A Meta-Learning Reinforcement Training Method for Machine Learning Image-To-Image Optical Proximity Correction

Albert Lin, Yen-Wei Feng, Hung-Tse Hsu, Han-Chun Tung, Ching-Hsuan Huang, Hsueh-Li Liu, and Peichen Yu

***Abstract*—**As the scaling down of semiconductor manufacturing nodes, optical proximity correction (OPC) has become more and more crucial, where the OPC using machine learning to construct models, or so-called machine learning optical proximity correction (MLOPC), is flourishing. The forward and reverse networks in MLOPC are two deciding parts. Training the image-to-image OPC dataset using various advanced ML models can lead to less fitted results. The tests on diverse mainstream image-to-image ML models, including Unet, Vnet, ResUnet, U2net, and attention Unet, are conducted in our studies. We propose a meta-learning method based on reinforcement learning (RL) with the forward and reverse networks cascaded to boost the training efficiency and accuracy for MLOPC. The meta-learning is performed so that the selected setting of the next training step is based on the Q-model in RL, and the baseline training method is the direct utilization of adaptive moment estimation (Adam). In the setting of the cascaded network with six different models and three types of datasets, the model using the meta-learning approach takes advantage of the cascade and RL-controlled training path to boost the model accuracy, and the results using meta-learning surpass the baseline results in most of the cases, especially the instances with highly complex data or network architecture. Among them, the networks trained via the proposed meta-learning RL method receive the averaged increments of 19.5%, 1.2%, and 4.8% in the forward models and the averaged increments of 12%, 3.1%, and 13% in the reverse models.

I. INTRODUCTION

Driven by the demand for higher performance, shrinking technology nodes is essential, but they have encountered various challenges nowadays. Photolithography is pivotal in manufacturing semiconductors by transferring photomask patterns onto silicon wafers, achieving the desired chip designs. However, the reduction of feature sizes causes the diffraction limit of light, increasing the complexity of printing precise patterns, such as line end pinching, line bridging, and corner rounding. Optical Proximity Correction (OPC) has been one of the most popular resolution enhancement techniques (RETs) to correct the variations in image intensity caused by diffraction and other optical effects, optimize mask printability, and improve the edge-of-corner placement accuracy of pattern features.

Over the years, different OPC methods have been proposed to address the inevitable geometric distortion by shrunk feature sizes. The conventional OPC methods can be categorized as rule-based OPC, model-based OPC, and inverse lithography technology-based (ILT) OPC. Rule-based OPC utilizes the pre-established rules, and the OPC process can be performed quickly, but this method is typically limited to non-systematic effects before 90 nm nodes [1, 2]. Model-based OPC is a more sophisticated approach with physical or empirical model parameters to predict the behavior of light during the lithography process. The optimized mask layouts were achieved using simulation models and iteratively adjusting the feature shapes and sizes [3-6].

The reverse engineering (RE) technique enables the analysis and dissection of existing products or systems to understand their design, structure, and function [7, 8]. Its application has proven significant across various industries. In semiconductor manufacturing, inverse lithography technology (ILT) is critical in reverse engineering operations. Conventionally, by developing a model for reverse mapping in ILT, optimized mask shapes can be obtained using pixel-wise optimization techniques without predefined rules or models while solving the OPC problem as an inverse problem. This approach enables finding inverse solutions for lithography that cover resist development and etching processes in addition to optics [9-12]. However, ILT requires significant computational resources due to the complex optimization algorithms involved. Since every pixel must be calculated in conventional ILT optimization processes, especially for complete IC design layouts or high-resolution patterns, a large number of iterations can be time-consuming.

MLOPC is a novel OPC technique that has emerged with machine learning networks. Instead of relying on manual rule definition, model calibration, and time-consuming iterative computing, it involves training a machine learning model to predict OPC solutions for new patterns based on input dataset features. Various techniques have been proposed in this regard. Gu et al. developed a linear regression model to modify the mask with fewer required iteration times than the original model-based OPC [13]. Jia et al. applied the stochastic gradient descent method to inverse mask synthesis as an MLOPC problem [14]. Luo generated the optimal mask pixel values using a multilayer perceptron (MLP) neural network (NN) [15]. With the evolution of deep learning, Ma et al. [16] and Cecil et al. [17] proposed a convolutional neural network (CNN) with an inverse lithography ML model, which boosted iteration efficiency and improved the quality of the optimized mask output. Borisov et al. presented a CNN network with small training parameters for lithography hotspot detection, which would consume less computation time and maintain a high level of

accuracy [18]. Yang et al. replaced shallow learning models with deep convolutional neural networks without a pooling process, detecting the hotspot with better accuracy and fewer false alarm penalties [19]. Yang et al. also extracted high-dimensional features using CNN with biased learning algorithms. The results showed a 5.9% improvement in hotspot detection for large-scale layouts [20]. Tung et al. showed that the modified attention layer embedded in Unet structure could achieve higher accuracy with self-attention mechanism [21]. Shao et al. used CNN models to build two pre-trained models, LithoNet and OPCNet. The first one was responsible for predicting masks pattern to Scanning Electron Microscope (SEM) images, and the other one predicted the optimized mask from LithoNet's output SEM images. They were connected together to achieve a fully data-driven framework of OPC [22]. U-Net, one of the robust Deep Neural Networks (DNNs), was initially proposed for the biomedical image segmentation problem [23]. More and more OPC research has adopted U-Net due to its flexibility and simplicity, especially when training with limited sample data [24]. Yu et al. [25] compared single-channel U-Net (SCU-Net) and multi-channel U-Net (MCU-Net) and demonstrated that MCU-Net can improve accuracy and significantly reduce computation time. Cao et al. combined GAN (Generative Adversarial Network) and OPC with the U-Net architecture, increasing speed and lithographic performance [26]. Furthermore, to further reduce GPU computations, Xiao et al. proposed a novel OPC solution by concatenating a forward and reverse model composed of U-Net, allowing for the direct transfer of the original mask to the optimized mask within a single model [27]. Tejender et al. combined OPC and etch proximity effects (EPC) with MLP neural networks and dimensionality reduction (DR) algorithms, overcoming the problem of the lack of physics EPC model and reducing the long computation time [28]. These approaches are expected to enhance lithography printability and accuracy in the field of semiconductor manufacturing.

The semiconductor industry faces challenges in deep learning applications due to the limited availability of relevant datasets, specifically in the context of the "few-shot learning problem." Meta-learning, also known as "learn to learn," has emerged as a promising approach to address the "few-shot learning problem" arising from the scarcity of data samples. Meta-learning involves two concepts: the base learner learns the task, while the meta-learner learns how to optimize the training of the base learner. The meta-learner can be implemented using evolutionary algorithms or supervised learning algorithms. With the optimized meta-learner, the learning strategies of the base learners can be improved. Several meta-learning frameworks have been proposed for various IC technology processes in recent years, showcasing promising results in resolving few-shot learning problems with limited data samples. Chen et al. combined meta-learning models with tree-based ensemble models and focused on virtual physical vapor deposition metrology to overcome the physical metrology delay of the wafer acceptance test [29]. Lee et al. applied a meta-learning approach, the Reptile algorithm [30], to resolve the few-shot regression problem of the sputtering process. The base learner adapted the variable fabrication parameters of titanium/nitride compounds, while the meta-learner optimized hyperparameters such as batch size, epochs, and others [31]. Tejender et al. proposed a meta-learned sampling method assisted by technology computer-aided design (TCAD), training the meta-learner to adjust the way of sampling [32]. The work here aims to contribute significantly to advancing the OPC within the Industry 4.0 framework by employing novel network architectures and innovative reverse engineering techniques integrated with meta-learning.

