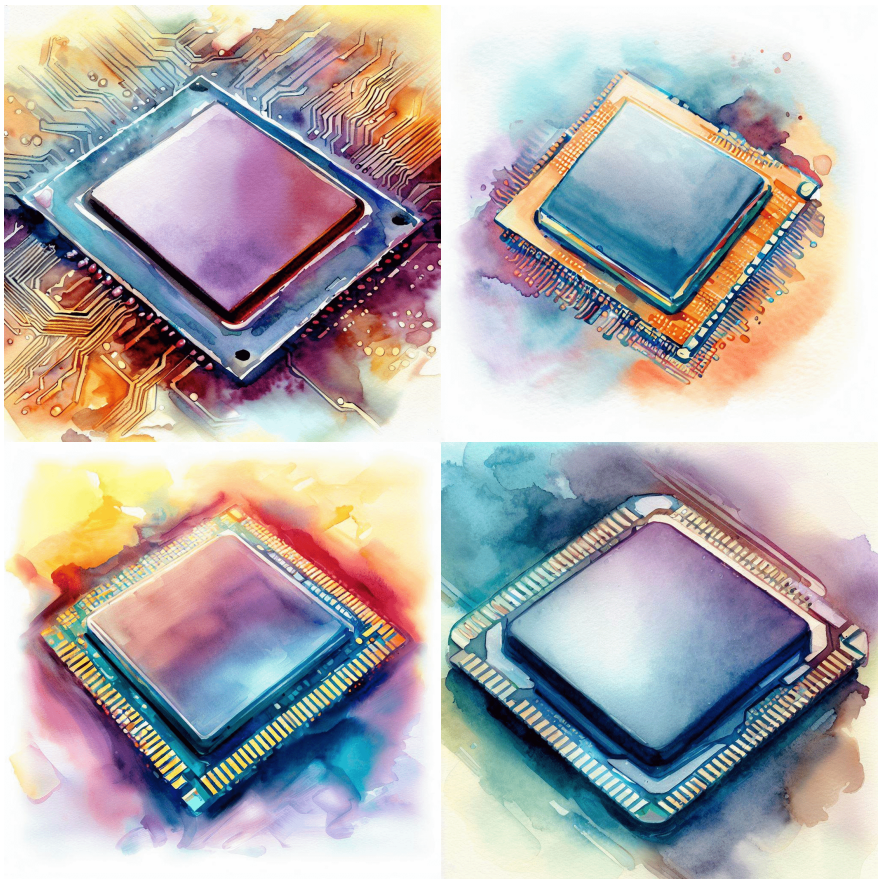
Self-Supervised Representation Learning of Semiconductor Wafer Maps

M.S. Project Report

University of Utah,
Department of Materials Science and Engineering



Mohammed Faris Khan

May 21, 2023

Approved:

Dmitry Bedrov (Committee Chair)

Taylor Sparks

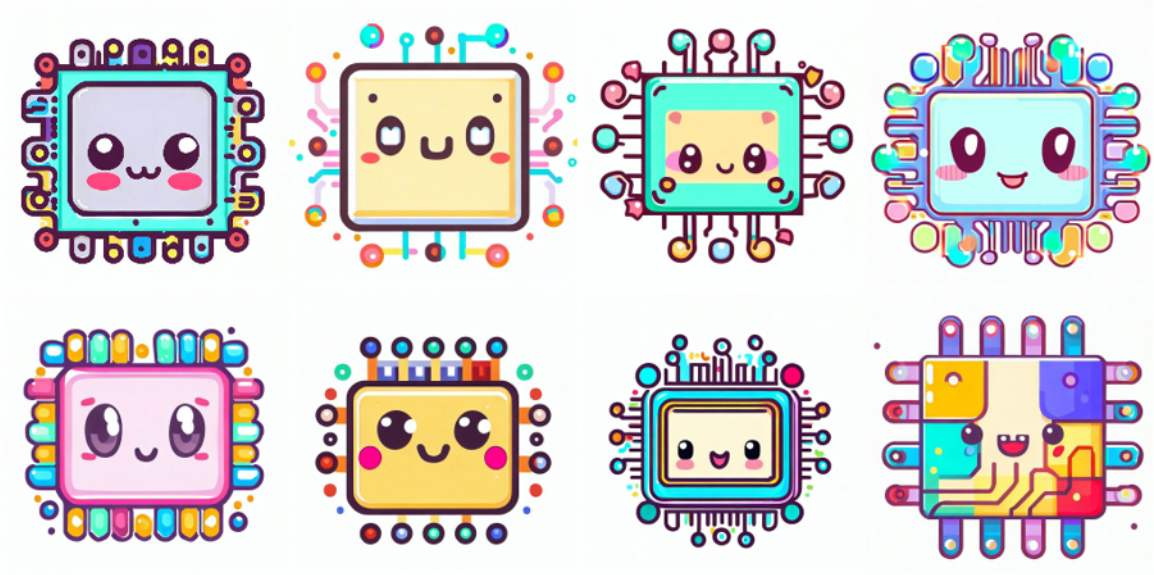
Michael Scarpulla

Acknowledgments

This project would not have been possible without the guidance and support I received. I would like to thank my colleagues at Texas Instruments for their invaluable input and feedback throughout this project. In particular, I am grateful to my manager Scott McClendon and my fellow data scientists Suku Kim and Zhou Fang for their expertise and encouragement. I would like to acknowledge all my professors and teachers who helped me get to where I am today, particularly Berton Earnshaw for his advice on this project and my high school teacher Phil Talbot for instilling in me a passion for science. I would like to thank Patsy Cadareanu for mentoring me in the Utah nanofab, my first experience with semiconductors, and I would also like to thank Kaai Kauwe and Taylor Sparks for their mentorship and guidance when I pivoted to data science almost three years ago. I also appreciate the questions and insights from my fellow graduate students Ramsey Issa and Sterling Baird. I'm thankful for the assistance of students who I've worked with on projects related to semiconductor data science over the last year, as these have helped me understand the value of data in the semiconductor industry. Thank you Alan Weber and Filemon Mateus for working with me on visualizing semiconductor data, Hibban Butt for working with me on implementing FastSiam, and Holden Ellsworth and Zahra Gholamishiri for analyzing self-supervised representation spaces with me.

I owe a special debt of gratitude to the open-source software community for providing me with the tools and resources to complete this project independently. As a materials science student with little formal training in software, I relied heavily on the online tutorials, blogs, videos, and code repositories that generously shared the latest developments and best practices in machine learning. I am amazed by the openness and collaboration that characterize this field and how committed the community is to the democratization of machine learning. To all the researchers who made their work accessible and reproducible, thank you!

Last but not least, I thank my family for their unconditional love and support. I literally worked on this project in my mom's basement, and the power draw of an RTX 3080 Ti while it trains large neural networks for days on end is nothing to scoff at! Mama, I promise the power bill will go down after this semester!



These friendly computer chips have gathered here to thank everyone who helped me along the way. To the reader, these chips gleefully welcome you and serve as a final respite before a far more technical discussion.

Abstract

In semiconductor manufacturing, wafer maps are used to summarize which die on the wafer are defective. As image data, wafer maps must first be featurized into vector representations for downstream data mining tasks such as similarity search or classification. Supervised representation learning is untenable in manufacturing settings because wafer map data is often unlabeled. Self-supervised learning (SSL) offers a promising alternative without the need for large labeled datasets. Prior work on the WM-811K wafer map dataset has largely focused on fine-tuning classifiers on top of self-supervised pretrained encoders to achieve state-of-the-art classification performance. Here, the limits of SSL on wafer map data are investigated by determining whether the raw representation vectors obtained from self-supervised pretrained encoders are semantically meaningful without any supervised fine-tuning. To this end, a k-NN classification benchmark is used to evaluate more than a dozen popular joint embedding and masked image modeling frameworks. These frameworks were originally designed for SSL on ImageNet-style datasets of class-balanced natural images, so it is investigated how well they can be adapted to highly imbalanced, uncurated datasets like WM-811K. Quantitatively and qualitatively, it is demonstrated that SSL is an effective way to learn visual representations of wafer maps without labels. Importantly, the features obtained through self-supervised pretraining are semantically meaningful before fine-tuning, as demonstrated by strong k-NN evaluation performance. Moreover, qualitative analyses of activations and attention maps show that self-supervised models "see" the salient portions of individual wafer maps. At a dataset level, the representation space is well separated. Through similarity search akin to content-based image retrieval, wafers that are close in the self-supervised representation space have visually consistent defect patterns. Code and data are available at https://github.com/faris-k/self-supervised-wafermaps.

Keywords: self-supervised learning · visual representation learning · semiconductor manufacturing · wafer bin maps · defect classification · contrastive learning · masked image modeling

Table of Contents

List of Figures

List of Tables

Acronyms

BERT Bidirectional Encoder Representations from Transformers. 6
BYOL Bootstrap Your Own Latent. 13

CNN Convolutional Neural Network. 3

DCL Decoupled Contrastive Learning. 11
DINO Self-**Di**stillation with **No** Labels. 14
DPW die-per-wafer. 21

GPT Generative Pre-trained Transformer. 6

MAE Masked Autoencoders. 7
MoCo **Mo**mentum **Co**ntrast. 10
MSN Masked Siamese Networks. 16

NLP Natural Language Processing. 6

PIRL Pretext-Invariant Representation Learning. 10

SIFT Scale-Invariant Feature Transform. 2
SimCLR **Sim**ple Framework for **C**ontrastive **L**earning of Visual **R**epresentations. 10
SimMIM **Sim**ple Framework for **M**asked **I**mage **M**odeling. 8
SimSiam **Sim**ple **Siam**ese Networks for Visual Representation Learning. 13
SSL Self-Supervised Learning. 5
SwAV **Sw**apping **A**ssignment between **V**iews. 11

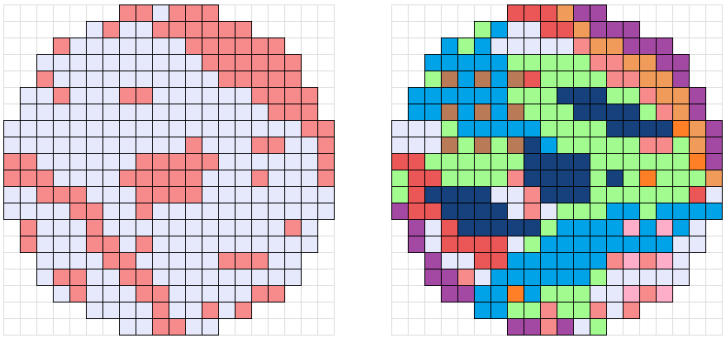
VICReg Variance-Invariance-Covariance Regularization. 15
ViT Vision Transformer. 7

1 Introduction and Motivation

Nearly every aspect of modern life is affected in some way by semiconductors. Major advances in industries such as healthcare, computing, transportation, communications, and even agriculture have been made possible by semiconductor devices. In semiconductor manufacturing, efficiency and quality is key, since end products often have strict standards for reliability and performance. Several semiconductor devices are trusted with human life, such as the chips inside pacemakers or automotive braking systems.

