COMPUTER VISION FOR ANALYSIS OF SCANNING ELECTRON MICROSCOPE (SEM) OF INTEGRATED CIRCUITS

Chieu Le Yang[1], Ryler Ng Kai Guan[1], Shen Bingquan[2]

[1]NUS High School of Mathematics and Science, 20 Clementi Avenue 1, Singapore 129957

[2]DSO National Laboratories, 12 Science Park Dr, Singapore 118225

Abstract

Integrated circuits (ICs) are essential in technology and national security, with their market estimated at $616.9 billion in 2024 and projected to grow to $1.9 trillion by 2032[3]. Quality control is critical, yet challenging due to the vast number of ICs produced annually. Computer vision models like Segment Anything Model-2 (SAM-2) [4], U-NET [5], and RSPrompter [2] offer solutions through image segmentation, but they require fine-tuning and large labelled datasets for accuracy. This study finetuned these models using images of IC features captured via scanning electron microscopy, optimizing parameters like learning rates and data volume. SAM-SEG outperformed with a 94% pixel-wise accuracy, followed by U-NET, while RSPrompter lagged slightly. To reduce effort required for labelling, semi-supervised learning was explored. Using 800 labelled and 200 unlabelled images yielded a 91% accuracy, demonstrating the feasibility of reducing labelled data without significant accuracy loss. Key findings highlighted SAM-2's robustness, U-NET's simplicity for small datasets, and semi-supervised learning's potential to balance effort and performance. Future research could explore advanced clustering and larger datasets for further improvements. This work underscores the importance of computer vision in IC quality control and opens pathways for efficient, scalable solutions.

Introduction

Integrated circuits are used in a wide range of devices, such as phones, computers and other electronic devices. The demand for integrated circuits has been growing recently – according to an ongoing study by Fortune Business Insights, this market is projected to grow to 1901.95 billion US dollars in 2032, at a compound annual growth rate (CAGR) of 13.4% [3]. However, each part of an integrated circuit needs to be carefully inspected for defects before it is used or sold. With the sheer number of integrated circuits produced annually, it is almost impossible for humans to keep up with the demands of this quality control task. Computer vision, which consists of objection detection and image segmentation, offers several advantages here – faster image processing, less resources needed for training, lower error rates and higher accuracy. However, computer vision models are not perfect, and have much room for improvement. Two major issues were identified – pre-trained computer vision models such as SAM-2 were trained on huge datasets but are not optimized and designed to do specific tasks, such as identifying components of an integrated circuit [4]. Accuracy in this industry is extremely important as amongst the billions of integrated circuits manufactured per year, many of them will have defects. Should these defects not be identified by computer vision models, they could affect the performance of electronics. In this paper, we attempt to finetune various pre-trained models to conduct supervised learning on a few types of integrated circuits. This model would take in images of features of integrated circuits taken using a scanning electron microscope and output a mask showing the features of the image, which can then be used for comparison and checking for defects. This brings up another issue – to train and finetune computer vision models, training data and images are needed. Large amounts of labelled data would need to be provided to the machine learning model to achieve good accuracy. This labelled data might not be easy or feasible to obtain in large quantities, as it would involve manually identifying features in every image, which can be tedious for upwards of 1000 images. Hence, in this paper, we also attempt to increase performance by using semi-supervised learning, so that no more than 1000 labelled images would be needed, and more unlabelled images can be used instead for training.

Part 1: Finetuning SAM-DET, SAM-SEG, RSPrompter and U-NET

We utilized a few different models, building off the SAM-2 [4] as well as making our own U-NET model [5]. We also mainly focus on the avr_m1 class of integrated circuits.

SAM-SEG Model

Loading Data

First, we loaded 1000 images using torch.utills.data.Dataloader. By editing the list of image and mask paths, we can control the amount of data that is used for training and for testing (for example, restricting the length of the list of images and masks used for training to only 600 and that for testing to 400, we can obtain a train/test split of 60/40). Augmentations such as rotation and flipping were also used to obtain a more diverse data set. Lastly, images were resized to the correct tensor dimensions ([256,256,3]) for training and testing.