II. Method

A. Dataset

From the previous work [33], the gds.py module 1.6.13 [34] of Python 3.9.5 [35] is employed to generate these patterns, which are then exported to K-layout 0.28.7 [36] for further processing. N-type silicon wafers are chosen as the substrate material, and KrF positive photoresist is utilized in the lithography process. The lithography system, Canon FPA-6300ES6a, operates at a DUV wavelength of 248 nm (resolution ≤90 nm). Subsequently, the developed mask layout is captured using the Hitachi SU-8010 Cold Cathode Field Emission SEM. A comprehensive dataset is typically prepared for model training in conventional machine learning training processes. However, in semiconductor manufacturing, producing a large number of samples before obtaining experimental results is time-consuming and resource-exhausted. The number of each one-bar pattern, two-bar pattern, tri-bar pattern, and polygonal rectangle pattern varies between about 90 to 210, along with their SEM images after the lithography process.

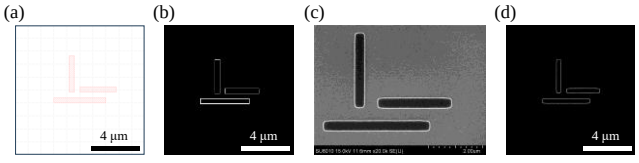

Fig. 1. Illustration of (a) K-layout mask pattern, (b) binary mask pattern, (c) lithography SEM image, (d) binary SEM image.

In the preprocessing stage of the training data, we first converted the pre-drawn K-Layout mask into a binary mask, as shown in Fig. 1, representing a tri-bar pattern. The edges were converted into white pixels (1), while the rest of the background was set to black pixels (0). The resulting binary mask had an image size of 256×256, with each pixel corresponding to approximately 24 nm.

The filled graph is more helpful for model training because the number of 0s and the number of 1s in the pixels of the original edge-only pictures differ too much, making it challenging for the model to learn effectively. Yet, the SEM images exhibited evitable noise in the backgrounds. Due to noise in the original SEM images, some regions' boundaries can become faint or disconnected after image processing. Filling these areas using an algorithm can lead to distortion, creating inaccuracies during data preprocessing.

To avoid propagating these errors and maintain accuracy, we utilized only the edge information in the final training data. Therefore, after image processing, the SEM photos were aligned onto a 256×256 black background and registered with the binary mask, resulting in the binary SEM image.

B. Algorithm

The forward and reverse models are both constructed using an Unet [23, 37] in Fig. 2. In order to test whether the meta-learner can adapt to various tasks, i.e., different types of networks, we replace the Unet with other variations of the Unet architecture. There were Attention Unet [37, 38] in Fig. 3, ResUnet-a [37, 39] in Fig. 4, R2Unet [37, 40] in Fig. 5, Self Attention Unet [37, 38] in Fig. 6, and Vnet [37, 41] in Fig. 7.

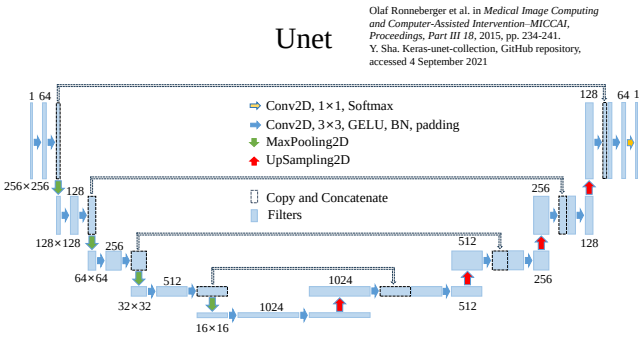

Fig. 2. Unet [23, 37]

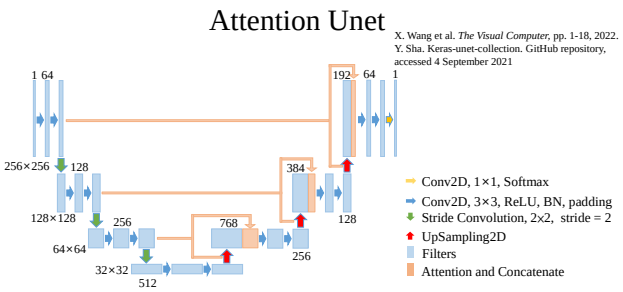

Fig. 3. Attention Unet [37, 38]

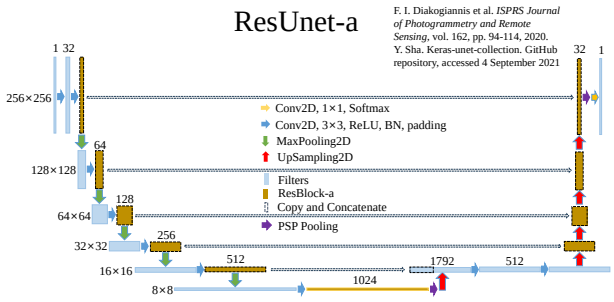

Fig. 4. ResUnet-a [37, 39]

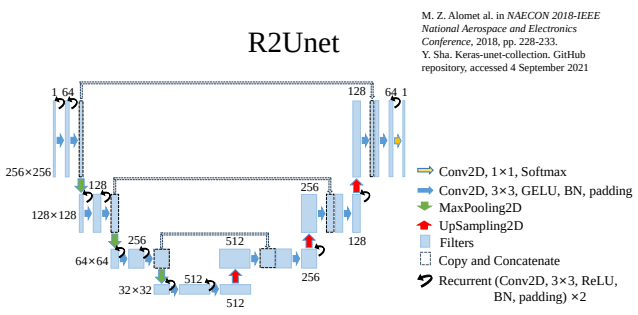

Fig. 5. R2Unet [37, 40]

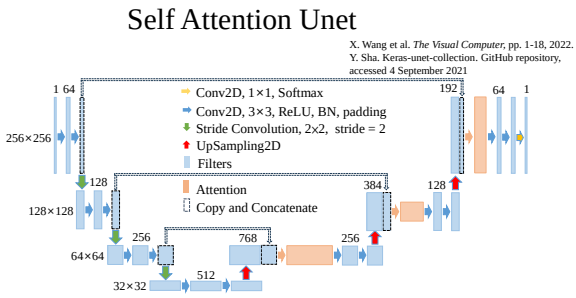

Fig. 6. Self Attention Unet [37, 38]

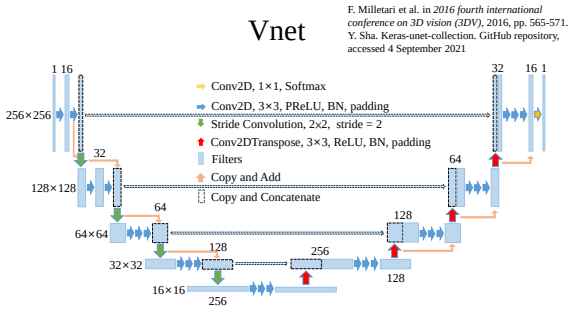

Fig. 7. Vnet [37, 41]

The baselines of the six networks in Fig. 2, Fig. 3, Fig. 4, Fig. 5, Fig. 6, and Fig. 7 are responsible for the primary synthesis and output of the dataset. We divide it into two parts: the forward model and the reverse model. In previous research, the focus has been on forward propagation, which means that when the input is a mask, the output generates SEM pattern images by (1).

$$Y = F(X : \vec{W}_f, \vec{b}_f) \tag{1}$$

Y is the predicted SEM image, X is the actual mask pattern, and W and b represent the weights and biases in the forward neural network. F expresses the network that converts the mask patterns into SEM images. Next, we aim to incorporate a reverse propagation model after this stage, which is intended to transfer the SEM patterns back into masks. R expresses the function that transfers the SEM images back to the mask pattern by (2).

$$X = R(Y : \vec{W}_b, \vec{b}_b) \tag{2}$$

X is the predicted mask pattern, Y is the actual SEM image, and W and b represent the weights and biases in the reverse neural network. Both forward and reverse models split 20% for validation data, and they are trained by their validation loss for 20000 epochs with the early stop patience of 30 epochs. The optimizer is Adam, with a 0.001 learning rate in the baseline. The validation loss is calculated by binary cross-entropy (BCE), and the test loss is decided by customized BCE loss, the BCE loss with edge enhancement process.