Inspecting device functionality after fabrication is therefore a critical component of semiconductor manufacturing. Multiprobe testing, also known as electrical die sorting, circuit probe, and wafer testing, is a die-level process that follows the back-end-of-line manufacturing processes and precedes integrated circuit packaging. Every die on the wafer is run through a series of electrical tests that determine whether the die is functional. 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 multiprobe tests are typically displayed as wafer maps to summarize die yield. Binary wafer maps like the one in Figure 1a only display whether the die are fully functional or defective. Thus, they show *where* the defects occur, but not necessarily which multiprobe tests the die failed. These are typically used in high-volume manufacturing scenarios when products and processes are more mature. In product development, however, there are often multiple process issues that need to be addressed simultaneously, so multi-bin wafer maps like the one in Figure 1b are often used to not only show where the defects occur, but also *how* the die are failing. This is typically done by providing a code corresponding to the specific multiprobe test which failed [2]. Both types of wafer maps provide a quick summary of which areas of the wafer have issues. Moreover, they often serve as 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.



(a) (b)

Figure 1: Examples of a binary wafer map (a) and a multi-bin wafer map (b).

1.1 Featurization of Wafer Maps

In semiconductor manufacturing, efficiently searching through thousands of wafer maps to identify similar ones can streamline root cause analysis. However, wafer maps must first be encoded into semantically

meaningful vector representations to effectively perform data mining tasks such as similarity search and defect classification. Simply computing element-wise differences between wafer maps is not an effective way to perform data mining. This is because small changes in spatial defect patterns, such as slight rotations, would lead to artificially high distances between similar wafer maps.

Manual Feature Engineering An engineer's first instinct might be to manually perform feature engineering using domain knowledge to obtain vector representations of wafer maps. Each dimension of this vector would be a number describing some aspect of the wafer map. For example, a hand-crafted representation vector might include die size, yield percent, and fraction of die failing a particular probe test. Although manual feature engineering leads to highly interpretable representations, this approach has two main disadvantages. First, each feature must be independent of the others and relevant to the downstream task. It can be challenging to manually define sufficiently many independent and discriminative features, and it is often unclear how granular these representations must be for them to be relevant to downstream tasks. Second, for downstream data mining, it is necessary to have a suitable distance metric that is compatible with the feature vector. Euclidean distance, for example, is not valid for feature vectors where the dimensions have different scales. It can be challenging to ensure that hand-crafted representations are both informative and compatible with typical distance metrics for use in data mining tasks such as clustering, similarity search, and anomaly detection.

Classical Image Analysis A better alternative to hand-crafted features is to treat wafer maps as image data and use featurization techniques from classical computer vision. Just as digital images are represented as tensors of pixel values, wafer maps spatially encode the results of multiprobe tests, usually in two-dimensional matrices. Some of the earliest large-scale featurization efforts on wafer map data closely resemble historical developments in computer vision. In fact, the development of the largest binary wafer map dataset WM-811K [3] was accompanied by a novel rotation- and scale-invariant featurization technique that closely resembled the scale-invariant feature transform (SIFT) algorithm from computer vision [4], [5]. SIFT itself can be applied to wafer map data to extract keypoints and descriptors as shown in Figure 2. Feature vectors can then be created using techniques such as the bag-of-visual-words model [6], and these vectors can easily be used as Euclidean vectors for downstream analysis. The contents of these representation vectors are not as easily interpretable, since each feature dimension no longer corresponds to manually specified descriptors of the data. Despite this, feature extraction based on computer vision typically produces far more robust representations than manual feature engineering, and it is much easier to scale to larger datasets.

Figure 2: SIFT keypoints overlaid onto a wafer map.

Deep Learning In the last decade, the revolution in machine learning has more or less led to the replacement of classical feature extraction techniques like SIFT with deep learning. In particular, convolutional neural networks (CNNs) have achieved remarkable success in image analysis. Typically, deep learning approaches to image featurization involve either transfer learning or representation learning from scratch.

Transfer learning involves using neural networks that have been trained on large image datasets such as ImageNet as either initialization points for further network fine-tuning or as fixed encoders to directly encode images as vector representations [7]. Using CNNs as fixed encoders assumes that the dataset for downstream analysis is very similar to the source dataset used for pretraining. With ImageNet weights, this assumption holds true for datasets of natural images, but this is usually not an effective means of representing out-of-domain data. Many filters learned by ImageNet-pretrained CNNs may be irrelevant to the downstream task, such as filters adapted to detect dogs or cars [8]. For image datasets that are slightly different from ImageNet, fine-tuning the later stages of pretrained CNNs on the transfer dataset can help alleviate this issue. This is why ImageNet weights can often be successfully used on tasks such as medical image analysis [9].

However, if the dataset is extremely different from ImageNet, transfer learning offers little to no advantage over representation learning from scratch. This is the case for wafer map data, where the only relevant filters from ImageNet pretraining would be those used for simple edge detection. The most popular approach to representation learning of wafer map data is to employ supervised pretraining. CNNs have widely been applied to the WM-811K wafer map dataset in this manner. Dozens of papers are published every year claiming state-of-the-art classification performance by making minor modifications to network architectures and training procedures [10], [11]. Because these developments rely purely on supervised learning, they fail to address an important issue in real-world semiconductor data. Wafer map data is typically unlabeled, so it is not straightforward to approach representation learning in a supervised fashion. It might seem simple to just label the unlabeled dataset and continue with supervised or semi-supervised learning like normal, but this approach is not always the best.

1.2 Drawbacks of Supervised Learning

Annotations are Flawed

In the context of supervised learning, annotations have several issues. One of the most common criticisms of supervised learning is that manually annotating large datasets is extremely expensive, so supervised learning is hard to scale to large-data regimes. Although this is true, machine learning does not require datasets to be entirely labeled. Semi-supervised learning, for example, can make use of both labeled and unlabeled data. The issue with supervision has more to do with the fundamental limitations of annotations. Annotations are inherently subjective. Even domain experts will disagree on how to label large datasets and with what granularity to perform such labeling. For example, the number of categories for a classification task might be ambiguous, and it may be unclear whether to treat the problem as a multi-label classification task. Considering the case of a discrete classification task with N proposed categories, it is almost always possible to create $N-1$ categories by merging two similar categories together, or $N+1$ categories by splitting an existing category apart.

Ambiguities in labeling can be illustrated by analyzing the categories of the WM-811K dataset. The wafer maps in this dataset are split into nine classes as shown in Figure 3. There is no reason to assume that defect

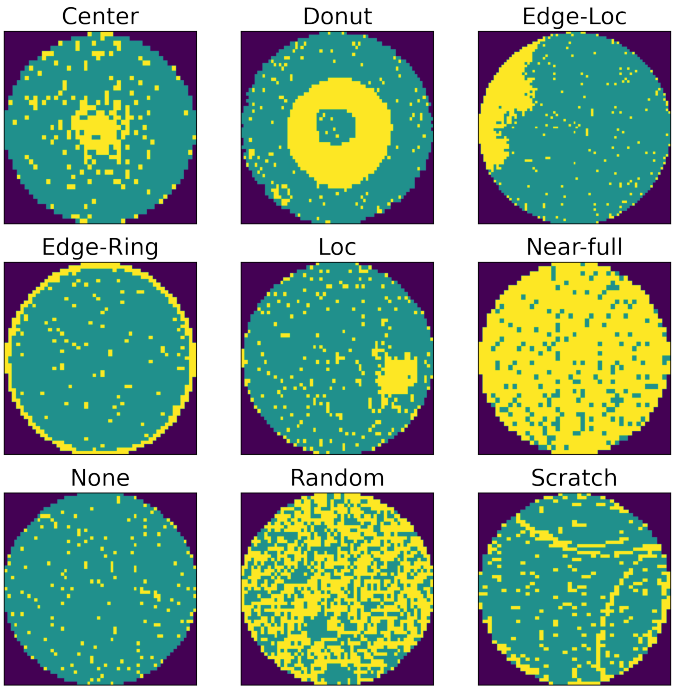


Figure 3: Example wafer maps from the nine categories of the WM-811K dataset. Green denotes passing die, yellow denotes failing die, and purple denotes non-wafer area.

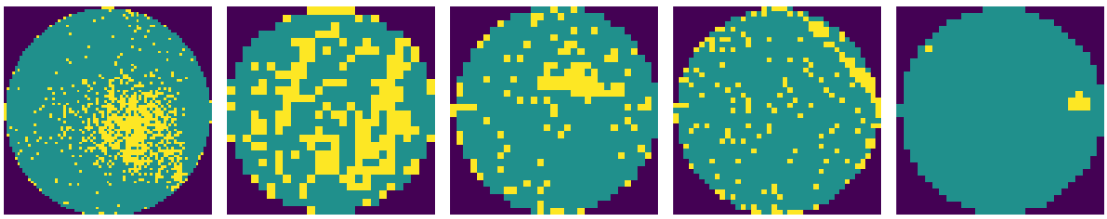


Figure 4: Example wafer maps from the "Loc" category.

patterns in wafer maps will always fall into nine categories like these. Furthermore, certain categories are filled with wafer maps that visually appear dissimilar. This is a major problem with these annotations, since wafers with a similar root cause usually have similar patterns of failing die. Figure 4 shows visually dissimilar wafer maps of the "Loc" class, which is a vague term indicating "localized" defects on a wafer. Some of these wafers could arguably be better described by other labels such as "Edge-Loc" for defects localized near the edge of the wafer or "Center" for failing die predominantly near the center of the wafer. Regardless of what class they should belong to, it is unlikely that the root cause of these wafers' defect patterns is the same, since the patterns themselves are very different.