Loading pre-trained model

Following the GitHub notebook attached to the SAM-2 paper, all the required files were saved to Google Drive [4]. This model consists of 2 main parts: a prompt encoder, and a mask decoder.

The model was imported using the model config, model weights and the built-in function build_sam.py provided amongst the files. Specifically, the “large” model was selected. Other sizes, such as “small” and “huge” also could work, but the “small” model was not complex enough for the task.

Prompts

One feature of SAM-SEG is that it requires prompts to obtain accurate masks. Hence, 100 prompts are provided per image during training. These prompts are obtained from the ground truth masks provided. As many images within the dataset had blank ground truth masks (without any features), initially choosing points at random and finding the pixel value of these points did not work well – for 5 or more epochs, as many of these prompts were negative prompts (prompts which suggest that that point is not part of any feature) which were obtained mainly from the blank ground truth masks. This then caused the model to gradually learn to predict blank ground truth masks regardless of the image provided. To solve this issue, prompts were selected so that all but one of the prompts were positive prompts (suggesting that there is a feature at the selected point), and the last prompt was chosen to be a negative prompt.

Finetuning

To finetune SAM-SEG, we took reference to the notebook “fine-tune-train-segment-anything-2-in-60-lines-of-code" [4]. We were able to freeze all but the mask decoder layers within the pre-trained model. Prompts were provided along with images and masks in batches of 10. The model was updated after every batch and 5 epochs were run. To find the ideal learning rate (one which was not too high and caused overfitting, and one which was too low to converge fast enough), different learning rates ranging from 1e-4 to 1e-8 were tested. The weight decay was always set to be 1/10000 of the learning rate. A custom loss function was also used, one which heavily penalizes the model for false negatives (declaring a point to not be within a feature when it is), to discourage the model from optimizing towards giving blank tensors. The Adam optimizer was chosen, and a scaler (GradScaler) was used to decrease the learning rate and weight decay by a factor of 10 every epoch. Lastly, to test performance drop due to low training data, the number of images provided for training was increased in increments of 200 from 200 up to 1000.

Testing

A set of data for testing was also loaded with the data loader, equal in size to that of the training data. For example, if 800 training images were provided, 800 images would be used for testing.

20 prompts were provided, and predicted masks were obtained from the model, before Gaussian normalization was used to obtain less grainy masks. Pixel-wise accuracy was used to check for the accuracy of mask. This was then compared to U-NET and finetuned RSPrompter.

U-NET

Model

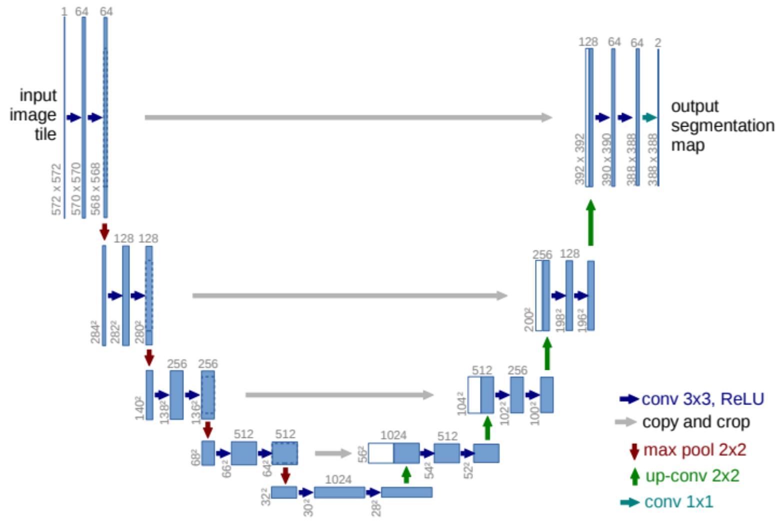


Diagram 1: diagram of the original model used, copied from the paper titled ”U-NET: Convolutional Networks for Biomedical Image Segmentation” [5] The input was modified to take in 256x256 inputs, in line with the size of the images. The padding was also changed to be the same in each layer in order to keep the output masks and the input images the same size

This U-NET model consists of downsampling, bottleneck and upsampling.

