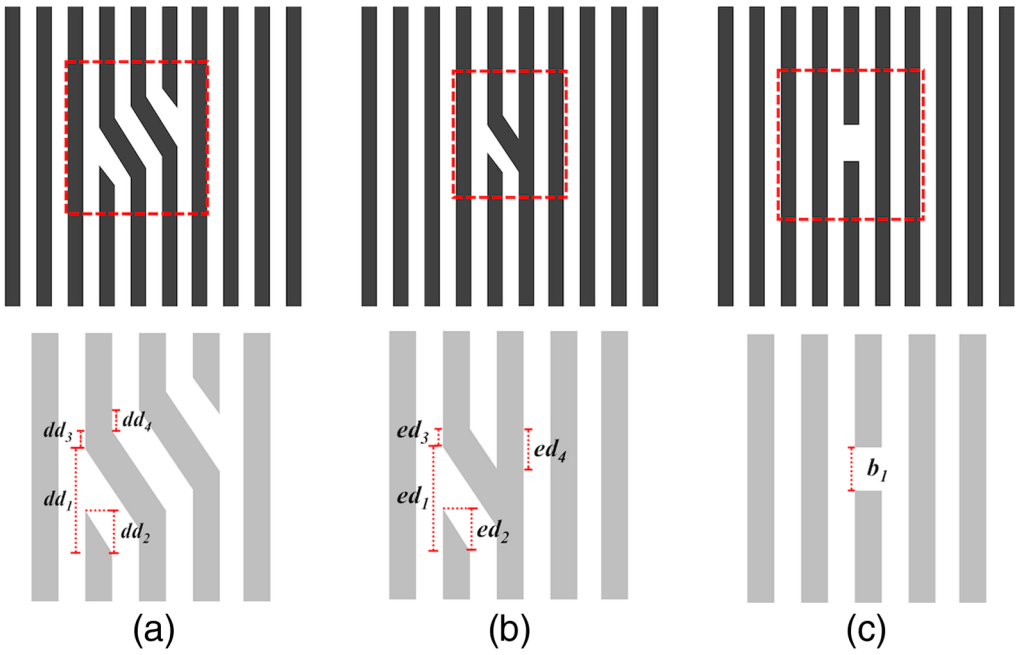

Fig. 6 Applied 2D external field along the *x* and *y* directions to generate defects through MC simulations (top) and zoomed views of defects (bottom) focusing on (a) DD, (b) ED, and (c) B defect types.

generated. For every image, values of geometry factors in the external field are randomly chosen; the values of dd_1 ($dd_2 = 0.05dd_1$), dd_3 and dd_4 are randomly chosen from 1.0 to 1.5, 0 to 0.06, and 0 to 0.06 (in the units of L_0), respectively.

One thousand ED defects are generated using a similar procedure [see Fig. 6(b) for the sketch of the applied external field with geometric variables ed_i, i from 1 to 4]. The values of ed_1 ($ed_2 = 0.05ed_1$ and $ed_4 = ed_1$) and ed_3 are randomly selected from ranges of 1.0 to 1.5 and 0 to 0.06 (in the units of L_0), respectively. Since ED has a lower kinetic energy barrier than DD, there are more quickly annihilated defects for the same period of relaxation simulation; these are not added to our simulation dataset while training the network.

Bridge (B) defects are generated by applying an external field shown in Fig. 6(c). Unlike DD or ED, all bridge defects annihilate right after switching off the external field. As discussed in detail in Delony et al.,[66] experiments and simulations have still not fully resolved the origin of the bridge defect. Therefore, we use a different procedure to stabilize the bridge defects. After the initial 1000 MC cycles, the external field is switched off only for regions where the bridge defect is not located, and field strength on bridge defect regions are weakened to $\lambda_\alpha N(\mathbf{r}) = 2$ or -2. A total of 1000 randomly placed bridge defects with varying sizes is generated using this procedure. The width of bridge b_1 in Fig. 6(c) is randomly selected from values ranging from 0.4 to $0.8L_0$. In addition, 1000 fully ordered morphologies are prepared using a defect-free external field.

The equilibrated configuration obtained is mapped into the normalized order parameter $S(p)$ at each simulation grid cell p:

$$S(p) = \frac{\phi_A(p) - \phi_B(p)}{\phi_A(p) + \phi_B(p)}, \tag{4}$$

where $\phi_A(p)$ and $\phi_B(p)$ are densities of A and B in a simulation grid cell. A grayscale image is generated using a linear mapping between $S(p)$ and the grayscale index. $S(p) = 1$ (A-rich) has a grayscale index of 1.0 (white), and $S(p) = -1$ (B-rich) corresponds to a grayscale index of 0 (black) with $S(p) = 0$ pointing to a grayscale index of 0.5. The field-off image is then annotated by drawing a bounding box around the defect. Note that the annotation procedure is automated as we already know the location of the defect from the applied external field. Annotated raw

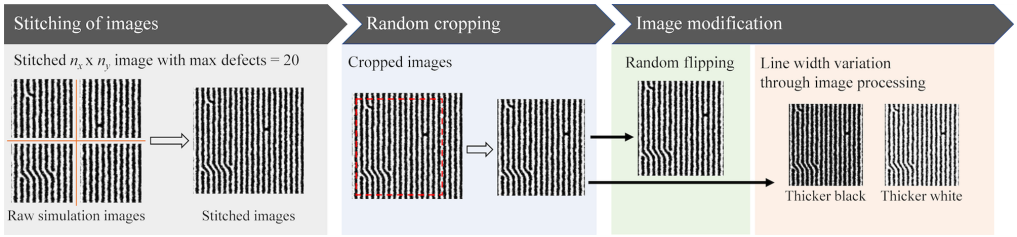

Fig. 7 Flowchart for the processing of images obtained from TICG simulations. Multiple defect-free and single-defect images are stitched together to generate a larger image that has many defects. Image with $n_x = 2$, $n_y = 2$, and 3 defects, obtained from combining four raw simulated images is shown. The image boundaries are then randomly cropped out while preserving the region where defects are located. The cropped image undergoes two different types of image modification, namely random flipping and line width variation.

grayscale images that are defect-free or with a single defect are postprocessed as described in Fig. 7 to generate a larger image that has many defects. Multiple images are stitched in a matrix fashion with a randomly chosen dimension of (n_x, n_y), where n_x is the number of images stitched horizontally and n_y is the number of images stitched in the vertical direction. The maximum value of n_x and n_y is 8. Every stitched image contains a random number of defects ranging from 0 to 20. For illustration, Fig. 7 shows an image with $n_x = 2$, $n_y = 2$ and 3 defects, obtained from combining four raw simulated images. The image boundaries are then randomly cropped out while preserving the region where defects are located. The cropped image undergoes two different types of image modification without perturbing the size of the image: random flipping and line width variation. Random flipping is performed by generating a random number in the interval 0 to 3, where 0, 1, 2, and 3 represent the operations of no-flipping, H-F, V-F, and HV-F. The width of black/white lines is adjusted by changing the relation between order parameter $S(p)$ and grayscale index. The modified $S(p)$ to grayscale index map now has the values of 0, 0.5, and 1 corresponding to $S(p)$ values of -1, f, and 1, where f is a randomly chosen value from an interval $(-0.6, 0.6)$. When the value of f is negative, black lines are thinner, and when it is positive, black lines are thicker (see Fig. 7). The processed simulated images are made ready for network training by rescaling them to an input layer size of 416×416 pixels while keeping the aspect ratio the same.

