Self-Supervised Learning Applied to Mixed-Type Wafer Maps

Mohammed Faris Khan

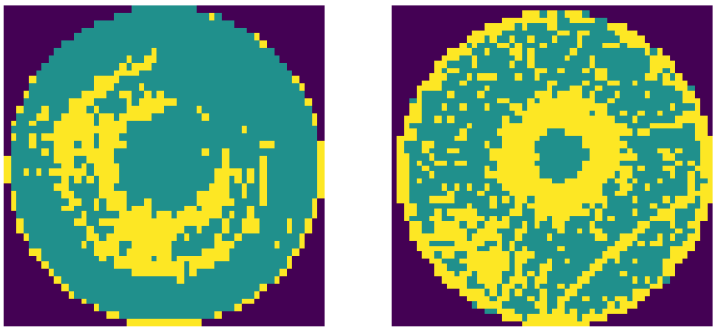
May 5, 2023

1 Introduction

Multiprobe testing is a critical component of semiconductor manufacturing. Following the back-end-of-line manufacturing processes, every die on a silicon wafer is put through a series of electrical tests to determine die yield. Some of these tests include simple tests for issues like short circuits, while others are more extensive, product-specific tests that verify that the die can perform all its more specialized functions [1]. In this series of functionality tests, a die is only deemed functional if it passes every test. Even a single failure means that the die is defective. The results of these multiprobe tests are typically displayed as wafer maps. These are simple images in which the pixel values correspond to a die's multiprobe testing results. In binary wafer maps, for example, non-wafer area is typically marked with 0's, while yielding die and defective die are marked with 1's and 2's, respectively. Binary wafer maps can be represented as simple grayscale images with discrete domains, i.e. $\{0, 128, 255\}$. Wafer maps provide a quick summary of which areas of a wafer have issues, and they are often the first step in identifying issues in fabrication processes. When similar patterns of failing die are observed on multiple wafers, it is often because they share a common root cause.

To streamline root cause analysis, semiconductor engineers are interested in performing "wafer map similarity searches," since wafers with similar defect patterns can be scattered across multiple lots in a process line. This needle-in-a-haystack problem is essentially an image retrieval task, but any sort of downstream data mining such as similarity search or defect classification of wafer maps requires meaningful vector representations of these images. Data is large and unlabeled in manufacturing settings, making supervised representation learning untenable. Self-supervised learning (SSL) has had remarkable success in this regard, since it enables strong representation learning without the need for large labeled datasets. SSL has had notable success on the WM-811K dataset [2], the largest publicly available dataset of real binary wafer maps. SSL leads to semantically meaningful representations of wafer maps, and self-supervised pretraining followed by fine-tuning leads to better performance than supervised baselines on classification tasks [3]–[5].

Unfortunately, the WM-811K dataset mostly has wafers with single-type defects, not mixed-type ones. In other words, most wafer maps in this dataset show only a single class of defects, such as scratch, center-localized, or edge-localized failures, but not combinations of *multiple* defects. Recently, a synthetic dataset known as MixedWM38 was created to this end; generative adversarial networks were trained on the WM-811K dataset to synthesize wafer maps with mixed defect patterns. Although this is a synthetic dataset, it offers an opportunity to explore how well SSL can be adapted for a more difficult learning problem. Namely, can SSL lead to strong *multilabel* classification performance on the MixedWM38 dataset? This has important practical implications for semiconductor manufacturing. Wafer maps with single-type defects are usually encountered in high-volume manufacturing scenarios where processes and products are mature. In product development, however, multiple process issues often need to be addressed simultaneously, leading to mixed-type wafer maps. It remains to be seen whether SSL can enable effective representation learning of such wafer maps. Moreover, there are several popular methods of SSL today that all have comparable performance on Imagenet-style datasets of natural images, but it is unclear whether every SSL technique can be used effectively on multilabel wafer map data.



(a) Single-type (b) Mixed-type

Figure 1: Examples of binary wafer maps with single-type and mixed-type defects. Here, green indicates passing die, yellow indicates failing die, and purple indicates non-wafer area. The single-type wafer map is a real wafer map from the WM-811K dataset, while the mixed-type wafer map is a synthetic image from MixedWM38.