In the downsampling path, there are DoubleConv and Max Pooling layers that extract features and with ReLU activation and batch normalization to obtain higher-level contextual and important features. The bottleneck (the deepest layer) applies a DoubleConv block to extract abstract features. The upsampling path then uses Transpose Convolution layers to reconstruct the resolution. Skip Connections combine the high-resolution features from the downsampling path with the upsampled features, ensuring that fine details are preserved, while a DoubleConv block then refines the combined feature maps, before the final Convolution layer reduces the channel dimension to produce the output segmentation map.

Training & Testing

Similarly to SAM-SEG, 5 epochs were run, and the learning rate and amount of training data provided were changed. The loss function used was BCEWithLogitsLoss, and the same scaler GradScaler was used to adjust the learning rate during training. The same testing method was used on the trained U-NET model as SAM-SEG, and pixel-wise accuracy values were obtained for comparison.

SAM-DET

We followed the steps outlined in the github page titled "RSPrompter: Learning to Prompt for Remote Sensing Instance Segmentation" [1]. SAM-DET was not expected to do as well as the other models but was still included in this comparison to provide a baseline for performance evaluation and to show the strengths of the other models at segmentation.

Loading Data

First, we converted 1000 masks from the .png format into the COCO format by first loading the image files into 256 by 256 arrays using cv2.imread(). Next, contours were generated for the arrays using cv2.findContours(). All the annotations for the 1000 avr_m1 train images and 1000 test images were then put into separate .json files following the COCO dataset format. The path to the COCO file is then put into the samdet.py configuration file.

Loading pre-trained model

We cloned the github repository. To adapt the pre-trained model for our project, we made some modifications to the samdet.py configuration file, such as setting the max number of epochs to 100 and changing the file paths to the annotations. The model consists of 4 main parts: the image encoder, the detector, the prompt encoder and the mask decoder. The "small" model was selected as it was the default model defined in the configuration files. The model was trained by running tools/train.py and inputting the modified samdet.py configuration file.

Finetuning

To finetune SAM-DET, we took reference to the github page "RSPrompter: Learning to Prompt for Remote Sensing Instance Segmentation based on Visual Foundation Model" [2]. By running the train.py, all but the detector was frozen. The model was updated after every batch and 5 epochs were run. Various learning rates were tried out, ranging from 1e-3 to 1e-8, with a higher learning rate resulting in a better model after 5 epochs, with the exception of the 1e-3 learning rate which resulted in silent failures where the model train function would just output a value of zero for all the loss metrics and an accuracy of 100. However, in order to prevent overshooting the optimal point, ensuring stable convergence and avoiding divergence or large errors, the learning rate was ultimately set to 5e-5 and the number of epochs set to 100 epochs. The weight decay was always set to be 0.05. The same loss functions as those in the original SAM-DET model are used, thus there are two different loss functions being used for this model, a classification loss and a bounding box regression loss. This classification loss is irrelevant as this is purely a segmentation task. The bounding box regression loss calculates the difference between the predicted and ground truth bounding box images, and usesSmoothL1Loss function. A cosine annealing scheduler was used in order to help the model converge more effectively towards the end of training. The same test of performance drops due to less training data was tested.

Testing

A set of data for testing was also loaded with the data loader, with all 1000 testing images being used at each time. The model was given the input image, and predicted masks were obtained from the model. Pixel-wise accuracy was used to check for the accuracy of mask.

RSPrompter

Similarly to SAM-DET, we followed the steps outlined in the github page "RSPrompter: Learning to Prompt for Remote Sensing Instance Segmentation based on Visual Foundation Model" [2].

Loading Data

The same annotations files in COCO format used for SAM-DET were used for RSPrompter.

Loading pre-trained model

The same github repository was used. As the default RSPrompter configuration pointed to using the "huge" model, the "huge" model was used.