In this research, each forward and reverse model is composed of the same kind of network. The forward and reverse models above were connected as the primary model, as shown in (3). In our research, there were two types of models: one is a forward-to-reverse cascaded model involving the reverse model connected to the forward model, in which the input is mask layout, and the output is the predicted mask.

$$\begin{cases} X = R(F(X : \vec{W}_f, \vec{b}_f)) \\ Y = F(R(Y : \vec{W}_b, \vec{b}_b)) \end{cases} \tag{3}$$

The other one is the reverse-to-forward cascaded model, including the forward model followed by the reverse model in (4).

$$\begin{cases} Y = F(R(Y : \vec{W}_b, \vec{b}_b)) \\ X = R(F(X : \vec{W}_f, \vec{b}_f)) \end{cases} \tag{4}$$

The train data is split by 20% for the validation data. The forward and the reverse models are trained with patience=1 in an early stop callback. Afterward, the cascaded model is trained for two epochs with an initial pre-trained learning rate of 0.001 to obtain the previous validation loss and the current validation loss. Then, the cascaded model is trained by meta-learning RL. The validation loss is calculated by BCE loss. The test loss is determined by customized BCE loss.

$$\underset{\vec{W}_f, \vec{b}_f, \vec{w}_b, \vec{b}_b}{Argmin}\ F\_R\_Cascade\_Losses(\vec{W}_f, \vec{b}_f, \vec{W}_b, \vec{b}_b) \tag{5a}$$

$$\underset{\vec{W_f},\vec{b_f},\vec{w_b},\vec{b_b}}{Argmin}\ R\_F\_Cascade\_Losses(\vec{W_f},\vec{b_f},\vec{W_b},\vec{b_b}) \tag{5b}$$

As shown in Fig. 8, the architecture involves cascading the forward and the reserves MLOPC model. In the forward-to-reverse cascaded model, the objective is to enhance the training of the initial cascaded model, with the aim of minimizing the validation loss as defined in (5a), where the *F_R_Cascade_Losses* are the BCE loss value of the predicted SEMs of the forward model and the predicted masks of the reverse model. Conversely, in the reverse-to-forward cascaded model, the training is conducted based on (5b), where the *R_F_Cascade_Losses* are the bce loss of the predicted masks of the reverse model and the predicted SEMs of the forward model. Training the cascaded models in this approach, together with the RL described in the next paragraph, aims to improve accuracy in the first model. The process is reversed if the input is a mask and the same procedure is followed in the opposite direction.

The meta-learner plays a role in adjusting the hyperparameters of the base learner based on the current performance of the base learner, which is cascaded in Fig. 8. We employ the DQN (Deep Q-Network) approach from RL (reinforcement learning) to establish a q-model as the meta-learner. In this framework, the state is defined as a combination of the validation losses at the first model in the cascaded model. We define four actions in our approach, which represent two sets of different learning rates (0.001 and 0.0001) and the weighted combination of losses from the forward and cascaded models in the cascade model. The weights for these combinations are [0.2, 0.8] and [0.8, 0.2], respectively. Consequently, there are four strategies of action resulting from the permutations and combinations, numbered [0,0], [0,1], [1,0], and [1,1]. Finally, the reward is defined as 1/loss of the first model in the cascaded model achieved by the action, where a minor loss corresponds to a greater reward for that action by (6).

$$action_{selected} = \underset{action}{argmax}\, r = argmax Q\left(state_{currnet}, action\right) \tag{6}$$

r is the reward in RL, and *Q* is the Q model. The q-model is composed of three dense layers. The first two layers each consist of 10 and 15 neurons with the ReLU activation function, while the final output layer comprises a single neuron with the swish activation function. The input to the q-model consists of four neurons. Specifically, the first two neurons receive inputs for the previous and current losses corresponding to the state of RL. The remaining two neurons receive the action codes as inputs. Once the q-model receives these inputs, it selects the action that maximizes the reward, thereby minimizing the validation data loss among the four possible actions in (7). The chosen action is then executed in the next iteration, and the q-model is repeatedly trained based on the newly obtained state, action, and reward combinations, as shown in Fig. 8. The goal is to teach the q-model the optimal action to select in different combinations of (previous state, current state) to yield the most significant reduction in the validation data loss. If the first model of the cascaded model fails to improve the validation loss within 30 epochs, the RL training process will be terminated.

$$\underset{\vec{W_q},\vec{b_q}}{argmin}\, r = \left(\left\| \begin{matrix} Q([state_{prev}, state_{current}], \\ [action_{prev}, action_{current}]) - r_{true} \end{matrix} \right\|^2 \right)_{mean} \tag{7}$$

The value of r_{true} was the state after the action.

Meta-learner employs the Epsilon Greedy [42] application method from Q-learning. When no values have been accumulated in the Q-table, we want the model to perform exploration in order to explore unknown territories to understand the feedback it brings. Initially, we set epsilon (ε) to 0, where there is a probability of 1-ε to execute actions randomly without considering the Q-table's results and a probability of ε to select the best action displayed in the Q-table

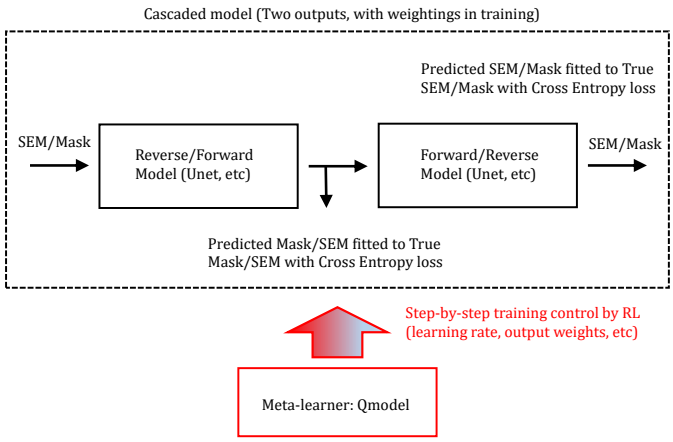

Fig. 8. The model of meta-learner and base-learner.

to achieve a maximized reward. Furthermore, ε increases over time because, in the beginning, we are uncertain about which actions are desired. We initially searched within the ε range of 0.005 to 0.03 with an increment of 0.005. We found that when ε = 0.015,

meat-RL outperformed most of the baselines. Therefore, following the same approach, we narrowed down the search range to ε = 0.01 to 0.02 with an increment of 0.001. We found that when a = 0.013, meta-RL exhibited the best performance. Consequently, we chose this value as our hyperparameter. As time progresses, ε increases by 0.013 after each epoch. This means that initially, there is more random behavior, but as time passes, decisions will be made more based on the Q-table, i.e., exploitation. Adjusting ε enables the model to transition between exploration and exploitation, allowing the model to achieve better results during training. To ensure that the Q-model maintains a certain level of flexibility in its choices until reaching the early stop condition of 30 times patience iterations, we set ε not to exceed 0.8. Additionally, suppose the patience is more than 20, regardless of the epoch. In that case, there is a 0.1 probability of selecting the worst action to prevent RL from getting stuck in local minima during training.

Once trained, the q-model serves as a meta-learner capable of self-training the cascaded model across different neural networks. This approach saves considerable time by automating the process of manually adjusting model parameters. Additionally, the meta-learner can adapt to various neural network architectures, thereby achieving the "learn to learn" objective in meta-learning.

TensorFlow 2.4.1 [43], NumPy 1.22.3 [44], OpenCV 4.7.0 [45], and GPU GeForce RTX 2080Ti in a Linux environment were used in this work.