2 Methods

2.1 Data

The MixedWM38 dataset was split into training, validation, and test splits corresponding to 70%, 15%, and 15% of the full dataset, respectively. While MixedWM38 is a multi-label dataset, there are 38 unique combinations of labels in the dataset that are fairly uniform in size.[1] Thus, data splitting can be performed in a stratified fashion to ensure independent and identically distributed splits for training, validation, and testing. For linear evaluations, smaller splits were created from the training data. The sizes of these smaller splits correspond to 1%, 5%, 10%, and 20% of the full dataset. Figure 2 displays an icicle plot of the data splits.

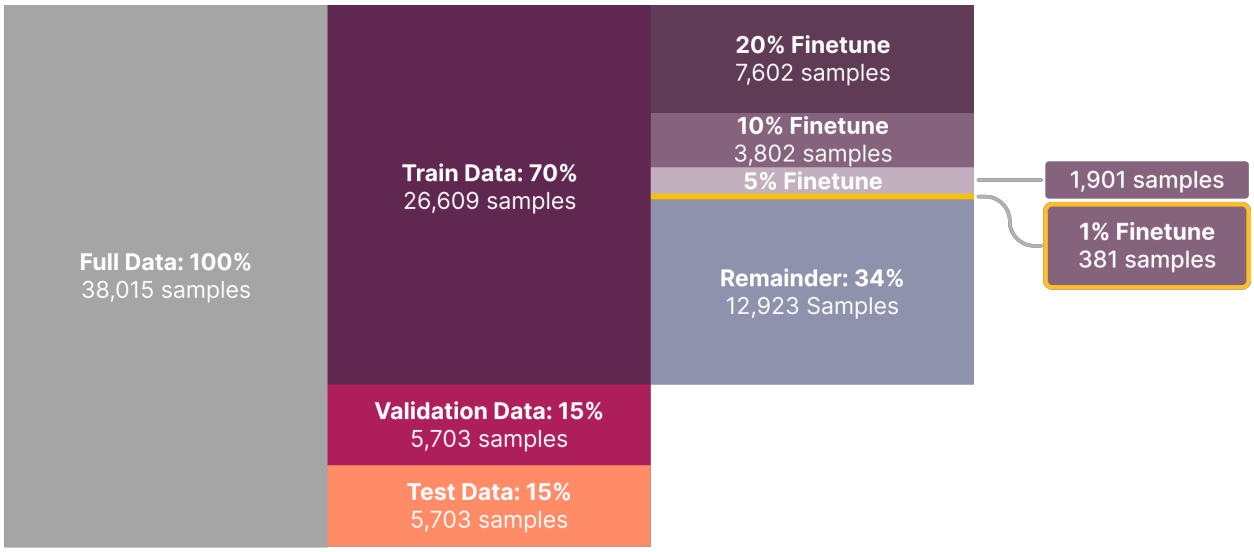


Figure 2: Hierarchical splits of the MixedWM38 dataset. Percentages denote the proportion of the full data.

[1] Aside: This is an unrealistic assumption made by the creators of MixedWM38. Categories of defect types are not equally sized in semiconductor manufacturing; some issues are seen far more often than others, such as center-localized failures vs. near-full failures

2.2 Investigating Multiple SSL Frameworks

Today, there are several popular techniques for self-supervised visual representation learning. At a high level, two categories have been most successful in recent years: masked image modeling and joint embedding. Several strategies of joint embedding exist to avoid trivial solutions and representation collapse. Contrastive learning is perhaps the most well-known, but non-contrastive techniques such as clustering, redundancy reduction, and self-distillation have recently received great interest. The specific frameworks investigated in this work are listed below, which represent some of today's most popular SSL techniques. A technical discussion of the exact details of each of these algorithms is beyond the scope of this work, and readers should consult the reference implementations instead. For those interested, high-level summaries of these techniques can be found in the excellent "Cookbook of Self-Supervised Learning" [6]. What matters is that these all work fairly well for natural images, but it is unclear how well they can be adapted to wafer map data. Hence, multiple techniques are investigated here.