Finetuning

Similar variations in learning rate were tested from 1e-4 to 1e-8. However, the RSPrompter training loop would crash when ran with the 1e-4 learning rate. Thus, a learning rate of 5e-4 was ultimately decided upon, with 10 epochs as it offered the highest learning rate that would not result in a crash. The weight decay was again set to 0.05. Three loss functions were used, classification loss of type CrossEntropyLoss with a weight of 2.0, mask loss of type

CrossEntropyLoss with a weight of 5.0 and dice loss of type DiceLoss with a weight of 5.0. A cosine annealing scheduler was used again in order to help the model converge more effectively towards the end of training. The configuration file was again modified to change the annotation file paths and to decrease the number of prompts generated from 100 to 12 as most images had less than 12 annotations on average.

Testing

The same testing methods and evaluation metrics applied to SAM-DET were also used for RSPrompter to ensure consistency and comparability in the results.

Results & Discussion

Table 1: Optimal models trained via supervised learning on 1000 images

U-NET	SAM-SEG	SAM-DET	RSPrompter
0.9312	0.9407	0.8477	0.9276

SAM-SEG and U-NET models both seem to have similar pixel-wise accuracy value, while SAM-DET does not perform as well, and has a lower accuracy score. However, visualising the images showed that U-NET does face a significant issue. Comparing U-NET against the other 2 models, all 3 models can give extremely accurate predictions for images containing only real features. Features on these images are reflected as objects within the ground truth. Example can be seen in images 1a, 1c, 1e and 1g in the appendix. However, there are harder images for these models to predict, such as images 1b 1d, 1f and 1h shown in the appendix, where despite an object-like feature being present in the image, this feature is not considered an object within the ground truth, and models have to learn to predict that there is no object present. This is very difficult for U-NET to do so, which might be due to U-NETs simplicity compared to the far more complex SAM-SEG. RSPrompter and SAM-DET also tend to label the entire object as the background in such images. It can also be seen that SAM-DET does not perform as well could be it is meant to do image detection and not image segmentation. As seen in 1e, it predicts the bounding boxes of the segmentation masks well but tends to fill them all in, resulting in lower accuracies than even U-NET.

Table 2: Effect of learning rate on pixel-wise accuracy. 1000 images were provided and 5 epochs were run. The value at the learning rate of 1e-4 for RSPrompter is empty as the RSPrompter training loop crashes when attempting to run it with a learning rate of 1e-4.

Learning rate	U-NET	SAM-SEG	SAM-DET	RSPrompter
1e-4	0.8814	0.9407	0.7259	na
1e-5	0.9312	0.8562	0.6096	0.8166
1e-6	0.9060	0.6231	0.3871	0.6537
1e-7	0.6609	0.4682	0.4321	0.6405
1e-8	0.6291	0.4679	0.4443	0.5743

We can see that for both the U-NET model and the SAM-SEG model, there is an optimal learning rate. This is expected, as too high a learning rate might result in the model oscillating around the optimal solution or even finding a suboptimal solution, while too low a learning rate would cause the model to improve too slowly. This optimal learning rate is 1e-5 for U-NET, and 1e-4 for SAM. Using this knowledge, we can use the best/second-best SAM model to conduct semi-supervised training in part 2 of this paper, which is obtained when learning rate is 1e-4 and 1e-5. For both SAM-DET and RSPrompter, it seems a higher learning rate simply results in a more accurate model. This is likely because even the highest learning rate of 1e-4 is not yet high enough to be the most optimal learning rate for the two models. Thus, it was simply decided to run more epochs for both models. The predicted masks for each model are with different learning rates are shown in Appendix images 2a, 2b, 2c and 2d: 4 images from the test dataset, along with the ground truth provided and the predicted mask from SAM-SEG and RSPrompter trained with different learning rates. One major issue that was identified with SAM-SEG was that should it be trained too slowly, or for too few epochs, it is unable to differentiate between the object (white) and the background (black). It can identify that the object is the background and vice versa. However, with enough training and sufficient prompts, the model learns to differentiate between object and background. One major issue with RSPrompter is that while it is good at differentiating the two classes, it sometimes mixes us the background with the object and will occasionally label the background as the object, resulting in it labelling the entire image wrong and having an accuracy of 0 for that image.