These ambiguities in annotations highlight the fact that top-down categorization of diverse datasets often leads to unclear labels. Large datasets have diverse, hierarchical categories, and annotations should reflect this. With the "Loc" wafers from before, it could be reasonable to say that they belong to a supercategory of wafers with defective die localized within certain regions. Within this large category, however, there should be finer subcategories reflecting the intricacies of the data. Hierarchical categorization is one of the strengths of ImageNet, which contains supercategories like "dog" but also subcategories like "husky" and "German

shepherd" [7]. Unfortunately, labeling large datasets in this hierarchical manner is much more difficult than simply partitioning them into large supercategories. This is why many labeled datasets suffer from label bias, where the set of labels used is not fully representative of the entire universe of potential labels.

Supervision Leads to Specialization, not Generalization

Although supervised learning is a successful paradigm in highly specialized task-specific learning, it is difficult for supervised models to generalize to semantically distant concepts. Concept generalization includes transferring knowledge to different datasets and different tasks. For a fixed task, cross-dataset generalization is often difficult because supervised models struggle with distribution shifts [12]. In supervised learning, a labeled dataset is typically separated into independent and identically distributed training and test splits. This paradigm assumes that at inference time, the distribution of unseen data will be similar to that of the training data. When this is not the case, supervised pretraining generally leads to poor transfer performance, even if the transfer dataset is semantically similar to the training dataset. For example, models trained on CIFAR-10 [13] have been shown to generalize poorly to the conceptually similar Tiny Images dataset [14], which illustrates that even natural variations in image data and minor differences in data distributions can affect supervised models [15].

Additionally, supervised learning typically assumes a fixed task during training. Image datasets can often be used for multiple tasks such as image classification, instance segmentation, and image captioning. If the data itself remains semantically consistent, then why should entirely different sets of labels be used to train entirely different models? The point of supervised learning is not necessarily to learn generalizable, task-agnostic representations of data. Rather, the goal is to simply match data to labels that are best aligned with a given prediction task.

1.3 Moving Away from Labels with Self-Supervision

Recently, Self-Supervised Learning (SSL) has received great interest in unsupervised visual representation learning. In contrast to semi-supervised learning, the only source of supervision during training is the data itself. SSL aims to learn semantically meaningful representations that can generalize across datasets and tasks, and it has had massive success in this regard on ImageNet-style data. However, ImageNet is not representative of all datasets. It is a class-balanced, highly curated set of natural images. Everything from the network architectures to the training recipes that have proven successful in SSL are highly optimized for this specific dataset bias. Still, numerous studies have shown that SSL is more robust to class imbalance and distribution shifts than supervised learning [12], [16]–[18]. Additionally, careful changes to network architectures and loss functions have enabled SSL to be successful on more diverse, uncurated datasets, with notable works including PMSN [19], SEER [20], and RoPAWS [21]. These developments highlight the potential of using SSL in domains where data is large, unlabeled, and uncurated, such as semiconductor manufacturing.

Only the earliest breakthroughs in self-supervised learning have been applied to wafer map defect analysis. Specifically, these include pretext learning by Kahng et al. [22] and contrastive learning by Hu et al. [23]. Both works successfully demonstrated that self-supervised pretraining followed by supervised fine-tuning leads to strong classification performance on the WM-811K dataset. However, there have been significant developments in SSL since these two works. Additionally, it has not been investigated whether

the representations obtained via self-supervised pretraining are semantically meaningful for data mining of wafer maps *without* supervised fine-tuning.

This thesis thoroughly investigates how well the most recent advances in SSL apply to wafer map data, focusing specifically on the efficacy of SSL for learning representations that are useful without downstream supervision. Intense emphasis is placed on probing the semantics of self-supervised feature spaces to understand each SSL framework's limits and answer the following fundamental question: how much can self-supervised models learn about wafer map data without being explicitly told what to look for?

2 Review of Self-Supervised Learning of Images

Over the years, SSL has been defined in various ways and associated with several seemingly disparate machine learning methods. If self-supervision involves learning without labels, what separates it from unsupervised learning? Unsupervised learning is an extremely broad term, essentially referring to anything that is *not* supervised learning. Thus, it encompasses a variety of probabilistic and non-probabilistic learning techniques such as clustering, dimensionality reduction, and representation learning, none of which use labels [24]. On the other hand, self-supervision mainly concerns representation learning. The distinction between "self-supervised" and "unsupervised" is to emphasize that self-supervised models learn *supervisory signals* using just the data itself instead of external labels. A supervisory signal is a type of feedback used to train a machine learning model. It provides information about the correct output for a given input, allowing the model to adjust its internal parameters and improve its predictions over time. In supervised learning, supervisory signals are derived by measuring the compatibility between predictions and ground truth labels. In SSL, supervisory signals are derived from the data by measuring the compatibility between different *representations* of the data, such as how compatible a predicted missing word is with the rest of a sentence, or how compatible two embeddings are for augmented views of the same image. A model that learns with this type of feedback signal has not been *label-supervised*. Rather, it has been *self-supervised*.

SSL has been most successful in Natural Language Processing (NLP), in which the SSL process is typically framed as predicting some hidden portion of the input data using the rest of it. This includes masked autoencoding in frameworks such as BERT [25], or autoregressive modeling as in GPT [26]. Because annotating extremely large corpora of texts is practically impossible, SSL has become the top choice for representation learning of text. In fact, the power of recent large language models such as LLaMA and GPT-4 is largely due to their self-supervised pretraining and not their subsequent supervised fine-tuning [27]–[29].

For SSL on image data, the earliest forms of generative modeling included denoising autoencoders [30], in which corrupted images would be encoded into a latent representation and then decoded to a reconstructed image. The earliest non-generative SSL techniques in vision largely involved pretext learning, in which networks are trained to learn representations by solving "pretext tasks" which are unrelated to downstream categorization tasks but hopefully help the model learn good visual representations. Examples include image rotation prediction [31], jigsaw transformation prediction [32], and grayscale image colorization [33].

Unfortunately, both pretext learning and denoising autoencoding lead to representations that tend to be better aligned with solving pretext tasks and reconstruction tasks rather than downstream categorization tasks such as image classification [34]. Today, these methods have largely been replaced by their more powerful extensions. In generative modeling, denoising autoencoders have largely been replaced by masked

autoencoders, while in non-generative modeling, joint embedding has overtaken pretext learning.

2.1 Masked Image Modeling

In NLP, masking has been an extremely successful form of self-supervision [25]. In vision, however, generative and autoregressive modeling via masking has historically proven much more difficult. With text, the entire set of possible words is discrete. Thus, it is possible to produce a distribution over a finite vocabulary and predict, for example, which word out of $50,000$ possible words should occur next. On the other hand, masked prediction for images involves a continuous signal rather than a discrete text token. Consider predicting the RGB pixel values of a small 3×3 masked patch in an image. With 256 possible values for each color channel, there would be a total of $3 \times 3 \times 256^3 = 150,994,944$ possible values in just this small region. Additionally, visual signals and text involve very different levels of semantic abstraction. A word can capture high-level semantic concepts, while a single pixel value cannot do the same because it is raw and low-level. Only very recently have the methods from NLP been successfully adapted to handling SSL of vision. These techniques generally involve much higher degrees of masking than in NLP, and they use regression rather than classification as the prediction task.

MAE Inspired by the masked autoencoding of the BERT language models, MAE introduces masked autoencoding for images [35]. Figure 5 illustrates the architecture of MAE. Input images are first divided into a grid of image patches, and a large portion of the image patches ($\sim 75\%$) are randomly masked with a constant pixel value. Then, a vision transformer (ViT) [36] is used to encode only the unmasked patches into a sequence of embeddings. The masked tokens are used to pad the unmasked tokens to a full sequence length, and a smaller decoder ViT then uses this sequence to reconstruct the image. The model is simply trained using mean squared error between the original, fully unmasked image and its reconstruction. The decoder is discarded for downstream tasks such as image classification, so just the trained encoder is used to featurize images into vectors.

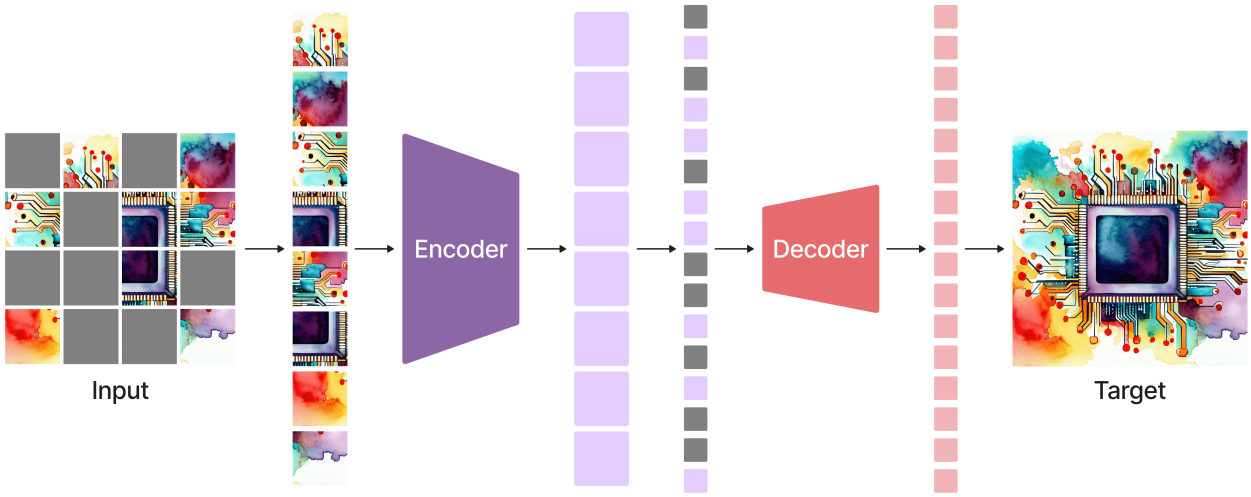


Figure 5: Overview of the MAE architecture. A ViT encoder is applied to unmasked patches of the input. The decoder processes the full set of masked and unmasked tokens to reconstruct the image at a pixel level. Schematic inspired by [35].

MAE is a more robust extension of denoising autoencoders with a few key differences. By using image

patches and ViTs, MAE can take advantage of the self-attention mechanism of transformers [37]. Because visible image patches can attend to other visible patches, the model learns much more about the structure of the image. Encoding only the unmasked patches at every forward pass allows MAE to be fast and scalable. This method of masked image modeling leads to representations that are much better aligned with downstream tasks than denoising autoencoders.