- Joint Embedding:
  - Contrastive Learning: Decoupled Contrastive Learning (DCLW) [7]
  - Clustering: Swapping Assignment between Views (SwAV) [8]
  - Redundancy Reduction: Variance Invariance Covariance Regularization (VICReg) [9]
  - Self-Distillation:
    - Bootstrap Your Own Latent (BYOL) [10]
    - Self-Distillation with No Labels (DINO) [11]
  - "Hybrid" (mask denoising, clustering, distillation): Masked Siamese Networks (MSN) [12]
- Masked Image Modeling: Masked Autoencoders (MAE) [13]

DCLW, SwAV, VICReg, and BYOL use a ResNet-18 as the backbone encoder. DINO and MSN use a ViT-S/16, and MAE uses the much larger ViT-B/32 since masked image modeling is generally performed with larger ViT models. All frameworks were implemented using the `lightly` package for SSL using PyTorch Lightning [14], [15]. All self-supervised pretraining was done using 16-bit Automatic Mixed Precision for 150 epochs on the training data using a batch size of 64. For each framework, the specifics of the projection heads and optimization schemes are outlined below. These details are inspired by the reference implementations but modified slightly to run on consumer-grade hardware. They are the results of mostly empirical results from the reference implementations, since it is quite difficult to perform hyperparameter optimization for unsupervised deep learning. Moreover, the loss functions are different for each of these frameworks. Note that all models are optimized using a learning rate factor of `batch_size / 256`. Since a batch size of 64 is used here, all base learning rates specified below are multiplied by 0.25.

DCLW The projection head uses dimensions $(512, 512, 128)$, corresponding to the feature, hidden, and embedding dimensions, respectively. The SGD optimizer is used with a base learning rate of 6×10^{-2}, a momentum of 0.9, and a weight decay of 5×10^{-4}. A cosine annealing learning rate schedule is employed.

SwAV The projection head uses dimensions $(512, 2048, 128)$, corresponding to the feature, hidden, and embedding dimensions, respectively. 3000 prototypes are used, and the multi-crop scheme uses 2 high-resolution crops and 6 low-resolution crops. The Adam optimizer is employed with a base learning rate of 1×10^{-3} and a weight decay of 1×10^{-6}, using a cosine annealing learning rate scheduler.

VICReg The projection head uses dimensions $(512, 2048, 2048)$, corresponding to the feature, hidden, and embedding dimensions, respectively. The LARS optimizer is employed with cosine warmup, using a base learning rate of 0.3, a momentum of 0.9, and a weight decay of 1.5×10^{-6}.

BYOL The projection head uses dimensions $(512, 4096, 256)$, corresponding to the feature, hidden, and embedding dimensions. The prediction head uses dimensions $(256, 4096, 256)$, corresponding to the embedding, hidden, and prediction dimensions. The SGD optimizer is used with a base learning rate of 6×10^{-2}, a momentum of 0.9, and a weight decay of 5×10^{-4}, using a cosine annealing learning rate scheduler.

DINO The student and teacher networks' projection heads use dimensions $(2048, 256, 2048)$, corresponding to the feature, hidden, and embedding dimensions, respectively. No batch normalization is performed in the projection head. As with SwAV, 2 high-resolution crops are used along with 6 low-resolution crops. The AdamW optimizer is employed with cosine warmup, using a base learning rate of 1.5×10^{-4}, a momentum of 0.9, a weight decay of 0.05, and betas $(0.9, 0.95)$.

MSN The projection head uses dimensions $(384, 2048, 256)$, corresponding to the feature, hidden, and embedding dimensions, respectively. 3000 prototypes are used. Again, the AdamW optimizer is employed with cosine warmup, using a base learning rate of 1.5×10^{-4}, a weight decay of 0.05, and betas $(0.9, 0.95)$. For random masking, a masking ratio of 15% is used.