Algorithm: Reinforce Learning for Meta Learner

```
For epochs
   action = argmax (Qmodel(state, action))
   if action == 1
      learning_rate = a1
      output_wt = b1
   elif
      learning_rate = a2
      output_wt = b2
   elif action == 1
      learning_rate = a3
      output_wt = b3
   else
      learning_rate = a4
      output_wt = b4
   Reward, Next_State = Cascade_model.fit(Xtrain,
Ytrain)
   Append/Update Rward, Next_State, action in Rarry,
   Sarry, Aarry
   Qmodel.fit([Sarry, Aarry], Rarry)
 end for
```

III. Results and Discussions

In our setting, the first model of the cascaded model can be improved in the meta-learned RL method, but the second one can be worse. Hence, the first model is used to compare with the model using the baseline training method. Fig. 9 to Fig. 14 show the test set predictions of actual SEMs in the three datasets for various models using the baseline Adam training or meta-learned RL training. Fig. 9 and Fig. 10 are the predictions of the two-bar data from the baseline and the meta-learning RL training method. Among them, the results of Unet, R2Unet, and the two kinds of Attention Unet are similar, and the predictions have no significant defects. Nevertheless, the baseline and meta-learning results of Vnet and ResUnet-a are very different. In Fig. 9, the border of Vnet's prediction graph is dim and bumpy, the outer frame of ResUnet-a's prediction is vague and unclear, and some regions on

the edges are disconnected. In contrast, using meta-learned RL, the edges of Vnet's predicted graph become regular, the corner sides of the outlines are improved, and the lines of ResUnet-a's predicted graph are seldom disconnected.

Fig. 11 and Fig. 12 are the predictions of the tri-bar dataset based on the baseline and meta-learned RL training method. The results of Unet and the two kinds of Attention Unet are similar; only the corners of the predicted graphs in these three models in the baseline are distorted and convex outward, while the results are slightly better when using meta-learned RL. R2Unet has similar results in the prediction of the two methods. The graphics in the baseline are less able to predict the changes in the corners and edges of the original SEM image. At the same time, the meta-learned RL method is more predictable, but the color of the corners will become lighter. The results of Vnet and ResUnet-a are significantly worse in the baseline. Vnet predicts graphs with rugged edges, unclear inner boundaries, and broken corners, and ResUnet-a can hardly predict complete pattern outlines. When meta-learned RL is used, although the corners of the prediction of Vnet are distorted as well, the edges become smoother. The prediction of ResUnet-a is more visible on its outline. Fig. 13 and Fig. 14 are predictions of the mixed dataset based on the baseline and meta-learned RL training. In this case, only the results of Vnet and ResUnet-a are worse than the other models. When using the meta-learned RL method, the gnarly edges of the Vnet prediction graphics are improved. Still, there will be some defects in the outlines. The disconnected regions on the border of the prediction from ResUnet-a become fewer.

Fig. 15 to Fig. 20 show the test set predictions of actual masks in the three datasets for various models using the baseline Adam training or meta-learned RL training. Fig. 15 and Fig. 16 are the predictions of the two-bar data from the baseline and the meta-learning RL training method. Among them, Unet, R2Unet, and the two kinds of Attention Unet reveal similar consequences in both training methods, whereas the predictions of Vnet and ResUnet-a from the two training are relatively distinct. When using baseline training, unet, R2Unet, and the two kinds of Attention Unet can predict the corners of the pattern outline well. Still, the outline edges are easily distorted, or the color of lines in some regions is faded. The outlines of Vnet and ResUnet-a predictions are uneven, and the contour edges are prone to noise. These problems are partially corrected when using meta-learning RL training. The flaws of the pattern outline can be fixed when using meta-learning RL training, but the prediction of the corners will be slightly worse. Overall, in terms of the customized BCE loss, the patterns with meta-learning RL training are closer to the actual SEM images.

Fig. 17 and Fig. 18 are the predictions of actual masks of the tri-bar dataset based on the baseline and meta-learned RL training method. In Fig. 17, the results of models with the baseline training show faded color in most regions on the edges, and deformations occur at the lower edges. Furthermore, Vnet and ResUnet-a, using the baseline training, can barely predict the complete pattern. On the contrary, as shown in Fig. 18, the outlines of the results become smooth with fewer disconnections if the meta-learning training is used, and the predictions from Vnet and ResUnet-a are more visible. Nevertheless, the lines of the patterns are slightly thicker in some cases. In Fig. 19 and Fig. 20, the results of Unet, R2Unet, and the two kinds of Attention Unet are satisfied, which are generally in line with the actual graphics, but some edges tend to become curved or translated when using the baseline training method. In contrast, these defects can be improved using the meta-learning RL training, but the angle at the corner will be slightly distorted. When Vnet uses the meta-learning RL training, the prediction is greatly enhanced. In addition to smoother edges, borders are no longer broken. The prediction of ResUnet-a improves the broken outline and uneven color, but the boundary of the internal test is still convex.

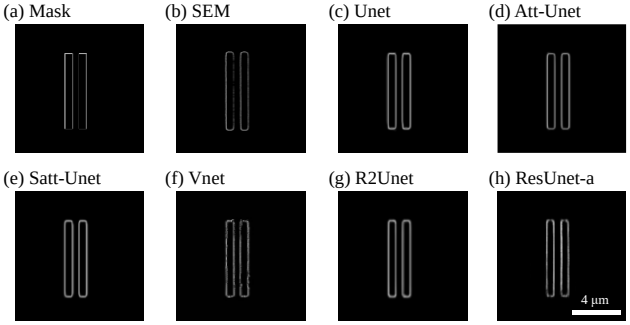

Fig. 9. The predicted SEMs and real SEMs of the two-bar patterns for the baseline training method.

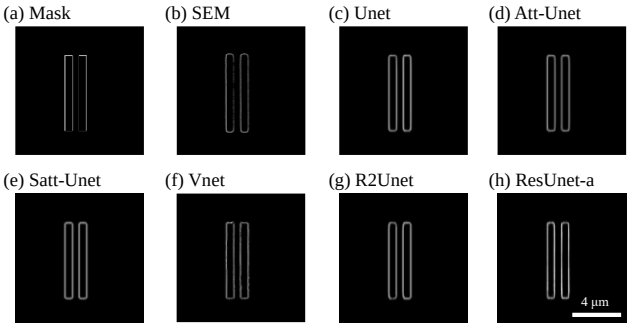

Fig. 10. The predicted SEMs and real SEMs of two-bar patterns for the meta-learning RL training.

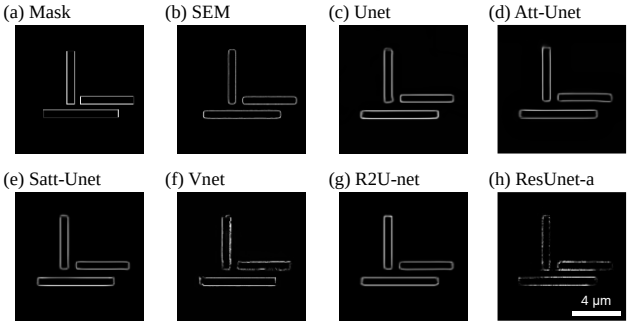

Fig. 11. The predicted SEMs and real SEMs of tri-line patterns or the baseline training method.

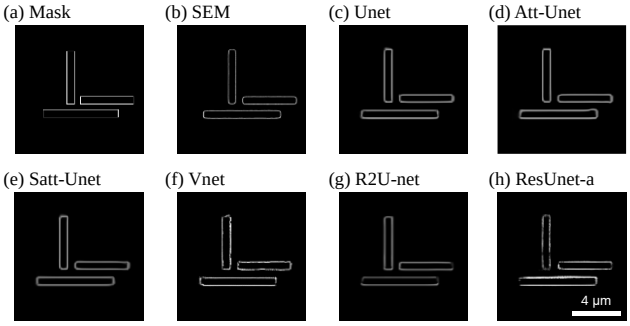

Fig. 12. The predicted SEMs and real SEMs of tri-line patterns for the meta-learning RL training.