SimMIM SimMIM and MAE were introduced simultaneously as independent inventions for masked image modeling through masked autoencoding [38]. SimMIM is mostly the same as MAE but with a few minor differences in how sequences are encoded and decoded. Although the random masking strategy is the same as MAE, SimMIM encodes both masked and unmasked patches in the encoder. The decoder is a simple linear layer rather than a small ViT, and an L1 loss is used for the reconstruction loss instead of an L2 loss.

2.2 Joint Embedding

Masked image modeling is a relatively recent breakthrough in SSL. Joint embedding has been far more common and just as successful. Although joint embedding is non-generative, it can be compared to masked image modeling by framing the scope of the prediction task. In masked image modeling, a model predicts the *observed* variable, i.e. pixel values. In joint embedding, a model predicts a *hidden* representation of the observed variable, i.e. some latent representation of the image.

Figure 6 shows the general flow of a joint embedding architecture. Each image x in a dataset $\mathcal{D}$ is transformed using some set of random image augmentations $\mathcal{T}$. This produces n augmented views of the image, $x_1, \ldots, x_n$. These n images are then jointly embedded by an encoder network, which includes a "backbone" featurization network f_θ and a projection head h_ϕ. The backbone network f_θ can be any common deep learning architecture used for feature extraction, such as a ResNet [39] with the classification head removed. The projection head is usually a shallow fully-connected network or multilayer perceptron (MLP). f_θ maps each input image view x_i to produce *representations* or *features* $y_i = f_\theta(x_i)$, and these representations are nonlinearly transformed by the projection heads to obtain *embeddings* $z_i = h(y_i)$. Loss functions in SSL are typically applied to the embeddings or further transformations of the embeddings, with the goal of maximizing the agreement between augmented views of the same image. In effect, this leads to a bottom-up categorization based on data-driven association rather than the top-down categorization that is done when datasets are manually divided into categories.

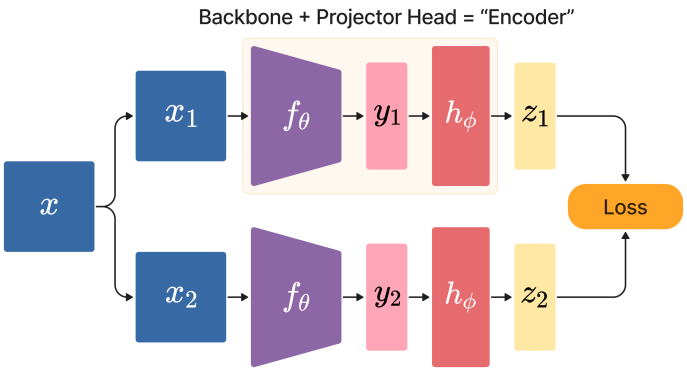


Figure 6: General overview of joint embedding. Most frameworks embed two augmented views of an image x as shown here. The union of the backbone and projector is often referred to as the encoder.

After self-supervised pretraining, the backbone network f_θ can be used for downstream tasks. For tasks requiring vector representations of images, such as image classification or similarity search, the backbone can directly be used to turn images x into vector representations y. For tasks such as object localization or instance segmentation where more fine-grained detail is needed, the backbone network can be used as part of a larger architecture. For example, a trained ResNet backbone could be used as part of a larger image segmentation framework, providing a much stronger initialization point than random weights and typically providing a better starting point than even ImageNet weights on out-of-domain datasets.

Just as the decoder networks are discarded after pretraining masked image modeling frameworks, the projector h_ϕ is also usually discarded. Empirically, the representation vectors y tend to be more semantically meaningful than the embedding vectors z [40], [41]. During training, however, it has been shown that applying self-supervised losses directly to the representations y degrades performance on downstream tasks. These loss functions typically encourage the network to ignore changes brought about by the set of image transformations $\mathcal{T}$, and this information is often useful for downstream tasks. Applying the loss on a projected view z essentially protects y by being "corrupted" by self-supervised losses.

A trivial solution to joint embedding is to output a constant vector for all images regardless of their information content. For example, consider a simple framework like the one illustrated in Figure 6 which outputs the same vectors for images of cats, dogs, and cars. This phenomenon is known as *representation collapse*. The multitude of joint embedding techniques that exist today differ in how they avoid representation collapse. This involves using different loss functions and network architecture heuristics. The following sections provide high-level overviews of some of the most popular techniques for joint embedding and the specific frameworks that have proven most successful.

2.2.1 Contrastive Learning

Contrastive learning is one of the oldest and most successful techniques in SSL, and it involves contrasting an anchor image against positive and negative samples. A dataset of images will have related and unrelated images, and the goal of contrastive learning is to assign similar embeddings for related images and dissimilar embeddings for unrelated images as illustrated in Figure 7.

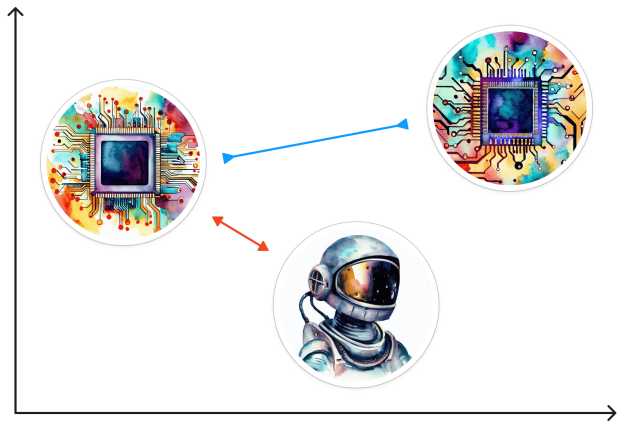


Figure 7: Schematic of contrastive learning. In the feature space, the embeddings of an image and its positive should be pulled together (blue), simultaneously pushing it away from the embedding of its negative (red).

Contrastive learning is not inherently self-supervised. In a supervised setting, an anchor image's positives

would include all images from the same class, while images from all other classes are negatives. Without labeled examples, positives are created by creating augmented views of an anchor image, and any other image besides the anchor is treated as a negative. The PIRL framework [42] was one of the first to do so by using pretext tasks such as jigsaw transformations and rotations from pretext learning. In PIRL, an anchor image is transformed to produce a heavily augmented view which is treated as its positive. A negative example is any other image in the dataset. Using a noise contrastive estimator (NCE) loss [43], the embeddings of an image and its augmentation are encouraged to be similar, while the embeddings of the image with other images in the dataset are encouraged to be dissimilar. This leads to strong visual representations that are invariant to the pretext transformations used, leading to features that are better aligned with downstream tasks than pretext learning [42].

SimCLR The SimCLR framework [40] is one of the most popular frameworks for contrastive learning, and it is perhaps the first example of modern SSL by showing results which were competitive with supervised learning. Instead of the NCE loss used in PIRL, SimCLR uses the improved normalized temperature-scaled cross-entropy loss, known as the NT-Xent or InfoNCE [44]. SimCLR introduced the concept of the projection head which is used in practically every modern SSL technique, showing that a learnable nonlinear transformation of image representations leads to better performance on downstream tasks. Additionally, it proposed a much more aggressive pipeline of random image augmentations to strengthen augmentation invariance. The SimCLR architecture is illustrated in Figure 8a.

The main drawbacks of SimCLR are its scalability requirements. To reach competitive results with supervised learning, SimCLR requires larger and wider backbone networks and projection heads [45]. More importantly, the contrastive loss requires many examples of strong negatives to converge more effectively and avoid representation collapse. The need for many negative samples leads to a large batch size requirement, making it difficult for SimCLR to scale down to single-GPU setups.

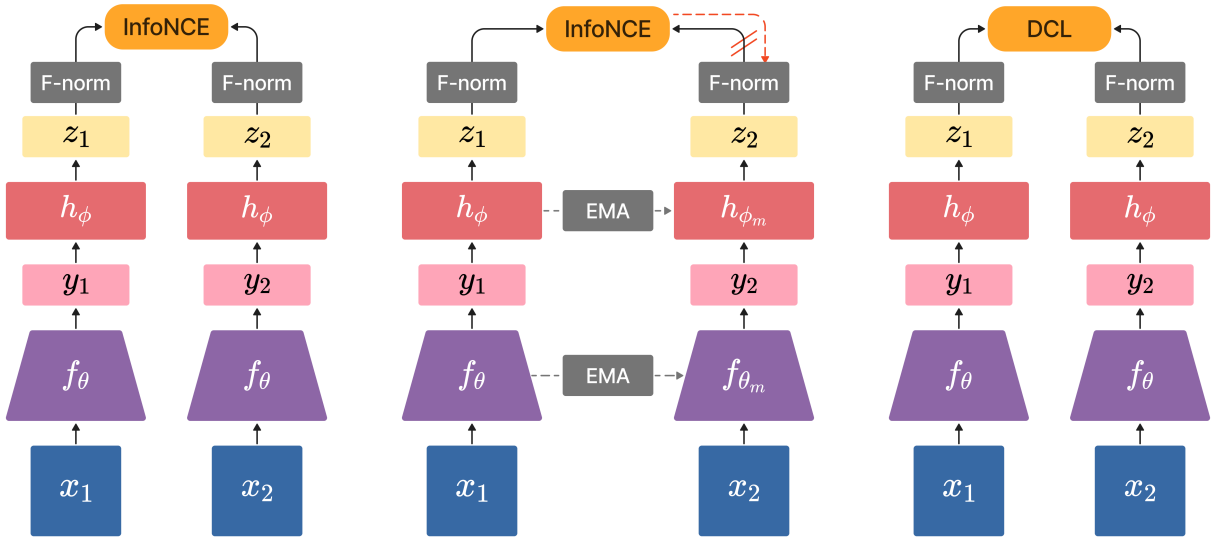
MoCo MoCo [46] introduces a momentum queue inspired by dictionary lookups in NLP to avoid large batch size requirements for contrastive learning. At a high level, masked language prediction techniques such as BERT use loss functions which involve a predicted word or *query* that is matched to the corresponding *key* in a dictionary of all possible words [25]. Extending this analogy to computer vision, the embedding of an image can be viewed as a query, and the dictionary would be a large set of image embeddings. Unlike in NLP, a dictionary of all possible image embeddings is not readily available in SSL, so it is created dynamically by creating a *momentum queue*. During training, the encoder on one branch of the joint embedding network is an exponential moving average of the encoder on the other side, creating a *momentum encoder*. Thus, the momentum queue is the set of image embeddings which are dynamically updated by the momentum encoder. The dictionary lookup problem is simply the contrastive loss using the query embedding and the corresponding key in the momentum queue. Figure 8b illustrates the MoCo architecture.