MAE The encoder is ViT-B/32. For the decoder, a ViT with a sequence length of 32, 1 layer, 16 heads, hidden dimension of 512, MLP dimension of 512×4, output dimension of $32^2 \times 3$, and no dropout is used. The AdamW optimizer is employed with cosine warmup, using a base learning rate of 1.5×10^{-4}, a weight decay of 0.05, and betas $(0.9, 0.95)$. For random masking, a masking ratio of 75% is used.

2.3 Augmentation Pipeline for Joint Embedding

Custom data augmentation policies are the key ingredient to effective self-supervised pretraining. In general, the augmentations used should match the set of invariances that need to be learned for downstream tasks. Not every augmentation technique is relevant for wafer map data, such as color jittering or Gaussian blur. Spatial transformations such as rotations and horizontal/vertical flips are employed since wafer map defect representations should be invariant to changes in the orientation of the defect pattern. Additionally, two domain-specific augmentation techniques are employed in this work to add or subtract "noise" from the defect patterns. To add noise, the randomized die noise augmentation proposed by Hu et al. [4] is used, in which passing and failing die on a wafer map have a fixed probability of being "flipped" to the opposite type. That is, a value of 128 in the image may be flipped to a 255, and vice-versa. Non-wafer area corresponding to zero-valued pixels is not affected. Here, each die has a 3% chance of being flipped to the opposite type, enabling the network to learn invariances to minor changes in defect patterns. To remove "salt-and-pepper" noise, each wafer map has a chance of being de-noised using median filtering with a kernel size of 3. This essentially removes all the "random" failures on the wafer map, keeping only spatially correlated or "connected" defects, such as large scratches or blotches on the wafer. The full randomized augmentation pipeline is outlined below, and Figure 3 provides example views which could be returned for a given image.

- Perform one of the following, with equal probability of choosing either:
  - Randomized die noise
  - De-noise using a median filter
- Resize the image to 224×224 using nearest-neighbor interpolation
- 50% chance of rotating by 90 degrees
- 50% chance of vertically flipping
- 50% chance of horizontally flipping
- Convert the image to a 3-channel image by copying the first channel three times
- Normalize the image

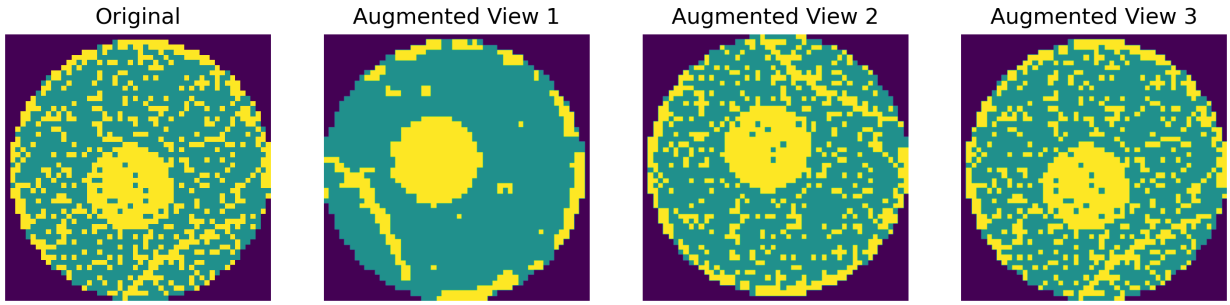


Figure 3: Examples of augmented views returned by the augmentation pipeline.

Images are resized to $3 \times 224 \times 224$ for compatibility with most popular backbone architectures. Although ResNets can be used with images of variable size, many ViTs expect images with these specific dimensions. Channel-wise pixel statistics were computed from the training data, and images from all data splits were normalized using a mean of $(0.4496, 0.4496, 0.4496)$ and a standard deviation of $(0.2926, 0.2926, 0.2926)$.

For joint embedding frameworks using mulit-crop augmentations, the parameters used here differ slightly from the reference implementations in SwAV, DINO, and MSN. The reference implementations will crop into images with variable aspect ratios, creating global crops with scale bounds $(0.4, 1.0)$ relative to the original image, and local crops with bounds $(0.1, 0.4)$. Here, square aspect ratios are maintained for all crops. The bounds of the global crop scales are changed to $(0.6, 1.0)$, leading to slightly less aggressive cropping. By default, no cropping is performed for the other joint embedding frameworks.