5 Results

We trained and evaluated the performance of the network on detecting and classifying defects in experimentally obtained SEM images. A total of 575 raw SEM images before augmentation through flipping was collected and annotated. As our study targets running the network with a small number of images available for a training set, we used only 100 SEM images for the training set (SEMD100) and 475 SEM images for the test set; SEM images for training were randomly selected from the entire dataset and had 144 B, 184 DP, 22 ED, and 27 DD with a total of 377 defects. The data augmentation as described in Fig. 4 was performed on the training dataset. By performing the three flipping operations on each image while retaining the original image, the augmented SEM dataset (Aug-SEMD100) has four times the number of images (400) as the SEM dataset (SEMD100). During the training, a batch size of 32 and learning rate with a cosine decay are used. For the first ~300 batch iterations, the learning rate was ramped up to a value of 1×10^{-3}. Then a cosine function was used to decay the learning rate to 1×10^{-8} in the next ~4200 batch iterations. The width and height of nine prior bounding boxes/anchor boxes (three boxes for each detection scale) are taken from the pretrained YOLOv3 network used on the COCO dataset;[18] specifically the values in pixels of (width, height) of anchor boxes arranged in ascending order of their areas are (10, 13), (16, 30), (33, 23), (30, 61), (62, 45), (59, 119), (116, 90), (156, 198), and (373, 326). YOLOv3, when it is used for more than 80 different classes, uses an architecture shown in Fig. 2 with F and R values of 32 and 2, respectively, and it demands close to 60 million trainable parameters. Aiming to reduce the number of trainable parameters for

our system with only four objects to be classified, the network is trained and tested for different R and F. The average precision (AP) for each defect and the mean of all Aps, termed the mean average precision (mAP), were used as metrics to evaluate the performance of our network. The AP is derived from the quantities' precision and recall; details of the equations to calculate the AP and mAP are provided by Everingham et al.,[67] where precision and recall are defined as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{5}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{6}$$

where TP (true positive) represents the number of correctly identified defects, FN (false negative) is the number of defects that the network is unable to detect, and FP (false positive) is the number of defects incorrectly predicted. If the value of IoU is >40%, where IoU is calculated between detected bounding box and ground truth bounding box, the prediction is TP; otherwise, it is classified as FP.

Figure 8(a) shows the AP for each defect and mAP averaged over five independent training runs starting with different random seeds, where R is set to its minimum possible value of 1 and F value is varied from 2 to 16. For comparison, we also show the individual APs and mAP obtained from the network trained on the dataset without any augmentation (SEMD100). For all filter values, augmentation through flipping significantly improves the APs of each defect type (e.g., 64.6 using Aug-SEMD100 as compared with 39.2 using SEMD100 for $F = 12$). In

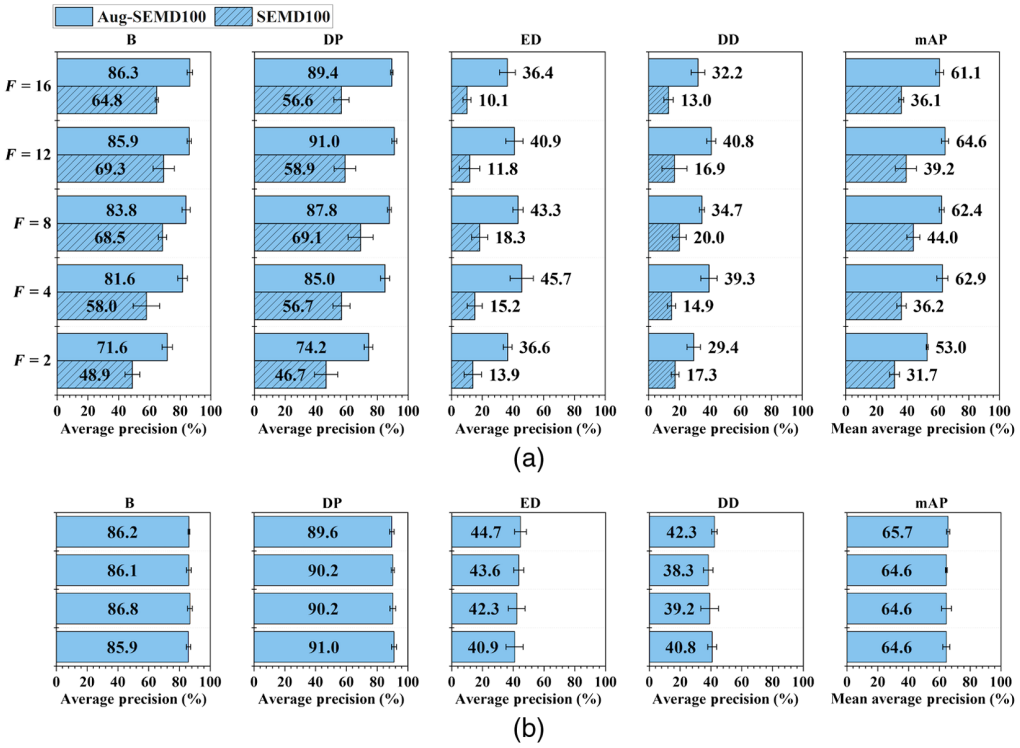

Fig. 8 Performance indices of the networks trained with SEMD100 and Aug-SEMD100 (augmented with flipping operations) datasets. (a) AP for each defect and mAP when the network architecture is varied with the minimum number of convolutional layer filters F of 2, 4, 8, 12, and 16 while fixing the minimum number of residual blocks R at 1. Two training datasets of SEMD100 consist of 100 images out of the available 575 SEM images acquired from experiments and Aug-SEMD100 in which SEM100 is augmented with flipping operations into 400 images. (b) The same performance metrics shown for the network with R values of 1, 2, 3, and 4 and with F fixed at 12. The networks were trained with Aug-SEM100.