The use of the momentum queue decouples the contrastive loss from the batch size, allowing MoCo to match SimCLR's performance with lower hardware requirements [46]. Additionally, revisions to MoCo improving its training procedure allow it to surpass supervised learning in performance after the pretrained encoder is fine-tuned. Since MoCo, it has been very common to see purely supervised learning techniques surpassed by self-supervised pretraining followed by supervised fine-tuning, suggesting that the gap between supervised and unsupervised representation learning has largely been closed [47], [48].

DCL DCL [49] can be seen as an extension of SimCLR that also decouples the contrastive loss from batch size requirements. Unlike MoCo, however, this is done by changing the loss function, removing the requirement of momentum encoder. The InfoNCE loss used in SimCLR and MoCo is formulated such that positives and negatives have a strong coupling interaction that reportedly reduces learning efficiency. Contrastive learning using the InfoNCE loss is inefficient in three main scenarios:

1. A positive sample is already very close to the anchor in the feature space.
2. A negative sample is already very far from the anchor in the feature space.
3. Very few negative samples are available due to a small batch size.

The decoupled contrastive learning loss (DCL) reformulates the InfoNCE loss and removes the negative-positive coupling, allowing DCL to improve the efficiency of contrastive learning and be less sensitive to hyperparameters compared to SimCLR [49]. A momentum queue is not needed since this loss already decouples contrastive learning from the batch size. Figure 8c illustrates the DCL architecture.



(a) SimCLR (b) MoCo (c) DCL

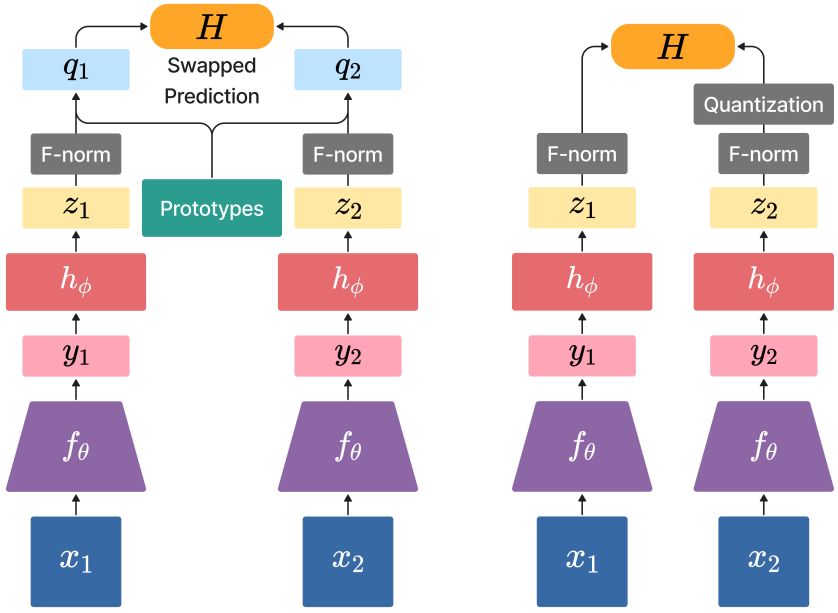
Figure 8: Conceptual comparison between different contrastive learning frameworks. Notation is the same as introduced earlier; x represents images, f_θ is the encoder's backbone which produces representations y, and h_ϕ is the encoder's projector which produces embeddings z. Non-parametric operations are shown in gray, with "F-norm" denoting feature normalization, and "EMA" denoting exponential moving average (momentum, hence the subscript $_m$). Stop gradient operations are denoted with dotted red lines and red slashes to indicate interrupted gradient flow to the momentum branch in MoCo.

2.2.2 Clustering

Contrastive learning involves pulling and pushing samples in feature space to make similar samples close together and dissimilar samples far apart. This essentially creates groups in the feature space, so SSL can be framed as a clustering problem. Most successful in this regard is SwAV [50], which performs online clustering of augmented image views to partition a feature space according to similarity.

SwAV The general idea behind SwAV is to assign the same cluster to different augmentations of an image. A trivial solution is to simply assign all images and their transformations to the same cluster, so representation collapse is prevented by controlling the clustering process. Specifically, a Sinkhorn-Knopp clustering scheme is used to enforce an equipartition constraint while also ensuring that each of the N samples in a dataset are similar to at most N/K samples, where K is a hyperparameter corresponding to the number of cluster prototypes. These prototypes are learnable vectors with the same shape as the embeddings that are initialized randomly. SwAV uses soft clustering instead of hard clustering, allowing samples to partially belong to multiple clusters. For every sample, a code q can be created which specifies how similar a sample is to every cluster prototype.

Next, a swapped prediction task is performed. Considering two image views used for joint embedding, this swapped prediction task would involve predicting the code of one view using the embedding of the other, i.e. z_1 to predict code q_2, and z_2 to predict q_1. This swapped view prediction ensures invariance to augmentation. The embeddings should capture similar information since they come from the same view, so it should be possible to predict the code from the swapped embedding. Figure 9a shows an overview of the SwAV architecture. At a high level, SwAV can also be viewed as a knowledge distillation method in which a teacher network produces quantized vectors q used as the target by a student network producing embeddings z. This is illustrated in Figure 9b.



(a) Full View (b) Distilled View

Figure 9: Overview of the SwAV architecture. Both the full view and simplified distilled view are shown for conceptual comparison. Here, H denotes the cross-entropy loss between codes and embeddings in the swapped prediction task.

SwAV also introduces a *multi-crop* augmentation policy which substantially improves its performance by allowing relations to be learned between smaller parts of objects and scenes [50]. In multi-crop, several small crops of an input image are used in addition to two larger crops as shown in Figure 10.

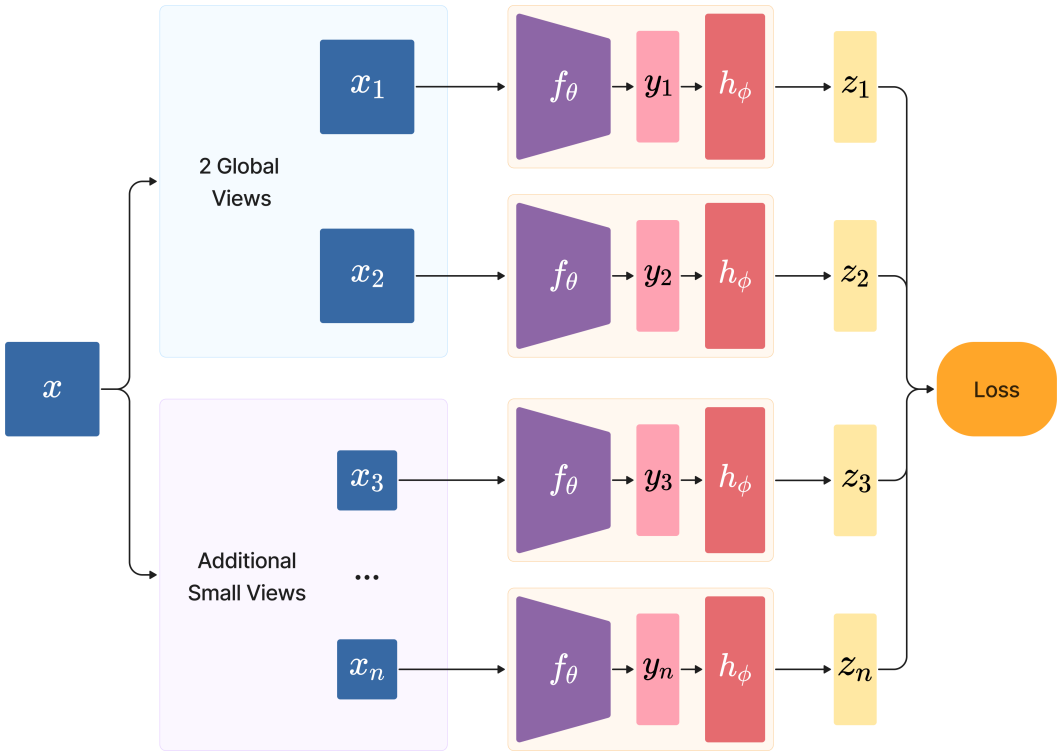


Figure 10: Diagram of the multi-crop technique used in SwAV. Schematic inspired by [50].

2.2.3 Self-Distillation

Several recent SSL frameworks do not explicitly avoid representation collapse through contrastive losses or clustering. Rather, they rely on architecture heuristics inspired by knowledge distillation [51], using asymmetries in different parts of the network to prevent trivial solutions.

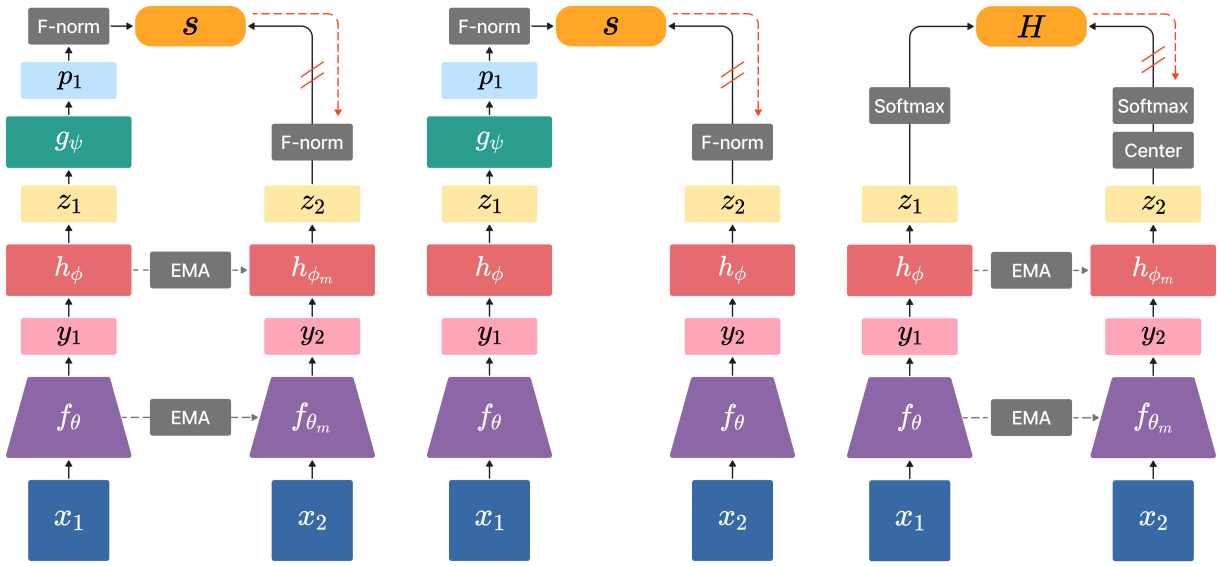
BYOL BYOL [52] uses three forms of asymmetry to avoid representation collapse. First, it creates asymmetry in the architecture by introducing an additional MLP to the "student" branch of the network called a prediction head g_ψ, which uses the embeddings to produce a prediction vector $p = g_\psi(z)$. Second, asymmetry in the learning rule is introduced by flowing gradients only through the student branch by using a stop-gradient operation on the "teacher branch." Third, asymmetry is introduced in the encoders by making the teacher encoder an exponential moving average of the weights of the student encoder, similar to MoCo. The loss function is simply a negative cosine similarity between the prediction p of the student branch and the embedding z of the teacher branch. These asymmetries help prevent representation collapse by ensuring that the embedding and prediction are different. Figure 11a shows the architecture of BYOL.