2.4 Linear Evaluation and Supervised Pretraining Protocol

To evaluate every self-supervised model, a linear classifier with 8 output neurons was trained on different label fractions (i.e. the 1%, 5%, 10%, and 20% splits) using *frozen features* from each model's backbone encoder. As a comparison, a ResNet-18 was trained in a supervised fashion on these same data splits. Early stopping was employed for the supervised ResNet and linear classifiers using the BCE loss on the validation dataset.

3 Experimental Results

3.1 Linear Evaluation

Table 1: Multilabel classification AUC obtained by finetuning frozen backbone encoders from self-supervised models on different label fractions. Performance is reported on the test set in all cases.

Framework	Backbone	Label Fraction			
		1%	5%	10%	20%
(Supervised)	ResNet-18	79.8	88.3	**98.4**	**99.0**
BYOL	ResNet-18	**91.7**	**93.6**	94.1	94.2
DCLW	ResNet-18	87.2	90.6	91.5	92.1
DINO	ViT-S/16	83.1	88.6	90.1	90.7
MAE	ViT-B/32	79.7	87.4	89.6	90.6
MSN	ViT-S/16	89.2	91.0	91.3	89.8
SwAV	ResNet-18	90.1	92.5	92.8	92.5
VICReg	ResNet-18	89.7	92.6	93.0	93.4

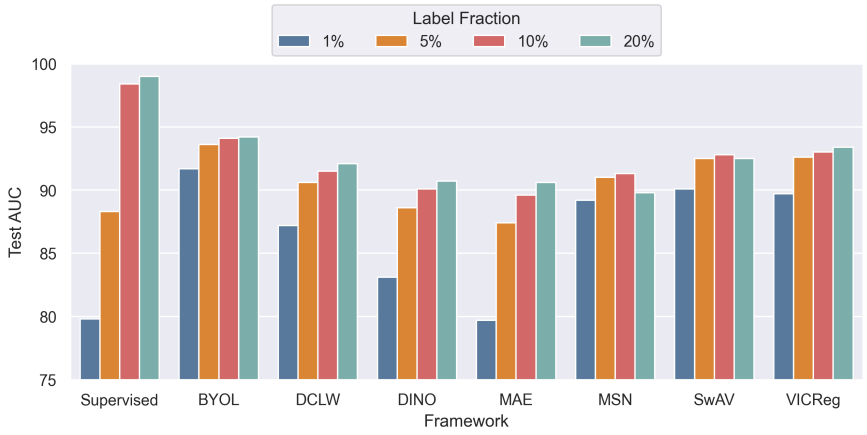
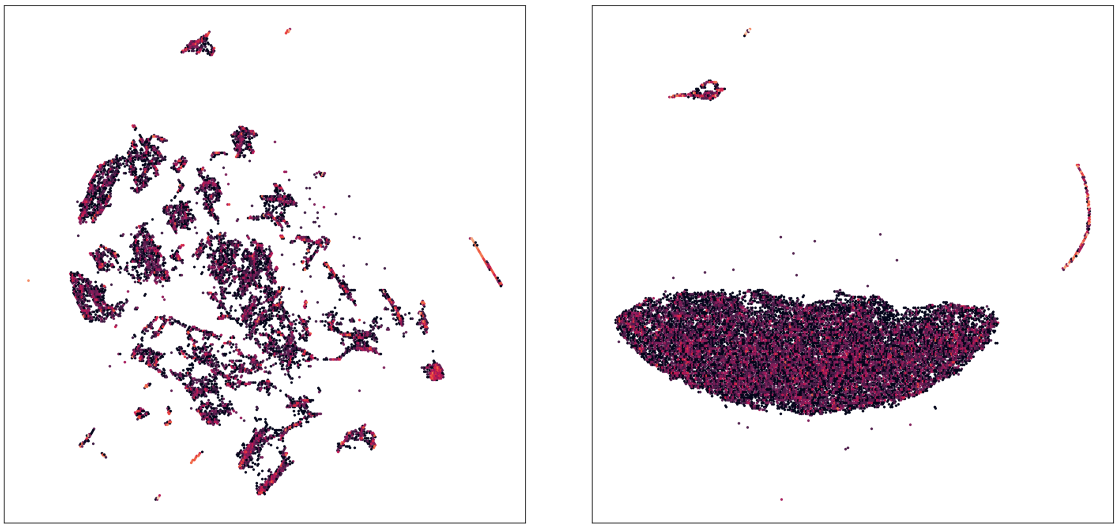