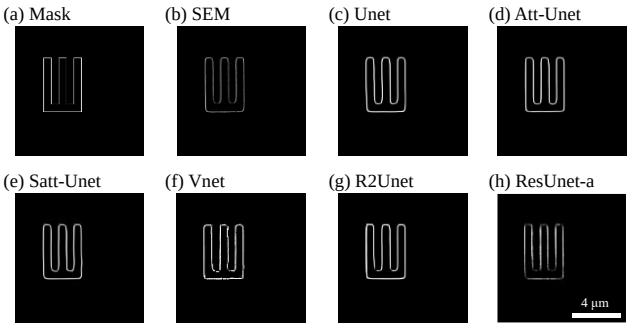

Fig. 13. The predicted SEMs and real SEMs of the mixed patterns for the baseline training method.

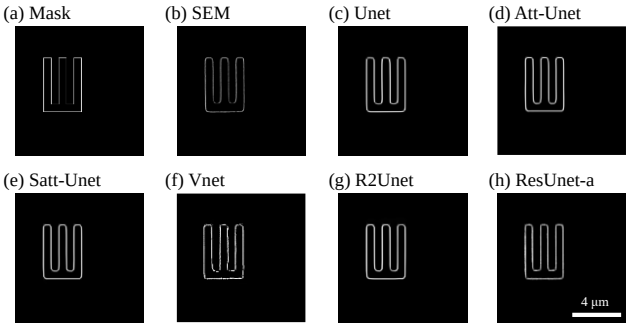

Fig. 14. The predicted SEMs and real SEMs of the mixed patterns for the meta-learning RL training.

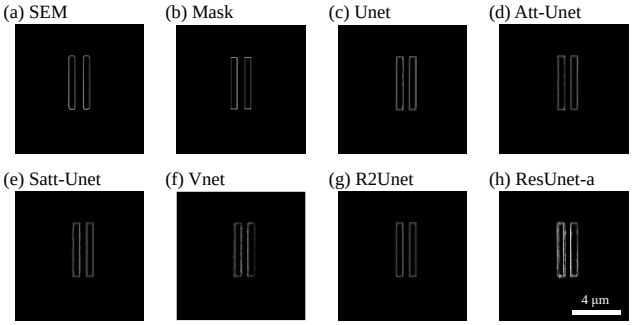

Fig. 15. The predicted Masks and real Masks of the two-bar patterns for the baseline training method.

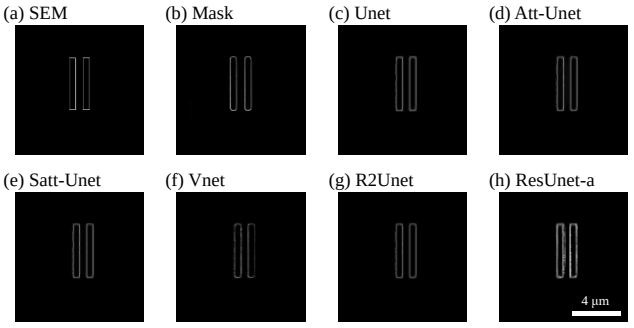

Fig. 16. The predicted Masks and real Masks of two-bar patterns for the meta-learning RL training.

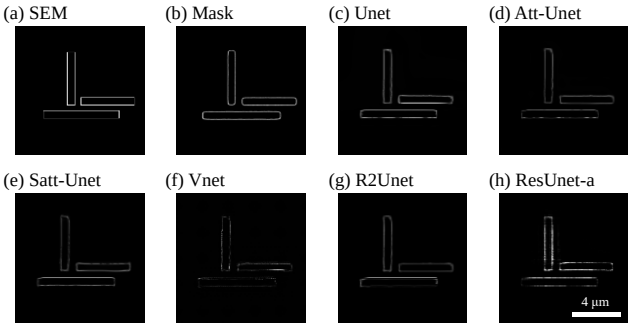

Fig. 17. The predicted Masks and real Masks of the tri-line patterns for the baseline training method.

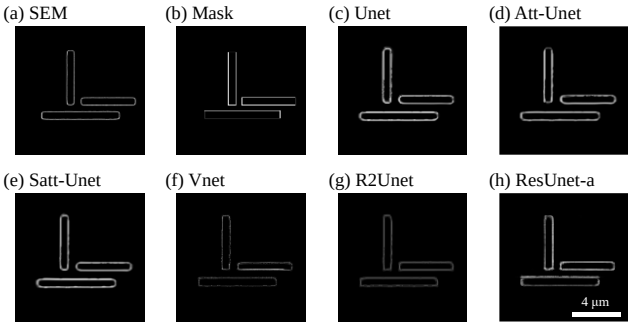

Fig. 18. The predicted Masks and real Masks of tri-line patterns for the meta-learning RL training.

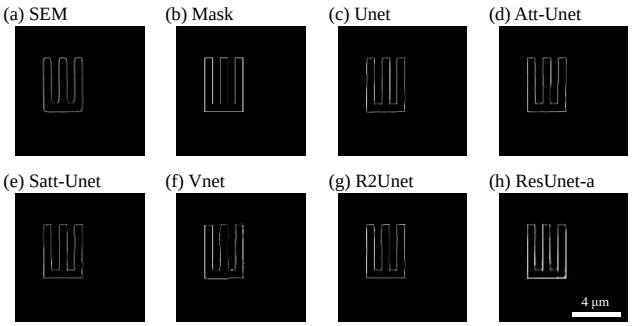

Fig. 19. The predicted Masks and real Masks of the mixed patterns for the baseline training method.

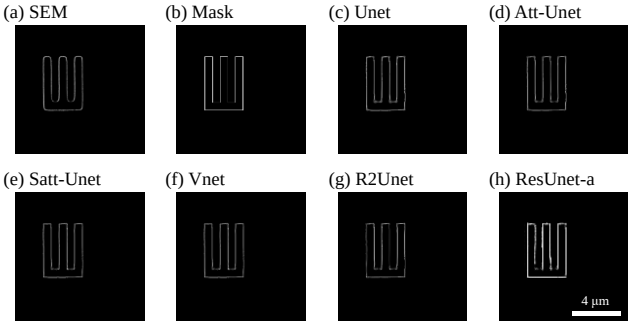

Fig. 20. The predicted Masks and real Masks of the mixed patterns for the meta-learning RL training.

Table 1 shows the results of the customized BCE loss value of the test set in six different models and three kinds of datasets from the baseline and the meta-learning RL with the forward-to-reverse cascade. Each model from the baseline and the meta-learning RL method has been trained five times, respectively, and the test set's maximum, minimum, and average customized BCE loss values are calculated. From the performance on either the BCE loss value of the validation set or the customized BCE loss values of the test set, the meta-learned RL is better than the baseline, where the models trained by the meta-learning RL can win 15 out of 18 cases in terms of the average values. In the tri-bar dataset, the customized BCE loss values of the six cases in the baseline show difficult training compared to the two-bar and mixed datasets. This observation shows the complexity of the tri-line dataset and the challenges imposed by a small data amount. Expressly, in the baseline results of tri-line patterns, the Vnet and ResUnet-a exhibit poor test set performance. It can be seen from the customized BCE loss score that the models are hardly trained well, and they also have many undesirable effects that can be observed in the test set predicted graphs, such as breaks or even distortions in the pattern outlines. On the other hand, if meta-learning RL is used, the results will have the opportunity to be significantly enhanced, especially for the models with more complex architectures such as ResUnet-a. In the rest of the other four models, an enhancement can also be observed in the customized BCE loss values of the test set. This indicates the meta-learning RL's effect in coping with these complicated situations. As for the two-bar and mixed datasets, although most of the baseline performance is already satisfactory, it can also be seen that Vnet and ResUnet-a models can be improved using meta-learning RL. In addition, Unet, two types of Attention Unets, and R2-Unet are also slightly enhanced by the meta-learning RL.