SimSiam SimSiam [41] is a very slight modification of BYOL created to test the limits of distillation-inspired methods in avoiding representation collapse. It can essentially be viewed as the result of thorough ablation studies on the BYOL architecture, seeing whether reductions in batch size and removal of certain asymmetries lead to trivial solutions. Architecturally, the only difference between BYOL and SimSiam is that the latter uses no exponential moving average of weights to update the two branches, suggesting that a momentum encoder is unnecessary to prevent collapse. It also uses a much smaller batch size than BYOL

while maintaining similar performance on downstream tasks. Figure 11b shows the architecture of SimSiam.

DINO Unlike most previous joint embedding frameworks, DINO [53] is designed mostly around the use of ViTs in the encoders. Like SimSiam, DINO can be seen as a modification of BYOL, but it introduces several key differences. The predictor head is removed, keeping the momentum encoder and stop-gradient from BYOL. Instead of negative cosine similarity as in BYOL and SimSiam, DINO uses a softmax followed by a cross-entropy loss. Asymmetry is introduced in the softmax of the teacher branch by performing a centering and sharpening operation. This is responsible for preventing representation collapse in DINO. Centering prevents a single dimension of the embedding from dominating over the rest, and the sharpening operation encourages the distribution to have high peaks. Figure 11c shows the architecture of DINO.

This softmax followed by cross-entropy is very similar to how supervised classification tasks are framed, only there are no classes involved in SSL. In DINO, the joint embedding network essentially creates its own pseudo-classification task using the embeddings of augmented images. DINO also uses the multi-crop augmentation policy from SwAV, using only the large image crops in the teacher branch and using both the large and small image crops in the student branch. Thus, this pseudo-classification task can be interpreted as using the embeddings of the small and large crops in the student network (global and local image information) to predict high-level descriptions of the same image from the embeddings of the large image crops encoded by the teacher branch. DINO's combined use of multi-crop, self-distillation, and self-attention allows it to generalize exceptionally well to different tasks and datasets [53].



(a) BYOL (b) SimSiam (c) DINO

Figure 11: Conceptual comparison between different distillation-inspired frameworks. The student and teacher branches refer to the left and right branches, respectively, which is somewhat different from the definition in knowledge distillation [51]. Here, s represents the negative cosine similarity loss. g_ψ and p refer to the predictor head and prediction vectors, respectively. As before, H denotes cross-entropy, and the dotted red lines and red slashes indicate stop-gradients which interrupt gradient flow to the teacher branch.

2.2.4 Redundancy Reduction

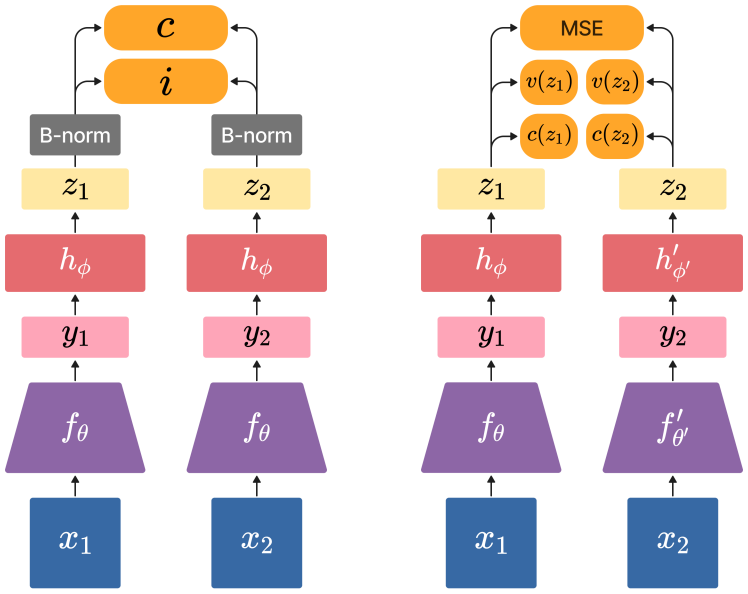
Contrastive learning, clustering, and distillation-inspired joint embedding are all similarity maximization techniques because they are explicitly formulated to maximize agreement between image embeddings. In contrast, joint embedding based on redundancy reduction is not formulated to explicitly do so. Rather, similarity maximization is implicitly ensured by maximizing the information content of the embeddings, or reducing redundant image features. This takes inspiration from concepts in neuroscience, namely Barlow's redundancy-reduction principle which hypothesizes that neurons in the brain have a limited capacity for spiking, leading to maximally efficient "spiking codes" [54].

Barlow Twins At a high level, the Barlow Twins framework [55] ensures feature invariance and independence; image features should be invariant under different data augmentations, and each dimension of a feature vector should be independent of the others to reduce redundancy. In practice, a cross-correlation matrix is computed between embeddings. The goal is to ensure this matrix is as close as possible to the identity matrix. Having values close to one only along the diagonal ensures that each dimension of the embeddings remains similar under different data augmentations, while having zeros on the off-diagonal entries ensures that separate dimensions of the embeddings encode different information to avoid redundancy. Representation collapse is prevented by performing a centering operation based on batch statistics before the cross-correlation matrix is computed.

Despite having a much simpler formulation, Barlow Twins is able to match the performance of several popular techniques based on contrastive learning, clustering, and distillation, suggesting that SSL does not require asymmetric network architectures or loss functions which specifically maximize the similarity between samples [55]. The Barlow Twins architecture is illustrated in Figure 12a.

VICReg VICReg [56] builds upon the redundancy reduction principle introduced by Barlow Twins to remove many of the popular architectural constraints of most joint embedding frameworks. It does not require that the encoders of both branches of the network be shared, nor does it require a momentum encoder on one branch that is a moving average of the other's weights. It also does not require feature normalization across samples or batches, or information quantization as in clustering techniques. There is also no requirement for architectural asymmetry.

This is accomplished by using three terms in the loss function which prevent representation collapse and reduce information redundancy. First, an invariance term is used that computes the mean squared error between different embedding vectors, ensuring that information content remains invariant to data augmentation. Second, a variance term is introduced to prevent representation collapse within a batch. This is done using a hinge loss that maintains the standard deviation of a batch of embedding vectors above a certain threshold. Third, a covariance term minimizes the covariance of every pair of feature dimensions in an embedding vector. This is similar to the feature decorrelation performed in Barlow Twins. Only the invariance term in the loss operates on both branches of the joint embedding network together; the variance and covariance regularization is performed independently on each branch. Figure 12b illustrates the VICReg architecture.



(a) Barlow Twins (b) VICReg

Figure 12: Conceptual comparison of redundancy reduction frameworks. In Barlow Twins, c and i represent the covariance and invariance terms in the loss, respectively. "B-norm" refers to the centering operation by batch normalization before the covariance matrix is computed. In VICReg, v is the variance term or hinge loss, and c is the covariance term. Both are applied independently on the embeddings of each branch, with only mean squared error (MSE) being computed between branches as the invariance term.

2.2.5 Mask Denoising

Very recent developments in joint embedding take inspiration from masked image modeling. Masked autoencoders learn representations by making predictions in the pixel space to reconstruct images. This involves learning low-level image details that may not be necessary for broad categorization tasks involving semantic abstraction. Joint embedding techniques involving mask denoising perform predictions in the embedding space instead. Specifically, the similarity is maximized between embeddings of masked and unmasked images.

MSN The Masked Siamese Networks (MSN) framework [34] can be seen as a hybrid of several frameworks across different SSL techniques. Like masked autoencoders, MSN uses ViTs for representation learning of partially masked images. Unlike MAE or SimMIM, however, the images are still put through randomized augmentation pipelines as is common in all joint embedding techniques. The augmented views are first divided into patches for use in the ViT. Following distillation-inspired techniques like BYOL and DINO, the network includes a student encoder and a teacher encoder created using an exponential moving average of the student encoder's weights. Masked images are embedded by the student branch, while unmasked images are embedded by the teacher branch. Multiple masking strategies are used to provide global and local information in the image, similar to the large and small image crops in the multi-crop policy used by SwAV and DINO. Specifically, random masking drops patches over the entire image to create a global view, while focal masking keeps a localized contiguous block of patches unmasked, dropping everything else. Figure 13 shows this masking strategy.

(a) No Mask (b) Random Mask (c) Focal Mask

Figure 13: MSN masking strategy. All images are "patchified" as shown in (a), with unmasked images encoded by the teacher branch. Masked images are encoded by the student branch. They may be randomly masked as in (b), or focally masked as in (c). Schematic inspired by [34].

Like SwAV, the prediction task involves a clustering step in which image embeddings are soft-assigned to cluster prototypes. The goal is to assign the embedding of the masked view to the same cluster prototype as the embedding of the unmasked view, optimized by a cross-entropy loss. This prediction step is slightly different than in SwAV, since no swapped view prediction is performed. Rather, prediction vectors p are computed using the softmax of the cosine similarities between embeddings and cluster prototypes (similar to the codes q in SwAV). Then, the prediction p_2 of the target branch is sharpened, and the cross-entropy loss is computed between the predictions of the two branches, similar to DINO. The MSN architecture is illustrated in Figure 14, which can be viewed as a hybrid of SwAV and DINO.