Figure 4: Barplot illustrating the scaling behavior of frozen features with different label fractions.

For the multilabel classification task on MixedWM38, self-supervised pretraining generally outperforms fully supervised pretraining in the regime of low label availability. Supervised pretraining outperforms SSL once 10% or more of the labels are available, though it should be noted that this could be due to the low model capacity of a linear classifier. The evaluation performance of self-supervised models would likely increase if the last few layers in the backbone networks were fine-tuned in addition to the classification head. Interestingly, linear probe performance for self-supervised models seems to plateau, while a fully supervised ResNet essentially memorizes the data and achieves near-perfect results with just 20% of the labeled data. Across different SSL techniques, there seems to be little difference in evaluation performance. DINO and MAE seemed to require more fine-tuning of the frozen features, which aligns with the observation that ViT-based methods generally require more fine-tuning for competitive downstream performance.

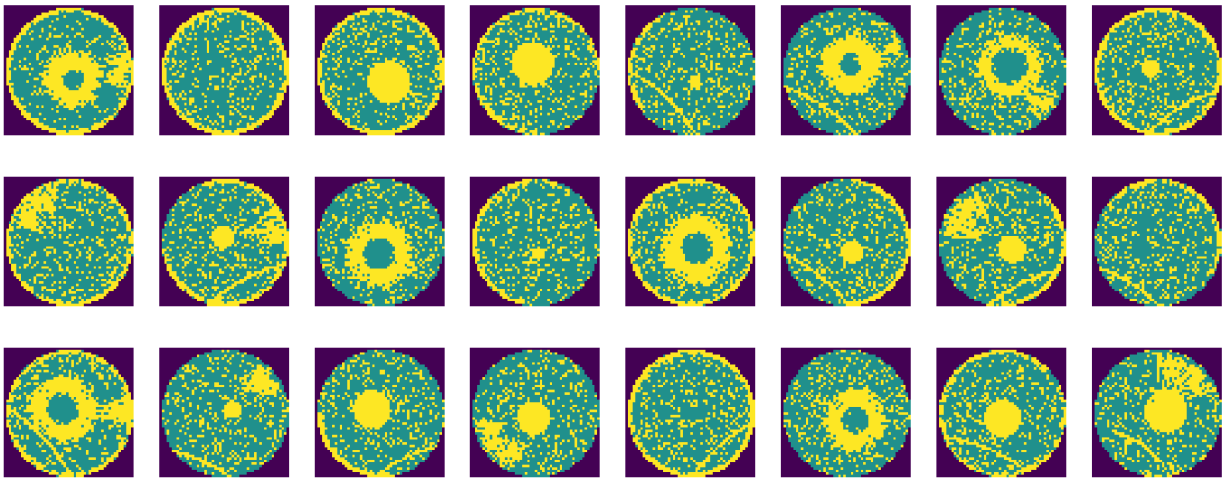
3.2 Qualitative Analysis of the Representation Space