Table 3: Effect of the amount of training images provided on pixel-wise accuracy. Learning rate was set as 1e-5 before training, and 5 epochs were run.

Number of avr_m1 images	U-NET	SAM-SEG	SAM-DET	RSPrompter

provided for training				
200	0.9119	0.7103	0.3799	0.6996
400	0.9050	0.8173	0.6206	0.8875
600	0.9281	0.8219	0.5735	0.8865
800	0.9454	0.8523	0.5940	0.8583
1000	0.9397	0.8562	0.6096	0.7946

For the SAM-SEG model, as the amount of training data provided increases, accuracy increases but converges around 85%. This is expected, as more training data would allow the model to be trained on a wider range of images, and thus not overfit on a specific type of image. However, this also suggests that there is a limit to how much performance can be increased just by providing more training data. This could be due to some images providing noise during training, which would cause the model to train slower or even incorrectly. On the other hand, the U-NET model does not see very significant increases in accuracy as compared to SAM-SEGwhen more data is provided. This could suggest the U-NET model can learn from a smaller set of data and still perform quite well. For SAM-DET and RSPrompter, it is interesting tp note that using 400 training images results in better results than using all 1000. Perhaps this is due to the 400 images (selected in alphabetical order of their file names) including types of images that are more like those in the test dataset. Interestingly, for RSPrompter, the accuracy decreases as the number of images it is trained upon increases from 400 to 1000. Perhaps this is due to RSprompter overfitting to the smaller training data set and said smaller training data set having a great similarity to the test data set. Nonetheless, this could suggest the RSPrompter model can learn from a smaller set of data and still perform well. See Appendix A, Images 3a and 3b, for the predicted masks of two test dataset samples, along with the ground truth and the masks predicted by SAM-SEG trained on varying amounts of data. The main effect observed by reducing the amount of training data was lower pixel-wise accuracy on average for each mask. Some fine details, such as the exact edges shown above, were not recognized by the model trained on low amounts of data easily, and thus providing a far less accurate mask than one trained on much more data.

Part 2: Semi-supervised Learning

We selected the SAM-SEG model to conduct semi-supervised training. Instead of using 1000 labelled images for training, we would use 700, 800 and 900 labelled images to train for 5 epochs. Next, we used this model to output pseudo-masks for some of the unlabelled images. Finally, we trained the model on` some number of images such that the total number of images used was 1000. For example, if 900 labelled images were used for training, 100 unlabelled images would be used for training too. The same pixel-accuracy function as before was used together with 1000 images from the test dataset to evaluate model performance.

Results & Discussion

Table 4: Effect of adjusting ratio of labelled to unlabelled data used for training on accuracy

Labelled	Unlabelled	SAM-SEG
700	300	0.9011
800	200	0.9199
900	100	0.8888
1000	0	0.9407

We can see from the table of accuracy values and images that as more unlabelled data is used, the pixel-wise accuracy tends to decrease. This is expected, as the less the amount of labelled data used, the less accurate the model is after the first round of training (on labelled data). This then affects the quality of pseudo-masks obtained. During the second round of training, these less-accurate pseudo masks are used, which would negatively affect training and accuracy. However, when 800 labelled and 200 unlabelled images are used, a decent accuracy of 91% is obtained. This is quite a small drop in accuracy from supervised learning, while requiring less labelled data. This is significant, as it means that there is potential for semi-supervised learning to be used to reduce the amount of effort needed to label an image and identify manually the object within each image.

Conclusion

In conclusion, we successfully attempted to finetune 4 different models (U-NET, SAM-SEG, SAM-DET and RSPrompter) under different conditions. An accuracy of 94% was achieved. We have also shown that semi-supervised learning can be useful, as it allows for significantly less data that requires labelling, which saves effort and time. Further work could be done to see how greater amounts of labelled data and unlabelled data could increase accuracy. Clustering could also be done in order to select a more representative set of data for training under low-data conditions.