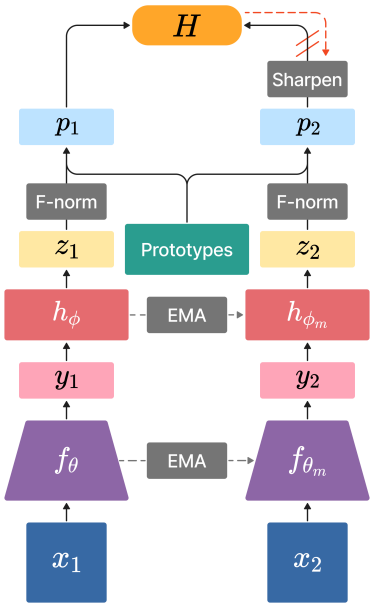


Figure 14: Conceptual diagram of MSN. For the sake of simplicity, cluster codes q are omitted since $p = \text{softmax}\left(\frac{z \cdot q}{\tau}\right)$. $\tau \in (0, 1)$ is a temperature parameter, and sharpening is performed by keeping $\tau_2 < \tau_1$.

PMSN PMSN [19] is an extension of MSN designed for use on class-imbalanced data. Although it has been shown that SSL is more robust to dataset imbalance than supervised learning [16], [17], most joint embedding

techniques assume a uniformly distributed feature prior. It can be shown that several frameworks such as SimCLR, VICReg, SwAV, and MSN employ a form of implicit K-Means clustering in the feature space which tends to uniformly cluster the data [19]. The assumption of having equally sized clusters in the feature space is fairly reasonable when there is no knowledge of underlying class distributions. On the other hand, sometimes there is some knowledge about the shape of the distribution even if the underlying categories' exact sizes are unknown, such as whether the distribution is long-tailed. In semiconductor manufacturing, for example, the exact number of nearly zero-yield wafers may not be known, but it would be reasonable to assume that a high-volume manufacturing scenario would have fewer wafers like these than high-yielding wafers.

PMSN encourages feature distributions to follow a power-law distribution to handle long-tailed distributions. It is architecturally the same as MSN, modifying only the loss function. In MSN, the cross-entropy loss is regularized with a mean entropy maximization term which encourages the model to use the full set of cluster prototypes, implicitly encouraging uniform clustering [34]. In PMSN, this term is simply replaced with the KL-divergence to a power-law distribution, encouraging the model to use cluster prototypes unequally. Interestingly, this change leads to improved performance on class-imbalanced data but significantly worse performance on class-balanced data, suggesting that it should only be used when data is strongly suspected to be class-imbalanced [19]. Comparisons of other SSL techniques on class-imbalanced data show that masked image modeling frameworks like MAE are completely unaffected by dataset imbalance, since generative SSL does not involve computing batch statistics of features [19].

2.3 Current State of Self-Supervised Learning of Wafer Maps

Only the earliest forms of contrastive learning have been applied to wafer map defect analysis. Specifically, Kahng et al. [22] developed a PIRL-like framework for self-supervised representation learning of wafer maps, while Hu et al. [23] developed a SimCLR-like framework and domain-specific image augmentation techniques. In other words, only the earliest forms of contrastive learning have been applied to wafer map data. Since then, there have been significant advances in SSL, leaving several unanswered questions about applying SSL to this domain. Do non-contrastive methods of joint embedding outperform contrastive learning frameworks? Is masked image modeling better than joint embedding for wafer map data since it is highly class-imbalanced? Do CNNs work better than ViTs on wafer maps? What data augmentations are necessary to provide a learning signal for joint embedding?

Additionally, the generalization performance of self-supervised features has not been thoroughly investigated for wafer map data. The applications of PIRL and SimCLR on wafer map data by [22] and [23] successfully showed that self-supervised pretraining followed by supervised fine-tuning leads to strong classification performance on the WM-811K dataset. What has not been assessed is whether the self-supervised features are semantically meaningful themselves *before* fine-tuning. Typically there are three means of evaluating self-supervised pretraining on classification tasks:

1. Use labeled examples to fine-tune the encoder, either partially or fully, and fit a linear classifier
2. Use labeled examples to fit just a linear classifier, leaving the encoder frozen
3. Use no labels, directly using frozen features for classification

The first and second methods involve the use of some supervision. Fine-tuning the encoder essentially measures how much supervision is needed to get the *encoder* to align well with classification tasks. Fitting

just a linear classifier measures how much supervision is needed for frozen *features* to be well-aligned with classification. The third method is a much better means of measuring feature generalizability. For a feature to truly be generalizable, it must already be well-aligned to broad categorization tasks like classification without using any supervision after self-supervised pretraining. k-NN evaluation can be used to this end. Specifically, the feature y_i of an image x_i can be compared to its k nearest features $y_1, \ldots, y_k$, using a simple majority vote to predict the label using the nearest neighbors' labels [57].

Finally, classification metrics alone are not enough to judge the quality of different SSL frameworks. It is possible for models with the same classification performance to make their predictions in entirely different ways. Qualitative interpretability studies should be performed to compare different SSL techniques. This can be done by visualizing the self-supervised representation space and analyzing whether points close together in the representation space are visually similar.

3 Approach

3.1 Dataset Cleaning

The WM-811K dataset has around 811,000 wafer maps, but only about 173,000 of these are labeled. The original authors have somewhat arbitrarily assigned 54,355 as "training" and 118,595 as "test" data. As shown in Figure 15, the label distributions across these slits are completely different. To ensure independent and identically distributed slits, a new aggregate dataset is created from the original splits in order to create stratified splits with identical label distributions. Specifically, the original train dataset is aggregated with all of the non-"None" type samples from the original test data. This new aggregate dataset comprises 62,248 wafer maps, and it is then split into training, validation, and test datasets in a 60/20/20 ratio. Figure 15 and Table 1 shows the class distributions of the new splits compared to the original "train" and "test" splits of WM-811K.

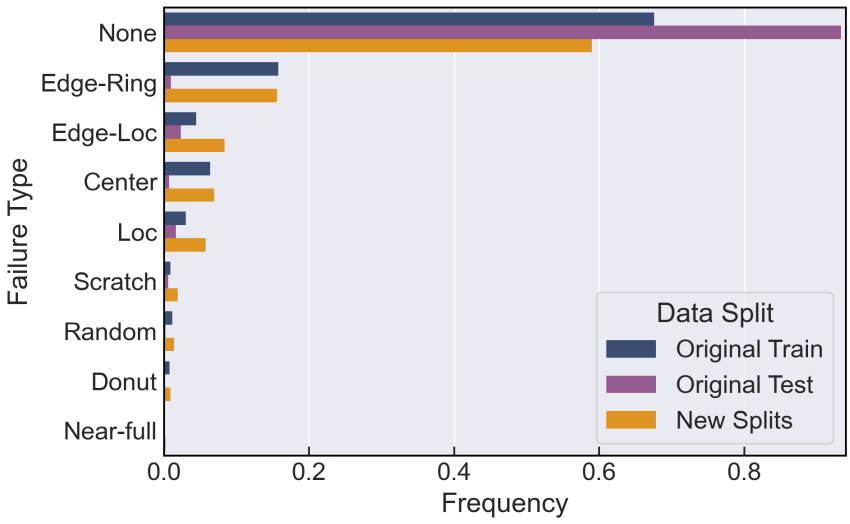


Figure 15: Relative class frequencies across the original training/test splits and the new splits.

Table 1: Relative class frequencies across the original training/test splits and the new splits.

Label	Original Train	Original Test	New Splits
None	0.676	0.933	0.590
Edge-Ring	0.157	0.00949	0.156
Center	0.0637	0.00702	0.069
Edge-Loc	0.0445	0.0234	0.0834
Loc	0.0298	0.0166	0.0577
Random	0.0112	0.00217	0.0139
Scratch	0.0092	0.00584	0.0192
Donut	0.00752	0.00123	0.00892
Near-full	0.000993	0.000801	0.00239

3.2 Data Augmentations for Joint Embedding

Strong augmentation pipelines are the key ingredient to effective pretraining through joint embedding. In general, the augmentations used should match the set of invariances that need to be learned for downstream tasks. Not every augmentation technique will be relevant for a given dataset, so the augmentations relevant to wafer map data are discussed here.

Spatial Transforms For wafer map data, simple spatial transforms are relevant since defect representations should be invariant to changes in the orientation of the defect pattern. In other words, the general shape of the defect is more important than its exact position. Manufacturing equipment will often move and rotate wafers during fabrication processes, so the same root cause will almost never result in exactly the same position of defects.

Pixel-level Transforms Binary wafer maps only have three possible pixel values representing failing die, passing die, and non-wafer area. Therefore, augmentations that introduce aggressive pixel-level changes such as color jitter, solarization, and Gaussian blur cannot be applied to wafer maps. These are fairly obvious pointers, but a far less obvious and often overlooked transformation is image interpolation. Although this is more of an implicit transformation than a concrete data augmentation technique, it introduces pixel-level changes when images are resized or rotated. Typically, the default interpolation mode in most computer vision libraries is bilinear interpolation, which does not clamp the output domain to the input domain. As illustrated in Figure 16, only nearest-neighbor interpolation is valid for binary wafer maps, since it ensures that the transformed image will only contain pixel values that were present in the original image.

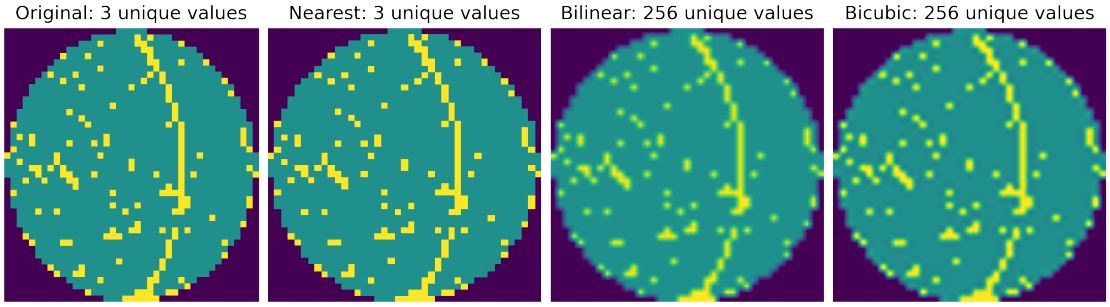


Figure 16: Comparison of three different interpolation methods when resizing a non-square wafer map.