addition to the fourfold increase in the number of defects in the augmented dataset, flipping also increases the randomness in the bounding box center location of the target defect, which effectively prevents the model from overfitting to a specific localized spatial position. However, even with the use of Aug-SEMD100, AP values for ED and DD are much lower than those of B and DP due to the lower number of these defects (88 ED and 108 DD defects as compared with 576 B and 736 DP defects). Henceforth, we use the B and DP AP values as a metric to select the optimal value of F. AP values of B and DP increased with F until 12 and then plateaued around 85% and 90%, respectively. Thus the optimal value for F is 12 in our case. Fixing F at 12 as we increase R increases the number of convolutional layers in the feature extractor unit (28 layers for $R = 1$ versus 96 for $R = 4$). But as shown in Fig. 8(b), the APs of B and DP stay around the values obtained for $R = 1$. Therefore, $F = 12$ with the minimum possible value of $R = 1$ was chosen as our optimal combination. Compared with the original implementation of YOLOv3, the number of trainable parameters for our network is close to 6 million, which is an order of magnitude ($\sim$10 times) smaller than the original implementation of YOLOv3. Also using $R = 1$, the number of convolutional layers in the feature extractor unit of our network is 28 as compared with 48 in the original YOLOv3.

For $F = 12$, we further explored various activation functions and different loss functions used for training the network, while using APs for B and DP as our performance indices. We started with two commonly used assortments for objectness loss (which optimizes the probability that a given grid cell has an object or not: probability value either 0 or 1): binary CE and FCE loss. The network with FCE loss resulted in better performance than with CE loss as shown in Fig. 9 with AP values for B and DP using FCE of 86% and 91%, respectively, as compared with 82% and 85%, respectively, for CE. As mentioned in Sec. 3, for every image, our multiscale network generates a total of $O(10^4)$ $(13 \times 13 \times 3 + 26 \times 26 \times 3 + 52 \times 52 \times 3 = 10{,}647)$ bounding boxes per image. Most of these boxes are marked on the background (straight lamellae lines) since our dataset has an order of 10 defects per image at maximum while the remaining area is background. When CE loss is implemented, a significant portion of objectness loss may be attributed to the background region, whereas FCE loss ensures that objectness loss is dominated by the few true objects in the network; therefore, FCE loss excels over CE loss in our case.

Complete intersection over union (CIoU) loss and use of the Mish activation function are shown to have a better convergence and accuracy than a network with GIoU loss and the Leaky ReLU activation function (for their definition and implementation details, see Refs. 68 and 69). However, for our system, Fig. 9 shows that there is no significant gain in the APs for B and DP in modifying our network to use the Mish activation function and CIoU loss. The APs of B and DP for GIoU versus CIoU and Mish versus Leaky ReLu are within 1% of each other. Here it is worth noting that that the AP of ED increased with the use of the Mish activation function, but the fluctuations (error bars) in its value are too large to claim any significant proclamations.

Although the use of a combination of GIoU loss, FCE loss, and Leaky ReLU activation function with $F = 12$ increases the performance of the network, the mAP of our network is

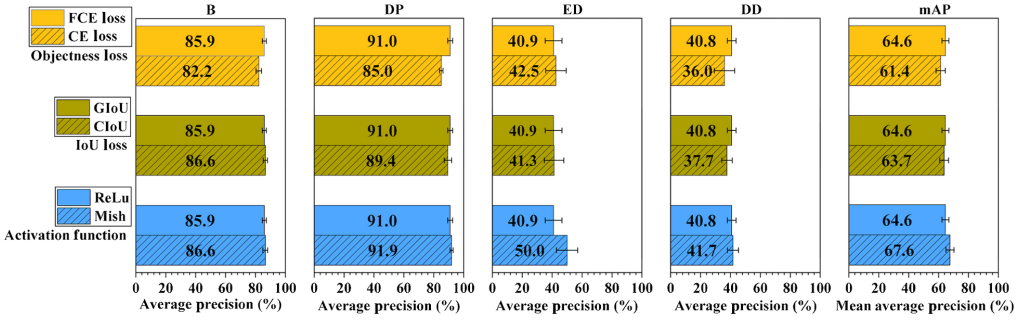

Fig. 9 AP for each defect and mAP of the network when different varieties of objectness loss, bounding box loss, and activation functions are used. The training dataset for all cases is the Aug-SEMD100 dataset. On the top, two types of objectness loss are compared: FCE and CE loss. In the middle, bounding box losses of GIoU and CIoU loss are tested. On the bottom, results of using different non-linear activation functions of Leaky ReLu and Mish are shown.

still close to 65%, due to the limited numbers of SEM images available even after flipping operations. To address the issue, we performed a further data augmentation in which we added images generated using molecular simulations to the training dataset. The time-consuming process of annotating objects in experimental data is bypassed when using simulated images with automated labeling of defects. Simulated datasets can be prepared as large as needed, as each simulation takes only an O(10) min for completion. Given the flexibility of customizing the size of simulated defects through random cropping (see Fig. 7) and rescaling cropped images to the 416×416 sized input layer without modifying the aspect ratio, we prepared two different simulated datasets: (1) the "distribution match" (DM) dataset, which has a similar width (W) and height (H) distribution as the SEM dataset, and (2) the "distribution mismatch" (DMM) dataset in which the defect size distribution of the simulated dataset is mismatched from the SEM dataset. For DM dataset preparation, we randomly select an image from the prepared simulated dataset and measure its diagonal length in units of pixels (square root of $W^2 + H^2$). Depending on the defect's size diagonally, it is sorted into evenly divided bins at a resolution of 7 pixels. The above operation is iteratively performed until the binned histogram of selected simulated images matches that of the SEM dataset. Performing this procedure of matching the diagonal length of the bounding boxes gives W and H defect distributions that have a strong overlap with the SEM dataset as shown in Figs. 10(a) and 10(b). For the DMM dataset, we only choose images with very large defects, with W and H values that are approximately above 100 pixels; this results in a distribution with only a small overlap with the SEM dataset distribution [see Figs. 10(a) and 10(b)]. S number of simulated defects are randomly selected from either DM or DMM datasets and mixed with Aug-SEMD100 (1508 total defects) for training; our test set remains unchanged. Keeping the remaining conditions and parameters identical, training