References

1. Chen, K. (n.d.). *RSPrompter: Learning to prompt for remote sensing instance segmentation*. GitHub. Retrieved December 22, 2024, from https://github.com/KyanChen/RSPrompter

2. Chen, K., Liu, C., Chen, H., Zhang, H., Li, W., Zou, Z., & Shi, Z. (2023). *RSPrompter: Learning to prompt for remote sensing instance segmentation based on visual foundation model*. arXiv. https://arxiv.org/pdf/2306.16269

3. Fortune Business Insights. (n.d.). *Integrated circuit market size, share and trends*. Retrieved December 22, 2024, from https://www.fortunebusinessinsights.com/amp/integrated-circuit-market-106522

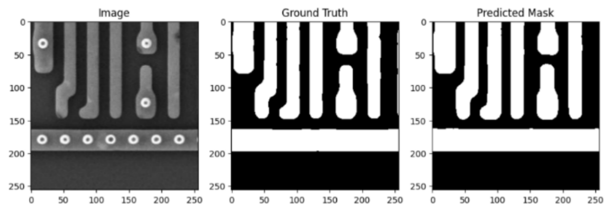
4.Sagieppel. (n.d.). *Fine-tune & train segment anything 2 in 60 lines of code*. GitHub. Retrieved December 22, 2024, from https://github.com/sagieppel/fine-tune-train_segment_anything_2_in_60_lines_of_code

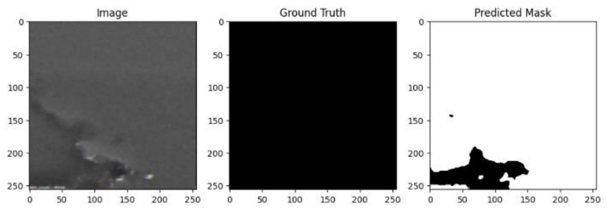
5. Ronneberger, O., Fischer, P., & Brox, T. (2015). U-NET: Convolutional networks for biomedical image segmentation. *arXiv*. https://arxiv.org/pdf/1505.04597

Appendix

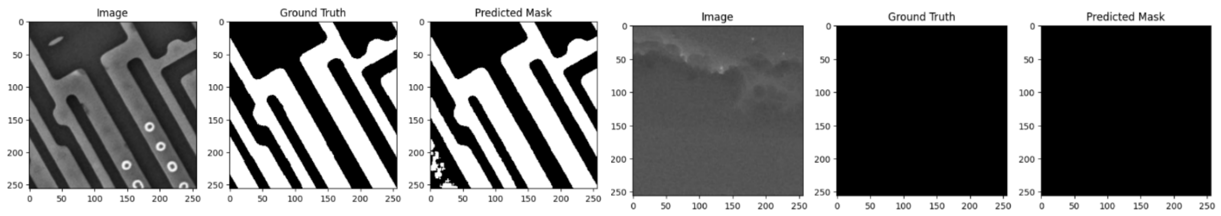
Images 1a, 1b, 1c, 1d, 1e and 1f: Predicted masks on different sample images using SAM, U-NET and RSPrompter models.