Table 2 shows the results of the customized BCE loss value of the test set in six different models and three kinds of datasets from meta-learned RL with reverse-to-forward cascading. Each model from the baseline and the meta-learning RL method has been trained five times, respectively, and the test set's maximum, minimum, and average customized BCE loss values are calculated. The models trained by the meta-learning RL can win 14 out of 18 cases in these cases. Among them, the Vnet and ResUnet-a models can be significantly improved in the three data sets due to the complexity of training. In the cases of Unet, the two types of attention Unets and R2Unet of the two-bar dataset, even if the results of the meta-learning method are similar to the baseline, the models can still be further improved by the meta-learning RL, which is reflected in corrections on the graphic outlines of the predictions. The results in

Table 1 and Table 2 manifest the effectivity of the meta-learning RL in either the forward-to-reverse or the reverse-to-forward cascades.

In the part of the two-bar dataset, the graph is more straightforward, and the amount of data is more than the other two, so the baseline result is relatively good, and the meta-learned RL method is not so helpful. As for the part of the mixed dataset, we think that although there are many types of patterns, most are relatively simple, so it is easier to train the model. This is also why the meta-learned RL can only be slightly helpful for some instances while using the mixed dataset. When the patterns are more complex, such as the tri-bar dataset, more complex error surfaces tend to exist under this situation. According to observations, it is more likely to be trapped in some saddle regions. It is necessary to use a flexible learning rate chosen to train to some extent for a better training result, especially for the more complicated models. As a result, the enhancements of Vnet and ResUnet-a, or sometimes Unet, are more prominent when using meta-learned RL.

The forward and reverse networks in MLOPC are two critical components. The forward network training is usually more difficult in our cases evaluated by the customized BCE loss, i.e., datasets, based on our test on the various mainstream image-to-image prediction ML models, including Unet, Vnet, ResUnet-a, R2Unet, and two types of Attention Unet. The goal is to utilize the cascaded model network and the meta-learning method to pull up the first model in the cascaded model, though this requires some considerations. Firstly, the successive substitution, where the repeated feeding, starting from the forward or the reverse network, is carried out for several runs. The doubt is whether the predicted SEM or mask will be more and more similar to the actual solutions. The test based on DUV-248 lithography in our study concludes that successive substitution is not a possible way to pull up the accuracy of the forward or the reverse network. The reason is that the error accumulates instead of diminishes, which seems to be understandable since the accuracy of the network is not perfect, and the error at each feeding can add up. Indeed, in theory, it is also possible that the forward and reverse network correct each other and leads to converged iteration in the successive substitution. The actual outcome, i.e., improved or degraded test set prediction, depends on the problem's nature and cannot be known beforehand. With the fitting only conducted at the final output after the successive substitution, we often observe the significantly degraded test set prediction. Additionally, after multiple substitutions, the bulk model leads to optimizer burdens.

It should be emphasized that the training setting is critical to boosting the mutual correction nature. The training should be implemented with auto-encoder-like architecture, and the fitting is established at the encoded vector and the endpoint for one iteration to make the mutual correction more pronounced. Furthermore, if successive substitutions are executed several times, more and more outputs are generated, and all of them should be fitted to the true SEMs or masks. The increased number of network outputs and the increased fitting loading result in unmanageable, poor convergence, which challenges the optimizers' training capability based on our test. In many test cases, four to six successive substations already lead to training problems where the binary cross-entropy loss minimization can easily be trapped, and the prediction accuracy of the forward and reverse model is not increased at all from the first or second epoch. Therefore, we finally choose only one substitution for enhancing the forward model, where the mask goes into the forward model first, and the predicted SEM goes into the reverse model, thus completing one iteration. More iteration only leads to degraded results.

Table 1. The customized BCE loss value of the test set in the six models using the baseline and the meta-learning RL training with the forward-to-reverse cascade.

Forward Model (patience = 30) F-to-R								
Tri-bar	Baseline				Meta-learned RL			
Model	val. BCE	test set customized BCE			val. BCE	test set customized BCE		
	avg	max	min	avg	avg	max	min	avg
Unet	0.03757	0.37668	0.36181	0.36590	0.03197	0.31737	0.29591	0.30553
Att. Unet	0.03831	0.37904	0.30536	0.34001	0.03277	0.31543	0.30088	0.30781
Self Att. Unet	0.04167	0.45026	0.33929	0.38461	0.03267	0.32904	0.29079	0.30601
Vnet	0.05401	0.49967	0.47175	0.48415	0.04047	0.41299	0.34563	0.37438

R2Unet	0.03235	0.32980	0.30499	0.31840	0.03357	0.32302	0.29928	0.31011
ResUnet-a	0.08736	0.92171	0.64350	0.76581	0.03754	0.43563	0.40354	0.41968

Two-bar	Baseline				Meta-learned RL			
Model	val. BCE	test set customized BCE			val. BCE	test set customized BCE		
	avg	max	min	avg	avg	max	min	avg
Unet	0.01662	0.20069	0.17742	0.18848	0.01588	0.19075	0.18272	0.18598
Att. Unet	0.01613	0.19085	0.18180	0.18610	0.01568	0.19463	0.18581	0.18932
Self Att. Unet	0.01607	0.18372	0.17397	0.17748	0.01624	0.18588	0.17967	0.18287
Vnet	0.02105	0.23293	0.20109	0.21710	0.01799	0.21644	0.20285	0.20961
R2Unet	0.01598	0.19352	0.18022	0.18497	0.01581	0.18013	0.17408	0.17627
ResUnet-a	0.01960	0.24555	0.21619	0.22963	0.01849	0.23817	0.20839	0.22400

Mixed-bar	Baseline				Meta-learned RL			
Model	val. BCE	test set customized BCE			val. BCE	test set customized BCE		
	avg	max	min	avg	avg	max	min	avg
Unet	0.01134	0.13226	0.12587	0.12940	0.01014	0.12796	0.11464	0.12199
Att. Unet	0.01260	0.16001	0.14585	0.15168	0.01086	0.14538	0.13444	0.14062
Self Att. Unet	0.01257	0.15839	0.14276	0.15306	0.01108	0.14174	0.13665	0.13970
Vnet	0.01662	0.21637	0.18958	0.20518	0.01439	0.20618	0.18824	0.19753
R2Unet	0.01119	0.15859	0.12315	0.13306	0.01117	0.14933	0.12824	0.13879
ResUnet-a	0.01667	0.36609	0.17035	0.24346	0.01382	0.40830	0.16467	0.22509

Table 2. The customized BCE loss value of the test set in the six models using baseline and meta-learned RL training with the reverse-to-forward cascade.

Backward Model (patience = 30) R-to-F								
Tri-bar	Baseline				Meta-learned RL			
Model	val. BCE	test set customized BCE			val. BCE	test set customized BCE		
	avg	max	min	avg	avg	max	min	avg
Unet	0.02254	0.32139	0.24338	0.26959	0.02306	0.22957	0.22392	0.22608
Att. Unet	0.02362	0.26155	0.22938	0.24466	0.02456	0.23981	0.22633	0.23273
Self Att. Unet	0.02549	0.32270	0.26449	0.29216	0.02392	0.23546	0.21316	0.22542
Vnet	0.03200	0.36934	0.34330	0.35668	0.02571	0.29662	0.25490	0.27340
R2Unet	0.02185	0.23288	0.19543	0.21488	0.02475	0.23116	0.22245	0.22877
ResUnet-a	0.04186	0.44684	0.40937	0.42795	0.03693	0.39774	0.36766	0.37589

Two-bar	Baseline				Meta-learned RL			
Model	val. BCE	test set customized BCE			val. BCE	test set customized BCE		
	avg	max	min	avg	avg	max	min	avg
Unet	0.01278	0.14194	0.13638	0.13840	0.01253	0.13956	0.13474	0.13684
Att. Unet	0.01293	0.14233	0.13414	0.13690	0.01289	0.14414	0.13734	0.14116