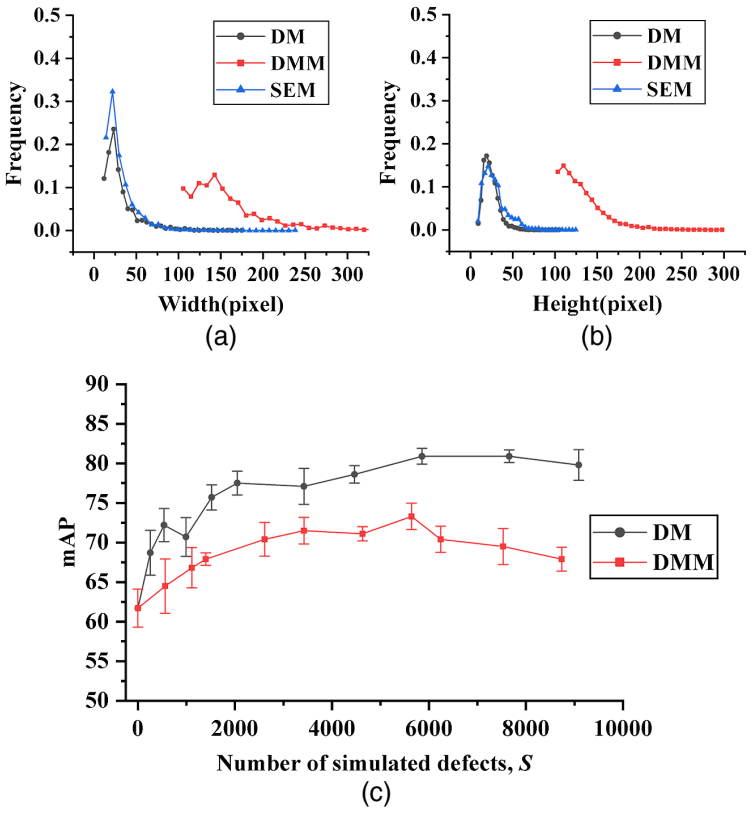

Fig. 10 (a) Width and (b) height distributions of bounding boxes in a DM dataset of simulated images containing defects that match the SEM dataset distribution, and DMM dataset of simulated image defects that mismatch the SEM dataset distribution. (c) Performance (mAP) variation as the network is trained with mixed datasets obtained by varying mixing ratios of defects in Aug-SEMD100 with simulated defects, from both DM and DMM. x axis marks the total number of defects in simulated images that are added to the Aug-SEMD100 dataset.

and testing of the network are performed; Fig. 10(c) shows the obtained mAP versus S. Even mixing with images from the DMM dataset with mismatched defect size, mAP increases on increasing S; the maximum value of obtained mAP is around 73%. Nevertheless, for S values beyond 6000, mAP starts to diverge since the network trained with large defect sizes overfits the bounding boxes with minimal overlap with the SEM dataset. With mixing of images from the DM dataset, mAP also increases. As the defects' sizes of the DM dataset, overlap with those of defects in the SEM images, a further increase in the value of S does not hamper the performance of the network as much as is seen with the DMM dataset; the maximum mAP is around 81%, which is 8% higher than the maximum value obtained by mixing images from DMM. Therefore, the trends observed using the DM and DMM datasets show that it is important to have a simulation dataset with defect size distributions that overlap with the SEM dataset when the training data set covers a vast range of samples.

When the network is trained with a mixture of ≈6000 simulated defects from both the DM and Aug-SEMD100 datasets, a mAP value of 81% is achieved. The mAP is significantly improved using the data augmentation strategies of both flipping and mixing with simulation images (from 39% to 81% using both strategies). The individual AP bar graph for the network is shown in Fig. 11(a). The individual AP values for B and DP are above 90%, whereas those of ED and DD are around 70%. As mentioned, the number of B (144) and DP (184) defects in our non-augmented SEMD100 dataset is an order of magnitude larger than ED (22) and DD (27) resulting in a better trained network with larger AP values; still, APs of ≈70% for both ED and DD with only 22 and 27 SEM defects used for training is more than adequate, signifying the excellence of our network when trained through augmented data. The network's predictability in accurately locating and classifying the defects is portrayed in Fig. 11(b) with two representative images in which all defects are detected and correctly classified. The confusion matrix, precision, recall, and F_1 score of the run resulting in the best mAP out of five independent runs are also shown in Table 1. The F_1 score is defined as the mean of precision and recall. The F_1 score averaged over all defect types has a high value of 84.5%. Similar to the trend observed in individual AP values, F_1 scores for B and DP are higher than for ED and DD. The diagonal entries of the confusion matrix represent the correctly classified defects, whereas the off-diagonal entries represent the fraction of defects that were classified incorrectly into another category. The small off-diagonal values of the confusion matrix for all defects show that our network performs the classification very effectively and feature differences among defects are sufficiently visible and distinct enough to avoid any significant classification errors.

To this point, we have shown the performance of the network trained with 100 SEM images out of a total of 575, which only accounts for ≈17% of the available dataset. We further investigate improvements in mAP using the data augmentation strategies for different numbers of original SEM images in the training dataset. SEMD200 and SEMD300 training datasets with 200 and 300 SEM images, respectively, were prepared. The SEMD100 dataset was appended to create the SEMD200 dataset by randomly extracting another 100 images from the 475 residual

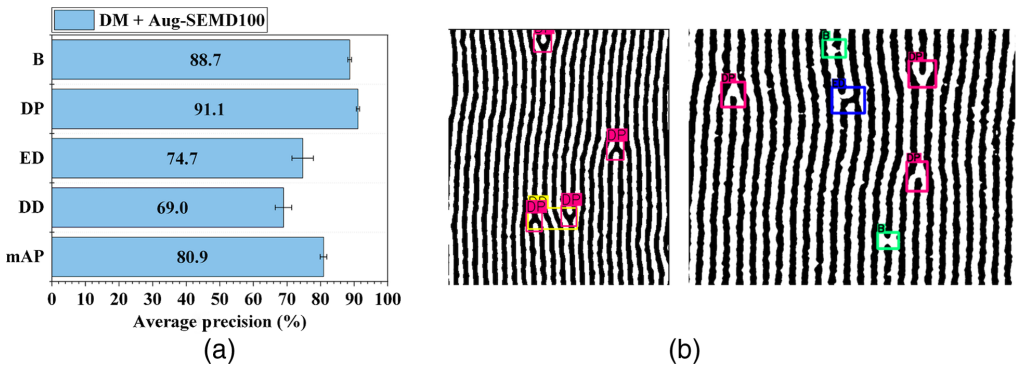

Fig. 11 (a) AP for each defect and mAP of the network trained with a mixture of ~6000 simulated defects from DM and Aug-SEMD100 datasets. (b) SEM images with bounding boxes predicted by the network. The predicted bounding boxes of B, DP, ED, and DD are marked in green, dark pink, blue, and yellow, respectively.