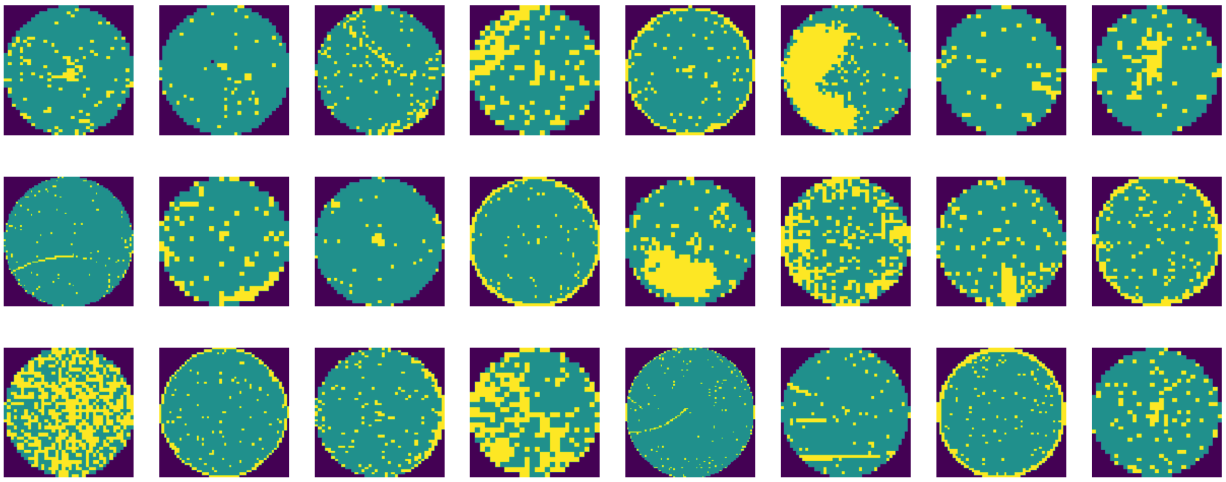
(a) VICReg (b) Supervised

Figure 5: UMAP embeddings of the representation space for VICReg's backbone and a supervised ResNet-18. In these hex bin scatter plots, the brighter colors indicate regions of higher density.

UMAP dimensionality reduction was performed on the representations from self-supervised backbone encoders as well as a supervised ResNet-18. Figure 5 qualitatively compares the representation space from the backbone of a ResNet-18 pretrained using the VICReg framework and a ResNet-18 trained in a typical supervised fashion on 20% of the full dataset. Although the supervised model benchmarked higher, the self-supervised feature space appears to be better separated. That is, categories obtained by self-supervision are well-separated without access to label information at all. This observation seems counter-intuitive, but even ImageNet weights for a ResNet-18 led to qualitatively similar results to the embedding shown in Figure 5b. This suggests that the classification performance gap between self-supervised pretraining and supervised pretraining is due to the low model capacity of the linear classifier. As a caveat, Figure 6 illustrates the fact that MixedWM38 is not a very diverse dataset, hence the poor feature space separability of the supervised model. MixedWM38 is a synthetic dataset, so results from this dataset may not apply well to real-world semiconductor manufacturing scenarios where defect patterns are more diverse and imbalanced.



(a) Randomly selected wafer maps from MixedWM38.



(b) Randomly selected wafer maps from WM-811K.

Figure 6: Comparison of randomly selected wafer maps from MixedWM38 and WM-811K show that the latter is much more diverse. Note that in the synthetic dataset, all scratches have the same shape, all "donut" patterns appear similar, and all circular "blotches" in the middle are nearly identical.

4 Conclusions

Self-supervised learning leads to effective representation learning of mixed-type wafer maps. Several techniques are successful here, such as contrastive learning, clustering, redundancy reduction, self-distillation, and masked-image modeling. The linear evaluation performance of self-supervised models exceeds that of supervised models in the low-label availability regime. The performance gap is closed once more labels are given to fully supervised models, but the evaluation performance of self-supervised models can be improved by performing finetuning of the last layers of the network as opposed to simple linear probing or using higher-capacity classifiers such as a two-layer network. Unfortunately, the MixedWM38 dataset is not very realistic. There is very little diversity in the defect patterns in this dataset, and the distribution of labels is uniform rather than tail-heavy, which is typical in semiconductor manufacturing. Nevertheless, the fact that self-supervised learning can learn at all from this dataset suggests that self-supervision can help enable better featurization of real wafer maps with mixed-type defects.

References

[1] K. A. Black, "Die level sorting of an integrated circuit," Thesis, Texas Tech University, Dec. 2000.