(From left to right) Images 1a and 1b: Predicted masks from U-NET model

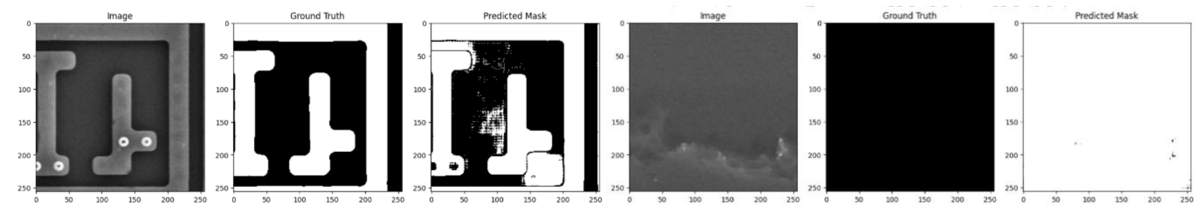




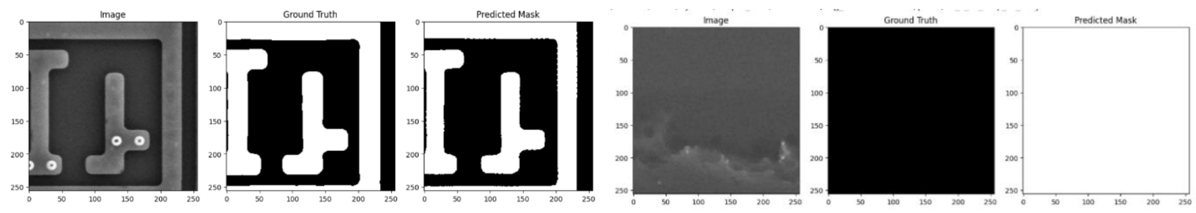
(From left to right) Images 1c and 1d: Predicted masks from SAM-SEG model



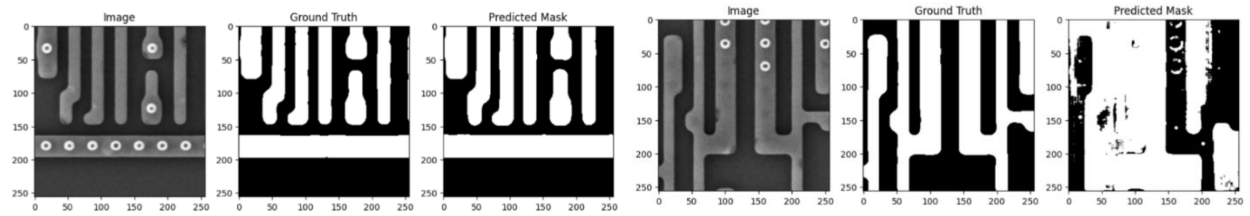
(From left to right) Images 1e and 1f: Predicted masks from SAM-DET model:



(From left to right) Images 1g and 1h: Predicted masks from RSPrompter model:



Images 2a and 2b: Predicted masks from SAM-SEG with initial learning rate of 1e-4 (left) and 1e-8 (right)



Images 2c and 2d: Predicted masks from RSPrompter with initial learning rate of 1e-5 (left) and 1e-8 (right)

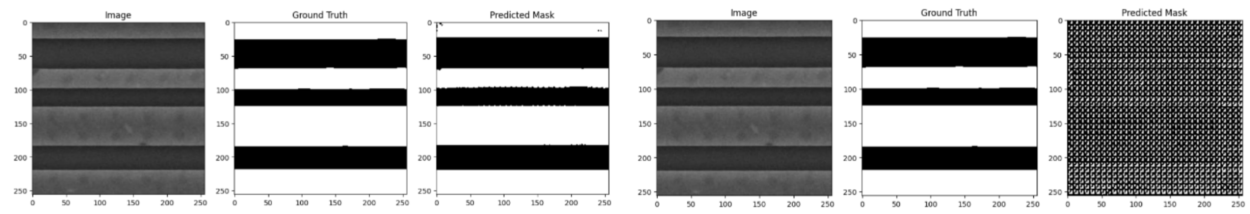
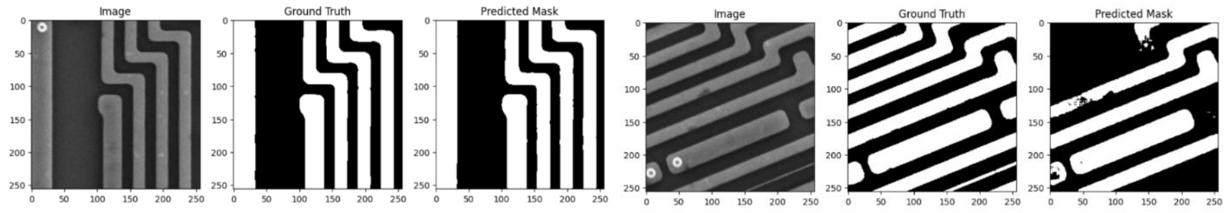


Image 3a and 3b: Predicted masks from SAM-SEG trained on 1000 images (left) and 200 images (right)



Images 4a and 4b: Predicted masks for two test dataset samples obtained after training SAM-SEG on 900 labelled images and 100 unlabelled images.