Table 1 Confusion matrix of the detected defects (in percentages) for the network trained with a mixture of ~6000 simulated defects in the DM dataset and Aug-SEMD100. Precision, recall, and F_1 score for each type of defect are also shown as the last three columns of the table.

		Predictable labels							
		B	DP	ED	DD	Total	Precision	Recall	F_1 score
True labels	B	**90.2**	*1.0*	*0.9*	*0.0*	684	90.2	90.2	90.2
	DP	*0.0*	**90.7**	*0.4*	*0.0*	788	90.2	90.7	90.5
	ED	*3.8*	*5.0*	**82.5**	*0.0*	80	74.2	82.5	78.3
	DD	*0.0*	*1.5*	*3.1*	**74.6**	130	83.6	74.6	79.1
		Total number of defects				1682	Average		84.5

images of the original dataset; an identical operation is performed to make SEMD300 from SEMD200. From the entire dataset, the operations to make SEMD200 and SEMD300 leave us with 275 remaining images for our test dataset. Even with 275 images, our test dataset has O(1000) defects, which is large enough to make statistically significant assertions. Figure 12 shows the enhancement in mAP through data augmentation versus the number of SEM images (M) for SEMD100, SEMD200, and SEMD300 datasets. To make a valid comparison among the obtained results for different M, a fix test dataset of 275 images as described earlier is used for all cases. Approximately 6000 simulated defects are segmented via the DM dataset. In all three cases, the data augmentation helps to increase mAP. However, the gain in mAP is more pronounced when the number of SEM images available for training is smaller. On both flipping and mixing with simulated images, the mAP increases about 40.9% for $M = 100$ compared with a gain of 29.7% for $M = 300$.

The capacity of our network can be broadened by implementing our network to a more enriched database of DSA with various processing conditions such as block copolymers, annealing temperature, types of substrate and guiding stripes, or line-space patterns from different lithography techniques such as EUV lithography. Such a diverse database is prepared by obtaining SEM images from selected publications[23–58] with defects belonging to one of the four categories classified in this work. Fifty-eight images were obtained with a total of 271defects in the prepared "Journal-SEM" database (JSEMD) of including 39 B, 151 DP, 10 ED, and 71 DD defects. (Note that images are processed to remove blurriness using a similar procedure as

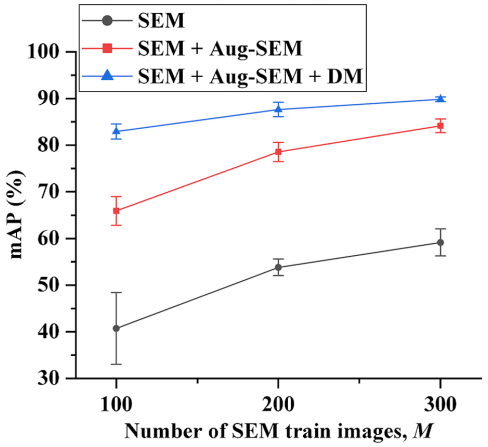

Fig. 12 Variation in mAP on changing the number of SEM images (M) selected to construct non-augmented training datasets. The mAP variation for networks trained using three different datasets is compared: (1) SEMD only, (2) SEMD is augmented with flipping (Aug-SEMD), and (3) SEMD is augmented with both flipping and mixing of simulation images (DM + Aug-SEMD).

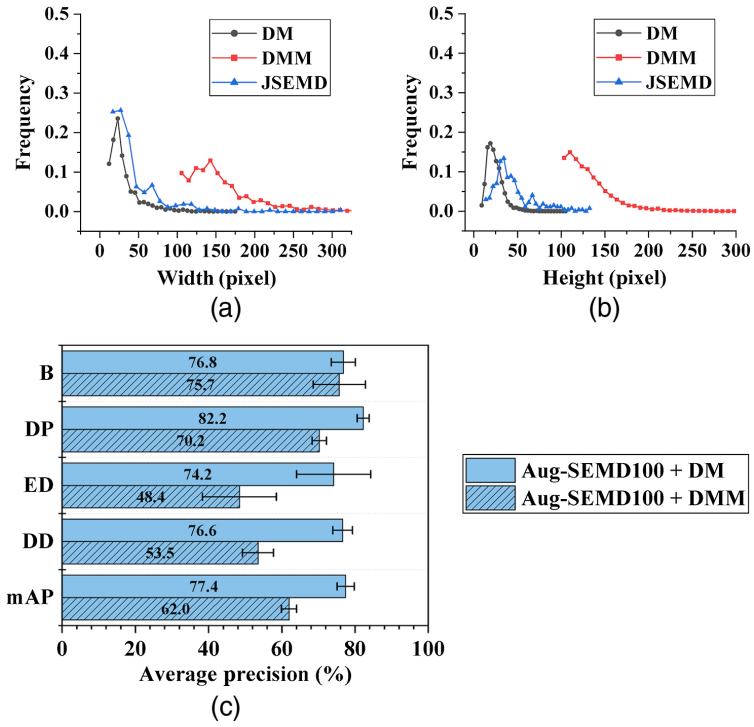

Fig. 13 (a) Width and (b) height distributions of defects from the database (JSEMD) from SEM images in selected publications. The corresponding distributions for defects obtained through simulations under DM and DMM datasets are also shown. (c) Using JSEMD as a test dataset, the performance (individual APs and mAP) of networks trained using the Aug-SEMD100 + DM and Aug-SEMD100 + DMM datasets.

described in Sec. 4.1.) A relevant consideration is whether our network trained using a dataset of SEM images under fixed experimental conditions is transferrable to perform the classification and detection of defects on the prepared diverse JSEMD dataset. Therefore, with JSEMD as our test dataset, we applied our network trained with the use of the Aug-SEMD100 dataset and ∼6000 defects from the DM dataset. The extent of overlap between the defect size distributions of the JSEMD and DM datasets is much more pronounced in comparison with the DMM dataset [see Figs. 13(a) and 13(b)], justifying our choice of using DM for training dataset augmentation. In Fig. 13(c), we compare mAP when the networks are trained on Aug-SEMD100 mixed with either DM or DMM datasets and tested on JSEMD. A high mAP value of 77% obtained when the DM dataset is used for augmentation shows the network's robustness toward different experimental designs targeting line-space patterns. However, a much smaller mAP value of 62% with the ≈6000 defect DMM dataset was obtained, highlighting the importance of overlap between size distributions of simulated defects and defects from SEM images. Our optimal network design and data augmentation strategies enable the network to have satisfactory transferability and to be generic enough to perform the L/S pattern defectivity analysis on data not constrained to fixed settings and unseen by the network.