[2] M.-J. Wu, J.-S. R. Jang, and J.-L. Chen, "Wafer map failure pattern recognition and similarity ranking for large-scale data sets," *IEEE Transactions on Semiconductor Manufacturing*, vol. 28, no. 1, pp. 1–12, Feb. 2015, Number: 1 Conference Name: IEEE Transactions on Semiconductor Manufacturing, ISSN: 1558-2345. DOI: 10.1109/TSM.2014.2364237.

[3] M. F. Khan, "Self-supervised representation learning of semiconductor wafer maps," Thesis, University of Utah, Apr. 25, 2023, 56 pp.

[4] H. Hu, C. He, and P. Li, "Semi-supervised wafer map pattern recognition using domain-specific data augmentation and contrastive learning," in *2021 IEEE International Test Conference (ITC)*, ISSN: 2378-2250, Oct. 2021, pp. 113–122. DOI: 10.1109/ITC50571.2021.00019.

[5] H. Kahng and S. B. Kim, "Self-supervised representation learning for wafer bin map defect pattern classification," *IEEE Transactions on Semiconductor Manufacturing*, vol. 34, no. 1, pp. 74–86, Feb. 2021, Conference Name: IEEE Transactions on Semiconductor Manufacturing, ISSN: 1558-2345. DOI: 10.1109/TSM.2020.3038165.

[6] R. Balestriero, M. Ibrahim, V. Sobal, *et al.*, *A cookbook of self-supervised learning*, Apr. 24, 2023. DOI: 10.48550/arXiv.2304.12210. arXiv: 2304.12210[cs].

[7] C.-H. Yeh, C.-Y. Hong, Y.-C. Hsu, T.-L. Liu, Y. Chen, and Y. LeCun, *Decoupled contrastive learning*, Issue: arXiv:2110.06848, Jul. 29, 2022. DOI: 10.48550/arXiv.2110.06848. arXiv: 2110.06848[cs].

[8] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, *Unsupervised learning of visual features by contrasting cluster assignments*, Issue: arXiv:2006.09882, Jan. 8, 2021. DOI: 10.48550/arXiv.2006.09882. arXiv: 2006.09882[cs].

[9] A. Bardes, J. Ponce, and Y. LeCun, *VICReg: Variance-invariance-covariance regularization for self-supervised learning*, Issue: arXiv:2105.04906, Jan. 28, 2022. DOI: 10.48550/arXiv.2105.04906. arXiv: 2105.04906[cs].

[10] J.-B. Grill, F. Strub, F. Altché, *et al.*, *Bootstrap your own latent: A new approach to self-supervised learning*, Issue: arXiv:2006.07733 Number: arXiv:2006.07733, Sep. 10, 2020. DOI: 10.48550/arXiv.2006.07733. arXiv: 2006.07733[cs,stat].

[11] M. Caron, H. Touvron, I. Misra, *et al.*, *Emerging properties in self-supervised vision transformers*, Issue: arXiv:2104.14294 Number: arXiv:2104.14294, May 24, 2021. DOI: 10.48550/arXiv.2104.14294. arXiv: 2104.14294[cs].

[12] M. Assran, M. Caron, I. Misra, *et al.*, *Masked siamese networks for label-efficient learning*, Issue: arXiv:2204.07141, Apr. 14, 2022. DOI: 10.48550/arXiv.2204.07141. arXiv: 2204.07141[cs,eess].

[13] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, *Masked autoencoders are scalable vision learners*, Issue: arXiv:2111.06377, Dec. 19, 2021. DOI: 10.48550/arXiv.2111.06377. arXiv: 2111.06377[cs].

[14] I. Susmelj, M. Heller, P. Wirth, J. Prescott, and M. Ebner, *Lightly-ai/lightly*, original-date: 2020-10-13T13:02:56Z, Apr. 2, 2023.

[15] W. Falcon, J. Borovec, A. Wälchli, *et al.*, *PyTorchLightning/pytorch-lightning: 0.7.6 release*, version 0.7.6, May 15, 2020. DOI: 10.5281/ZENODO.3828935.