Self Att. Unet	0.01263	0.14739	0.13910	0.14428	0.01260	0.15243	0.13790	0.14130
Vnet	0.01655	0.16865	0.15842	0.16313	0.01357	0.15256	0.14770	0.14979
R2Unet	0.01293	0.14528	0.13659	0.14038	0.01265	0.14181	0.13627	0.13882
ResUnet-a	0.02918	0.37063	0.22398	0.28481	0.01910	0.28562	0.23995	0.25849

Mixed-bar	Baseline				Meta-learned RL			
Model	val. BCE	test set customized BCE			val. BCE	test set customized BCE		
	avg	max	min	avg	avg	max	min	avg
Unet	0.00749	0.11485	0.08765	0.09856	0.00782	0.10627	0.09087	0.09985
Att. Unet	0.00886	0.11762	0.10182	0.11106	0.00815	0.10930	0.10364	0.10690
Self Att. Unet	0.00861	0.12131	0.10478	0.11301	0.00815	0.11392	0.10287	0.10734
Vnet	0.01472	0.17850	0.14857	0.15890	0.01012	0.13326	0.12079	0.12588
R2Unet	0.00744	0.11171	0.09232	0.10269	0.00856	0.11216	0.10942	0.11046
ResUnet-a	0.01028	0.58830	0.15950	0.32804	0.01033	0.15608	0.12766	0.14185

Moreover, the complex model trained by the baseline method or under complex data often gives results with large fluctuations owing to the GPU variability, indicating the instability of the baseline method. In contrast to the models using baseline training, the test set results of the model trained by the meta-learning RL with the cascaded model are relatively stable under the same setting. Thus, besides outperforming the baseline training method in most cases, the predictions of the models trained with meta-learning RL training are similar in each run.

In addition to the cascade order, the flow in meta-learning RL training can be in different settings and take some considerations. First, we can train the forward and the reverse network in advance using the early stop patience setting. After that, the trained forward and reverse network is cascaded together in the order described above and trained as a whole, with the weights set as trainable. The weights of the second network in the cascaded model can also be set as untrainable if their accuracy is high. The second way is training the cascaded model from scratch, which means the weights in the forward and reverse models start from tensorflow weight initializers. Finally, it is also possible to train both the forward and reverse models slightly using small patience in the early stop, and the training of the cascaded model is established from that. Regarding the final cascaded model training, it is also the users' choice to fit the final output, i.e., masks, or the intermediate and final output, i.e., SEMs and masks. As discussed above, in theory, the mutual correction phenomenon should be more pronounced if we fit the intermediate output to drag back the deviated network predictions.

Regarding the network hyperparameter tuning, novel training scheme, and avoiding poor training, some investigation deserves attention in OPC network training. Based on our tests, the training is prone to settle on the highly local minimum in the forward OPC problems. Therefore, it is desirable to have a way to avoid these pitfalls and boost the training automatically. The manual tuning of the training is not pleasant. Hyperparameter tuning means repeated training on the training set, which is time-consuming. Manual tuning becomes impossible if the training scheme is made even more complex. For example, if we decide to change the learning rate, output weights, and other settings, manual trials on the train path can be unmanageable. In this work, we use a meta-learning approach using reinforcement learning. Actually, many different algorithms can be used to realize meta-learning besides RL. This includes genetic algorithms (GA), artificial neural networks (ANN), any ML algorithms together with an optimization algorithm, and all optimization algorithms alone, especially those that require no gradient. We choose RL here since the effect of using RL is satisfactory, though the same meta-learning flow presented in this work can be accomplished with any other algorithms. When constructing the meta-learning framework here, the first consideration is how many training setting possibilities are used in the procedure and how many training possibilities are enough to boost the accuracy. Second, we should assess if the meta-learning algorithm is efficient with so many training combinations. Here, it is decided that the learning rate, the fitting output weights, and the network connection in training, i.e., cascaded or isolated, are essential factors based on the test before meta-learning. Adding more training possibilities can overload the meta-learning algorithms and increase the GPU runtime. A key, reduced set of training possibilities should be used unless necessary. On the other hand, RL is a versatile framework that can tackle large-scale problems and is thus suitable for our purpose.

Regarding the RL implementation, the decision of the approximate state can be a crucial consideration. Due to overly large state space, it is generally impossible to use precise states in many real RL problems, including ours. The approximate state should reflect the true state in the environment and facilitate action-state-reward mapping. The reward is the validation set binary cross-entropy of the first model in the cascaded model, which can also be changed to customized loss as described in section II further to highlight the critical part of SEMs or masks. Actions can be incremental type, i.e., +1/0/-1, or non-incremental type, i.e., directly transition to any state. Incremental type action leads to smooth convergence but is less efficient, while the non-incremental type

action gives faster convergence but can lead to convergence problems in some cases. In this work, we use a non-incremental action type since the learning rate and fitting output weights do not need to be changed smoothly in theory to prevent convergence problems. In addition, since the training is conducted at each step in RL, using incremental type action limits the search space unless an additional state variable is added to the total state to indicate the potential pause of training at the current step.

IV. Conclusion

This study can enhance the MLOPC model accuracy through the proposed meta-learning RL training method. The training path controlled by a meta-learner is shown to be effective in getting out of a predicament when a model is prone to get stuck in a local minimum, which can be crucial in many practical fields where a dataset is complicated or the model used is difficult to train. Various mainstream image-to-image ML models have been tested, including Unet, Vnet, ResUnet-a, R2Unet, and two types of Attention-Unet, to verify the usability of the proposed meta-learning training. The selection of RL actions at each training step adjusts the training settings, and exploitation is tuned using the Epsilon Greedy algorithm. With the combination of the cascaded forward and reverse models and meta-learning, the optimal model training can achieve improved results in comparison to the baseline Adam optimizers. After training, the lower customized BCE values of the test set predictions reflected that the models trained by a meta-learning RL have enhanced comprehension of the SEM image outlines, such as smoother outlines or improved corner regions closer to real SEMs. The averaged increments of 19.5%, 1.2%, and 4.8% can be seen in the forward models, and the averaged increments of 12%, 3.1%, and 13% in the reverse models can be seen in the three datasets, where the enhancements of ResUnet-a in the more complicated dataset are particularly evident.

References

[1] J. Wang, A. Wei, P. Verma, and W. Wilkinson, "Optimization of rule-based OPC fragmentation to improve wafer image rippling," in *31st European Mask and Lithography Conference*, 2015, pp. 79-94.

[2] J.-S. Park, C.-H. Park, S.-U. Rhie, Y.-H. Kim, M.-H. Yoo, J.-T. Kong*, et al.*, "An efficient rule-based opc approach using a drc tool for 0.18/spl mu/m asic," in *Proceedings IEEE 2000 First International Symposium on Quality Electronic Design (Cat. No. PR00525)*, 2000, pp. 81-85.

[3] N. B. Cobb and Y. Granik, "Model-based OPC using the MEEF matrix," in *22nd Annual BACUS Symposium on Photomask Technology*, 2002, pp. 1281-1292.

[4] Y. Granik and N. B. Cobb, "New process models for OPC at sub-90-nm nodes," in *Optical Microlithography XVI*, 2003, pp. 1166-1175.

[5] K. Patterson, Y. Trouiller, K. Lucas, J. Belledent, A. Borjon, Y. Rody*, et al.*, "Improving model-based OPC performance for the 65-nm node through calibration set optimization," in *Design and Process Integration for Microelectronic Manufacturing III*, 2005, pp. 294-301.

[6] D. Chou and K. McAllister, "Line End Optimization through Optical Proximity Correction (OPC): A Case Study," in *Optical Microlithography XIX*, 2006, pp. 1099-1110.

[7] R. Torrance and D. James, "The state-of-the-art in IC reverse engineering," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2009, pp. 363-381.