6 Conclusion

We adopted YOLOv3, a well-known object detection/classification network for defect inspection of line-space patterns on block copolymer films. Although the architectural layout of the network is fixed to be the same as YOLOv3, the network variables such as filter size and number of residual blocks were chosen based on the convergence in the network's performance, represented by AP for individual defects and by mAP. Our optimized network has ∼6 million trainable parameters, which is an order of magnitude smaller than the original implementation of YOLOv3. The number of convolutional layers in the feature extractor unit of our network is

also reduced at 28, compared with 48 in the original YOLOv3. We found that FCE loss excels over CE loss whereas GIoU versus CIoU loss and Mish versus Leaky ReLu activation functions performed similarly.

Under the condition of a limited dataset, the training of the network is performed by inflating the data using two different data augmentation strategies. Strategies of flipping images and mixing of simulated images greatly enhanced the performance of our network. With the use of only 100 (M) images accounting for 17% of the SEM dataset for training, a mAP of ~81% was obtained using the augmentation strategies. Increasing the value of M further increased the mAP; mAP of 89% was observed for $M = 300$. However, the gain in mAP is higher when M is smaller; on both flipping and simulation mixing, the gain of 40.9% for $M = 100$ was obtained as compared with the gain of 29.7% for $M = 300$. The network trained with a simulated dataset with defect size distributions that overlap with the SEM dataset was shown to have a better performance than the dataset with a negligible overlap, and this highlighted the importance of selecting the optimal range of defect sizes present in simulated images. Although the experimental images need to be manually annotated, our data augmentation strategies bypass such a time-consuming process as (1) the defect locations in flipped images are mathematically derived from the original images and (2) defect locations in simulated images are already known as we precisely have the geometry of defect generating external fields while running the simulations.

The aforementioned results were obtained from an experimental database that is restricted to particular fixed processing conditions applied on cylinder forming block copolymers. However, the generalizability of our trained network was demonstrated by testing it on a more diverse dataset (JSEMD) prepared by gathering SEM images from selected publications. This dataset varies in many processing conditions including annealing temperature, photo/EUV lithography, and different types of substrate patterns to direct the line-space assembly. The network trained with the $N = 100$ augmented dataset was tested on JSEMD, and a high mAP value of 77% was obtained. This demonstrates the robustness of augmentation strategies, especially that of mixing simulation images, toward different experimental designs targeting line-space patterns.

As discussed, the use of these data augmentation strategies helps the most in increasing the performance of the network when the number of real images (non-augmented SEM images) is smaller. With only 300 real images in the training set, data augmentation strategies result in a network having almost 90% accuracy. This is impressive given that our network is not only doing the classification of multiple defects per image but also performing their detection, which involves estimating the location, width, and height of defects. Also it is notable that the number of trainable parameters of our network is considerably smaller than YOLOv3, making it faster to train and test. We believe that there are two ways of further increasing the accuracy of our network. One obvious way is to increase the number of real images in the training set. Another is to improve the quality of our simulation dataset that is mixed with the SEM dataset. One of the limitations of the simulation dataset is that the external fields having perfect line-space patterns (as shown in Fig. 6) generate defects that fail to replicate the line edge roughness (LER) and tilting of lines observed in the experimental images. The LER and tilting of lines can be reflected in the simulated images by modifying the values of simulation parameters, which are kept constant in this work. For example, different χN values can prepare a simulation dataset with images having different LER values. Instead of modifying simulation parameters, one could use the novel sampling strategy proposed by Ma et al.,[70] in which the fusion of SEM and simulation data based on transfer learning is performed using generative adversarial networks. In this approach, the information in an SEM image is transferred to the simulation image to generate a synthetic image of better quality, i.e., an image containing important features found in the real image. Our attempts to use this novel strategy are already underway and will be published in a future study.

In the future, our work will aim to extend the deep learning model in ways that also estimate the LER and line width roughness (LWR) along with defect classification and detection. Recently, work by Chaudhary et al.[71] used a group of neural networks to measure the LER and LWR of SEM images. The unification of their proposed network with our network will provide a more complete machine learning tool to assess the performance of lithographic processing conditions.

Acknowledgments

The authors would like to acknowledge the funding support by the National Research Foundation of Korea (NRF) grant funded by the Korea Government (MSIT) (Grant Nos. 2018R1A5A1025224 and 2018H1D3A1A01036430). The code used in this work is available at: https://github.com/Ahn-JH/YOLOv3_tf2. The written code is based on the code provided in the url, https://github.com/YunYang1994/TensorFlow2.0-Examples/tree/master/4-Object_Detection/YOLOV3. The authors declare no conflicts of interest.

References

1. B. Wu and A. Kumar, "Extreme ultraviolet lithography and three dimensional integrated circuit: a review," *Appl. Phys. Rev.* **1**(1), 011104 (2014).
2. S. O. Kim et al., "Epitaxial self-assembly of block copolymers on lithographically defined nanopatterned substrates," *Nature* **424**(6947), 411–414 (2003).
3. C.-C. Liu et al., "Chemical patterns for directed self-assembly of lamellae-forming block copolymers with density multiplication of features," *Macromolecules* **46**(4), 1415–1424 (2013).
4. D. V. den Heuvel et al., "Investigation of the performance of state-of-the-art defect inspection tools within EUV lithography," *Proc. SPIE* **8324**, 83240L (2012).
5. K. Sah et al., "Inspection of stochastic defects with broadband plasma optical systems for extreme ultraviolet (EUV) lithography," *IEEE Trans. Semicond. Manufact.* **33**(1), 23–31 (2020).
6. L. Xie et al., "A novel defect detection and identification method in optical inspection," *Neural Comput. Appl.* **24**(7–8), 1953–1962 (2014).
7. B. Zheng and G. X. Gu, "Machine learning-based detection of graphene defects with atomic precision," *Nano-Micro Lett.* **12**(1), 181 (2020).
8. D. Tabernik et al., "Segmentation-based deep-learning approach for surface-defect detection," *J. Intell. Manuf.* **31**(3), 759–776 (2020).
9. X. Yin et al., "A deep learning-based framework for an automated defect detection system for sewer pipes," *Autom. Constr.* **109**, 102967 (2020).
10. S. S. Kumar et al., "Deep learning-based automated detection of sewer defects in CCTV videos," *J. Comput. Civil Eng.* **34**(1), 04019047 (2020).
11. A. Koirala et al., "Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of 'MangoYOLO'," *Precis. Agric.* **20**(6), 1107–1135 (2019).
12. C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data* **6**(1), 60 (2019).
13. J. Shijie et al., "Research on data augmentation for image classification based on convolution neural networks," in *Chin. Autom. Congr. Cac*, pp. 4165–4170 (2017).
14. L. Taylor and G. Nitschke, "Improving deep learning with generic data augmentation," in *IEEE Symp. Ser. Comput. Intell. Sci.*, pp. 1542–1547 (2018).
15. R. Carrasco-Davis et al., "Deep learning for image sequence classification of astronomical events," *Publ. Astron. Soc. Pac.* **131**(1004), 108006 (2019).
16. M. Cha et al., "Improving SAR automatic target recognition using simulated images under deep residual refinements," in *IEEE Int. Conf. Acoust. Speech Signal Process.*, pp. 2606–2610 (2018).
17. J. C. Holtzman et al., "Radar image simulation," *IEEE Trans. Geosci. Electron.* **16**(4), 296–303 (1978).
18. J. Redmon and A. Farhadi, "YOLOv3: an incremental improvement," https://arxiv.org/abs/1804.02767 (2018).
19. Y. C. Kim et al., "Shear-solvo defect annihilation of diblock copolymer thin films over a large area," *Sci. Adv.* **5**(6), eeaw3974 (2019).
20. F. A. Detcheverry et al., "Monte Carlo simulation of coarse grain polymeric systems," *Phys. Rev. Lett.* **102**(19), 197801 (2009).
21. F. A. Detcheverry et al., "Theoretically informed coarse grain simulations of block copolymer melts: method and applications," *Soft Matter* **5**(24), 4858–4865 (2009).

22. F. A. Detcheverry et al., "Monte Carlo simulations of a coarse grain model for block copolymers and nanocomposites," *Macromolecules* **41**(13), 4989–5001 (2008).
23. L. D'Urzo et al., "A comprehensive approach for micro and multiple bridges mitigation in immersion photolithography," *Proc. SPIE* **9425**, 94251Y (2015).
24. T. Terasawa et al., "Actinic phase defect detection and printability analysis for patterned EUVL mask," *Proc. SPIE* **7636**, 763602 (2010).
25. P. A. R. Delgadillo et al., "All track directed self-assembly of block copolymers: process flow and origin of defects," *Proc. SPIE* **8323**, 83230D (2012).
26. K. Nakano et al., "Analysis and improvement of defectivity in immersion lithography," *Proc. SPIE* **6154**, 61544J (2006).
27. E. W. Edwards et al., "Binary blends of diblock copolymers as an effective route to multiple length scales in perfect directed self-assembly of diblock copolymer thin films," *J. Vac. Sci. Technol. B* **24**(1), 340 (2006).
28. E. Han, M. Kim, and P. Gopalan, "Chemical patterns from surface grafted resists for directed assembly of block copolymers," *ACS Nano* **6**(2), 1823–1829 (2012).
29. M. Somervell et al., "Comparison of directed self-assembly integrations," *Proc. SPIE* **8325**, 83250G (2012).
30. A. P. Marencic and R. A. Register, "Controlling order in block copolymer thin films for nanopatterning applications," *Annu. Rev. Chem. Biomol. Eng.* **1**(1), 277–297 (2010).
31. H. Pathangi et al., "Defect mitigation and root cause studies in 14 nm half-pitch chemo-epitaxy directed self-assembly LiNe flow," *J. Micro Nanolithogr. MEMS MOEMS* **14**(3), 031204 (2015).
32. R. Gronheid et al., "Defect reduction and defect stability in IMEC's 14 nm half-pitch chemo-epitaxy DSA flow," *Proc. SPIE* **9049**, 904905 (2014).
33. P. R. Delgadillo et al., "Defect source analysis of directed self-assembly process," *J. Micro Nanolithogr. MEMS MOEMS* **12**(3), 031112 (2013).
34. S. O. Kim et al., "Defect structure in thin films of a lamellar block copolymer self-assembled on neutral homogeneous and chemically nanopatterned surfaces," *Macromolecules* **39**(16), 5466–5470 (2006).
35. T.-H. Chang et al., "Directed self-assembly of block copolymer films on atomically-thin graphene chemical patterns," *Sci. Rep.* **6**(1), 31407 (2016).
36. W. Li and M. Müller, "Directed self-assembly of block copolymers by chemical or topographical guiding patterns: Optimizing molecular architecture, thin-film properties, and kinetics," *Prog. Polym. Sci.* **54**, 47–75 (2016).
37. M. J. Maher et al., "Directed self-assembly of silicon-containing block copolymer thin films," *ACS Appl. Mater. Inter.* **7**(5), 3323–3328 (2015).
38. G. Wu et al., "Directed self-assembly of hierarchical supramolecular block copolymer thin films on chemical patterns," *Adv. Mater. Interfaces* **3**(13), 1600048 (2016).
39. R. P. Kingsborough et al., "Electron-beam directed materials assembly," *Proc. SPIE* **7637**, 76370N (2010).
40. G. Wu et al., "Directed self-assembly of hierarchical supramolecular block copolymer thin films on chemical patterns," *Adv. Mater. Interfaces* **3**(13), 1600048 (2016).
41. G. S. W. Craig and P. F. Nealey, "Exploring the manufacturability of using block copolymers as resist materials in conjunction with advanced lithographic tools," *J. Vac. Sci. Technol. B* **25**(6), 1969 (2007).
42. U. Nagpal et al., "Free energy of defects in ordered assemblies of block copolymer domains," *ACS Macro Lett.* **1**(3), 418–422 (2012).
43. K. Nakano et al., "Immersion defectivity study with volume production immersion lithography tool for 45 nm node and below," *Proc. SPIE* **6924**, 692418 (2008).
44. G. Liu et al., "Integration of density multiplication in the formation of device-oriented structures by directed assembly of block copolymer–homopolymer blends," *Adv. Funct. Mater.* **20**(8), 1251–1257 (2010).
45. R. P. Kingsborough et al., "Lithographically directed materials assembly," *Proc. SPIE* **7271**, 72712D (2009).
46. M. Harumoto et al., "LWR and defectivity improvement on EUV track system," *Proc. SPIE* **9776**, 977628 (2016).