[8] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," in *Proceedings of the 48th Design Automation Conference*, 2011, pp. 333-338.

[9] L. Pang, Y. Liu, and D. Abrams, "Inverse lithography technology (ILT): What is the impact to the photomask industry?," in *Photomask and Next-Generation Lithography Mask Technology XIII*, 2006, pp. 233-243.

[10] D. S. Abrams and L. Pang, "Fast inverse lithography technology," in *Optical Microlithography XIX*, 2006, pp. 534-542.

[11] L. Pang, G. Xiao, V. Tolani, P. Hu, T. Cecil, T. Dam*, et al.*, "Considering MEEF in inverse lithography technology (ILT) and source mask optimization (SMO)," in *Photomask Technology 2008*, 2008, pp. 631-644.

[12] B.-G. Kim, S. S. Suh, B.-S. Kim, S.-G. Woo, H.-K. Cho, V. Tolani*, et al.*, "Trade-off between inverse lithography mask complexity and lithographic performance," in *Photomask and Next-Generation Lithography Mask Technology XVI*, 2009, pp. 458-468.

[13] A. Gu and A. Zakhor, "Optical proximity correction with linear regression," *IEEE Transactions on Semiconductor Manufacturing,* vol. 21, pp. 263-271, 2008.

[14] N. Jia and E. Y. Lam, "Machine learning for inverse lithography: using stochastic gradient descent for robust photomask synthesis," *Journal of Optics,* vol. 12, p. 045601, 2010.

[15] R. Luo, "Optical proximity correction using a multilayer perceptron neural network," *Journal of Optics,* vol. 15, p. 075708, 2013.

[16] X. Ma, Q. Zhao, H. Zhang, Z. Wang, and G. R. Arce, "Model-driven convolution neural network for inverse lithography," *Optics Express,* vol. 26, pp. 32565-32584, 2018.

[17] T. Cecil, K. Braam, A. Omran, A. Poonawala, J. Shu, and C. Vandam, "Establishing fast, practical, full-chip ILT flows using machine learning," in *Optical Microlithography XXXIII*, 2020, pp. 12-29.

[18] V. Borisov and J. Scheible, "Lithography hotspots detection using deep learning," in *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2018, pp. 145-148.

[19] H. Yang, Y. Lin, B. Yu, and E. F. Young, "Lithography hotspot detection: From shallow to deep learning," in *2017 30th IEEE International System-on-Chip Conference (SOCC)*, 2017, pp. 233-238.

[20] H. Yang, J. Su, Y. Zou, B. Yu, and E. F. Young, "Layout hotspot detection with feature tensor generation and deep biased learning," in *Proceedings of the 54th Annual Design Automation Conference 2017*, 2017, pp. 1-6.

[21] H.-C. T. Albert Lin, Yen-Wei Feng, Peichen Yu, "Investigation on attention-based convolutional architectures applied to optical proximity corrections," presented at the SPIE Optics + Photonics, San Diego, 2023.

[22] H.-C. Shao, C.-Y. Peng, J.-R. Wu, C.-W. Lin, S.-Y. Fang, P.-Y. Tsai*, et al.*, "From IC layout to die photograph: a CNN-based data-driven approach," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 40, pp. 957-970, 2020.

[23] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, 2015, pp. 234-241.

[24] Y. Yan, X. Shi, T. Zhou, B. Xu, C. Li, W. Yuan*, et al.*, "Machine learning virtual SEM metrology and SEM-based OPC model methodology," *Journal of Micro/Nanopatterning, Materials, and Metrology,* vol. 20, pp. 041204-041204, 2021.

[25] C. Yu and X. Ma, "Thick-mask model based on multi-channel U-Net for EUV lithography," in *DTCO and Computational Patterning II*, 2023, pp. 504-513.

[26] Q. Cao, P. Xu, J. Wei, R. Liu, F. Li, J. Fan*, et al.*, "Hybrid deep learning OPC framework with generative adversarial network," in *DTCO and Computational Patterning II*, 2023.

[27] L.-Y. Xiao, J.-N. Yi, Y. Mao, X.-Y. Qi, R. Hong, and Q. H. Liu, "A Novel Optical Proximity Correction (OPC) System Based on Deep Learning Method for the Extreme Ultraviolet (EUV) Lithography," *Progress In Electromagnetics Research,* vol. 176, pp. 95-108, 2023.

[28] P. Parashar, C. Akbar, T. S. Rawat, S. Pratik, R. Butola, S. H. Chen*, et al.*, "Intelligent photolithography corrections using dimensionality reductions," *IEEE Photonics Journal,* vol. 11, pp. 1-15, 2019.

[29] C.-H. Chen, W.-D. Zhao, T. Pang, and Y.-Z. Lin, "Virtual metrology of semiconductor PVD process based on combination of tree-based ensemble model," *Isa Transactions,* vol. 103, pp. 192-202, 2020.

[30] A. Nichol, J. Achiam, and J. Schulman, "On first-order meta-learning algorithms," *arXiv preprint arXiv:1803.02999,* 2018.

[31] J. Lee and C. Yang, "Deep neural network and meta-learning-based reactive sputtering with small data sample counts," *Journal of Manufacturing Systems,* vol. 62, pp. 703-717, 2022.

[32] T. S. Rawat, C. Y. Chang, Y.-W. Feng, S. Chen, C.-H. Shen, J.-M. Shieh*, et al.*, "Meta-Learned and TCAD-Assisted Sampling in Semiconductor Laser Annealing," *ACS omega,* vol. 8, pp. 737-746, 2022.

[33] A. Lin, T. Rawat, C.-Y. Chang, H.-C. Tung, H.-L. Liu, and P. Yu, "Optical Proximity Correction Using Machine Learning Assisted Human Decision," *IEEE Photonics Journal,* vol. 15, pp. 1-9, 2022.

[34] L. H. Gabrielli. *L. H. Gabrielli, "gdspy 1.6.13", retrieved https://pypi.org/project/gdspy/.* Available: https://pypi.org/project/gdspy/

[35] G. a. D. Van Rossum, Fred L., *Van Rossum, G. & Drake, F.L., 2009. Python 3 Reference Manual, Scotts Valley, CA: CreateSpace.*: CreateSpace, 2009.

[36] *https://www.klayout.de/*. Available: https://www.klayout.de/

[37] Y. Sha. (2021). *Sha, Y., 2021: Keras-unet-collection. GitHub repository, accessed 4 September 2021, https://doi.org/10.5281/zenodo.5449801*. Available: https://github.com/yingkaisha/keras-unet-collection

[38] X. Wang, Z. Hua, and J. Li, "Cross-UNet: dual-branch infrared and visible image fusion framework based on cross-convolution and attention mechanism," *The Visual Computer,* pp. 1-18, 2022.

[39] F. I. Diakogiannis, F. Waldner, P. Caccetta, and C. Wu, "ResUNet-a: A deep learning framework for semantic segmentation of remotely sensed data," *ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 162, pp. 94-114, 2020.

[40] M. Z. Alom, C. Yakopcic, T. M. Taha, and V. K. Asari, "Nuclei segmentation with recurrent residual convolutional neural networks based U-Net (R2U-Net)," in *NAECON 2018-IEEE National Aerospace and Electronics Conference*, 2018, pp. 228-233.

[41] F. Milletari, N. Navab, and S.-A. Ahmadi, "V-net: Fully convolutional neural networks for volumetric medical image segmentation," in *2016 fourth international conference on 3D vision (3DV)*, 2016, pp. 565-571.

[42] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra*, et al.*, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602,* 2013.

[43] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro*, et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467,* 2016.

[44] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau*, et al.*, "Array programming with NumPy," *Nature,* vol. 585, pp. 357-362, 2020.

[45] G. Bradski, "The openCV library," *Dr. Dobb's Journal: Software Tools for the Professional Programmer,* vol. 25, pp. 120-123, 2000.