47. E. W. Edwards et al., "Mechanism and kinetics of ordering in diblock copolymer thin films on chemically nanopatterned substrates," *J. Polym. Sci., Part B* **43**(23), 3444–3459 (2005).
48. S.-M. Hur et al., "Molecular pathways for defect annihilation in directed self-assembly," *Proc. Natl. Acad. Sci. U. S. A.* **112**(46), 14144–14149 (2015).
49. M. Muramatsu et al., "Pattern defect reduction and LER improvement of chemo-epitaxy DSA process," *Proc. SPIE* **10144**, 101440Q (2017).
50. R. Tiron et al., "Pattern density multiplication by direct self assembly of block copolymers: toward 300 mm CMOS requirements," *Proc. SPIE* **8323**, 83230O (2012).
51. T. Segal-Peretz et al., "Quantitative three-dimensional characterization of block copolymer directed self-assembly on combined chemical and topographical prepatterned templates," *ACS Nano* **11**(2), 1307–1319 (2017).
52. A. M. Welander et al., "Rapid directed assembly of block copolymer films at elevated temperatures," *Macromolecules* **41**(8), 2759–2761 (2008).
53. A. Stein et al., "Selective directed self-assembly of coexisting morphologies using block copolymer blends," *Nat. Commun.* **7**(1), 12366 (2016).
54. J. Doise et al., "Strategies for increasing the rate of defect annihilation in the directed self-assembly of high-χ block copolymers," *ACS Appl. Mater. Inter.* **11**(51), 48419–48427 (2019).
55. J. Y. Cheng et al., "Templated self-assembly of block copolymers: top-down helps bottom-up," *Adv. Mater.* **18**(19), 2505–2521 (2006).
56. L. Wan et al., "The limits of lamellae-forming PS-b-PMMA block copolymers for lithography," *ACS Nano* **9**(7), 7506–7514 (2015).
57. L. D. Williamson et al., "Three-tone chemical patterns for block copolymer directed self-assembly," *ACS Appl. Mater. Inter.* **8**(4), 2704–2712 (2016).
58. C.-C. Liu et al., "Towards electrical testable SOI devices using directed self-assembly for fin formation," *Proc. SPIE* **9049**, 904909 (2014).
59. J. Choi et al., "Gaussian YOLOv3: an accurate and fast object detector using localization uncertainty for autonomous driving," in *IEEE/CVF Int. Conf. Comput. Vision*, pp. 502–511 (2019).
60. P. Zhang, Y. Zhong, and X. Li, "SlimYOLOv3: narrower, faster and better for real-time UAV applications," in *IEEE/ CVF Int. Conf. Comput. Vision*, pp. 37–45 (2019).
61. M. A. Al-Masni et al., "Simultaneous detection and classification of breast masses in digital mammograms via a deep learning YOLO-based CAD system," *Comput. Meth. Prog. Bio.* **157**, 85–94 (2018).
62. W. Xu and S. Matzner, "Underwater fish detection using deep learning for water power applications," in *Int. Conf. Comput. Sci. Comput. Intell.*, pp. 313–318 (2018).
63. H. Rezatofighi et al., "Generalized intersection over union: a metric and a loss for bounding box regression," https://arxiv.org/abs/1902.09630 (2019).
64. T.-Y. Lin et al., "Focal loss for dense object detection," https://arxiv.org/abs/1708.02002 (2017).
65. J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," https://arxiv.org/abs/1705.02950 (2017).
66. J. B. Delony, P. J. Ludovice, and C. L. Henderson, "Understanding and mitigating bridge defects in block copolymer directed self-assembly through computational materials design and optimization," *Proc. SPIE* **11326**, 113261K (2020).
67. M. Everingham et al., "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vision* **88**(2), 303–338 (2010).
68. Z. Zheng et al., "Distance-IoU loss: faster and better learning for bounding box regression," https://arxiv.org/abs/1911.08287 (2019).
69. D. Misra, "Mish: a self regularized non-monotonic activation function," https://arxiv.org/abs/1908.08681 (2019).
70. B. Ma et al., "Data augmentation in microscopic images for material data mining," *npj Comput. Mater.* **6**(1), 125 (2020).
71. N. Chaudhary, S. A. Savari, and S. S. Yeddulapalli, "Line roughness estimation and Poisson denoising in scanning electron microscope images using deep learning," *J. Micro Nanolithogr. MEMS MOEMS* **18**(2), 024001 (2019).

Jihun Ahn completed his BS degree in School of Polymer Science and Engineering at Chonnam National University. He is currently a MS degree student in Department of Polymer Engineering at Chonnam National University. His research at Chonnam National University focus on characterizing the phases and analyzing defects observed in directed self-assembly of block copolymer using deep learning.

Ye Chan Kim completed his BS degree in chemical engineering at Ulsan National Institute of Science and Technology (UNIST). He started his combined MS and PhD course at UNIST in 2015. After he received his PhD in 2021, he has been working as a postdoctoral researcher at Seoul National University. During the PhD study, his research focus on the control of block copolymer nanostructures in sequential reordering processes, and the analysis of their structural order.

So Youn Kim is an assistant professor of the School of Chemical and Biological Engineering at Seoul National University (SNU). She received her PhD at the University of Illinois at Urbana-Champaign in 2011. Before joining SNU, she was assistant/associate professor from 2014 to 2021 at Ulsan National Institute of Science and Technology. Her research interest lies in understanding fundamental interactions in soft materials from the molecular level and relating to the bulk state properties.

Su-Mi Hur obtained her PhD from Chemical Engineering Department at University of California, Santa Barbara. She continued her research career as postdoctoral associate in Institute for Molecular Engineering at University of Chicago and Argonne National Laboratory. In 2015, she joined polymer science and engineering department at Chonnam National University, Korea. Her research expertise lies in applying statistical mechanical theory and field-/particle-based coarse grained simulations to investigate structural, thermodynamic, and dynamic phenomena in polymer-based soft materials.

Vikram Thapar studied chemical engineering at the Indian Institute of Technology, Mumbai, India. He received his PhD in the field of computer simulations of self-assembly of anisotropic nanoparticles from Cornell University, Ithaca, New York, USA. After finishing his PhD, he worked as a postdoctoral fellow in collaboration with IMEC, Leuven, Belgium, and the University of Chicago. Since 2018, he has been working at Chonnam National University, Gwangju, South Korea, and focused on understanding the self-assembly of polymers using molecular simulations.