Cropping Randomized cropping is a popular and highly successful augmentation technique for joint embedding architectures, but it is unclear whether it fully applies to wafer map data. Cropping makes sense for natural images since camera orientations aren't always fixed, so the subject of the image is not always in the center of the image. On the other hand, wafer maps will always have the same image composition; wafers will never be shifted off-center because wafer maps are not photographs. This has important implications for joint embedding. The similarity between crops will be maximized in these frameworks, but this only makes sense if the crops are semantically similar to begin with. For example, a defect that covers a small portion of the wafer may be missed entirely by randomized cropping, as illustrated in Figure 17.

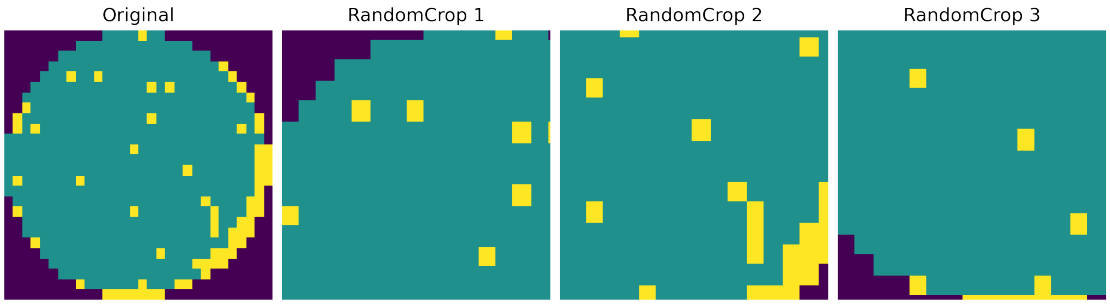
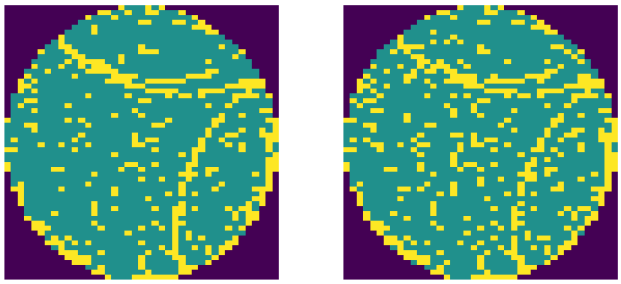


Figure 17: Randomized crops of a wafer map with an edge-localized defect.

On the other hand, a wafer with several defects may benefit from having its cropped features be maximally similar, since joint embedding would learn invariances between correlated defects. Because it is difficult to determine whether cropping is valid for wafer maps, it is not used by default here. However, it is still used in the joint embedding techniques designed specifically around multi-crop augmentations or variants of it, namely SwAV, DINO, MSN, and PMSN. This is done to test whether multi-crop helps or hurts representation learning of wafer maps.

Domain-Specific Data Augmentations This work adopts the randomized die noise augmentation from Hu et al. [23] illustrated in Figure 18, in which passing and failing die have a fixed probability of being flipped to the opposite type. Here, each die has a 3% chance of being flipped to the opposite type. Non-wafer area is not affected. This transformation enables the network to learn invariances to minor changes in defect patterns.



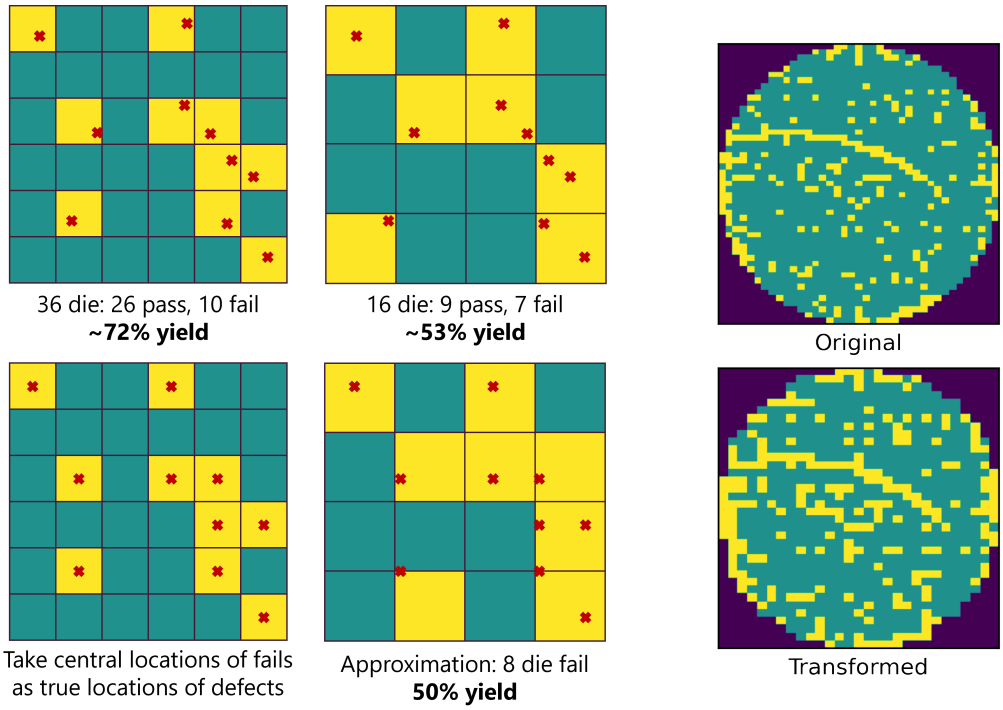
(a) Before (b) After

Figure 18: The die noise transformation introduces subtle changes to defect patterns.

This work introduces a novel augmentation technique specific to binary wafer maps: a die-per-wafer (DPW)

transformation. In semiconductor manufacturing, a process line will always see wafers of the same overall size (typically 300 mm wafers), but the number of die on the wafer may be different. Die size varies across products, and a fixed defect pattern will cause greater yield loss on wafers with larger die (or lower DPW). Consider a wafer with 100 die and a wafer with 10 die. If a small particle defect only causes 1 defective die on each wafer, the first wafer has 99% yield, while the second has 90% yield.

For wafer map data, the true locations of defect sources such as particulates are not known *a priori*. Thus, to approximate yield loss for a lower DPW product given any wafer map, the central coordinates of the failing die can be assumed to be the "true" locations of defect sources. In a DPW transformation, these coordinates are then mapped to a wafer map with fewer die. Figure 19 illustrates this augmentation. Note that the DPW transformation is an *estimate* of how a defect pattern would look on a lower DPW variant of a wafer map, so it inherently introduces some noise. In the augmentation pipeline for this work, either randomized die noise or the DPW transformation is applied. We choose not to apply both simultaneously to prevent excessive changes in the defect patterns.



(a) Schematic of DPW transform

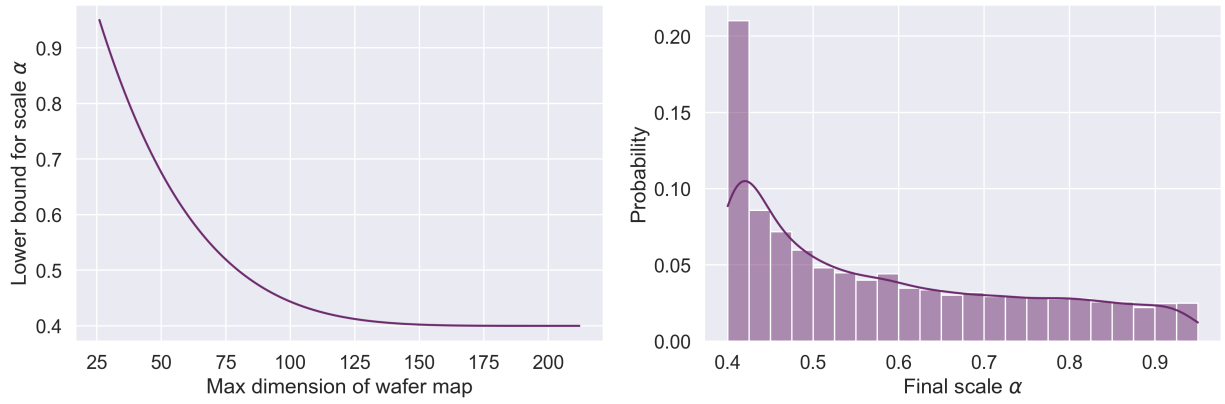
(b) Before and After

Figure 19: (a) describes the DPW transformation. The top row illustrates a fixed defect source affecting wafers with different die size, and the bottom row illustrates how the transformation approximates a lower DPW wafer map. (b) shows the effect of the DPW transformation. The result is a lower DPW wafer map with higher yield fallout.

In practice, a lower DPW wafer map is initialized by using a scale parameter α which multiplies the dimensions of the original wafer map. For example, a scale parameter $\alpha = 0.75$ given a wafer map with dimensions 40×50 would yield a wafer map with dimensions 30×37. A new wafer map can be created by initializing a zeros-matrix with these new dimensions. The coordinates of the passing and failing die in this new wafer can be calculated by first finding the relative *central* coordinates of the passing and failing die in the original wafer. These coordinates are then mapped to the domain of the lower DPW wafer map, and the

matrix indices corresponding to these coordinates are filled in accordingly. Listing 1 in Appendix A provides a code snippet for these matrix operations.

Randomness is a critical component of data augmentations used in joint embedding. Randomness is introduced to the DPW transformation by creating a bound from which to choose the scale parameter α. Die sizes in the WM-811K dataset follow a long-tailed distribution, with the majority of wafer maps being low DPW. Thus, the high DPW wafer maps should use a lower scale parameter α than the already low DPW wafers. As shown in Figure 20a, a power law distribution is used to map the maximum dimension of the wafer map, i.e. `max(width, height)`, to some lower bound. The upper bound for α is set to 0.95 for minor changes to very low DPW wafer maps, and the absolute lower bound for α is set to 0.4. As the maximum dimensions of the wafer map increases (that is, as the DPW of the original wafer increases), the lower bound of α quickly decays to the absolute lower bound of 0.4. Then, the final scale parameter α is randomly chosen from a skewed beta distribution, lower bounded by this previous value, and upper-bouded at 0.95. Figure 20b shows how the final scale parameter would be chosen if the lower bound was 0.4.



(a) (b)

Figure 20: (a) shows the power-law distribution used to map the max dimension of a wafer map to the lower bound for the DPW transform's scale parameter α. (b) shows the beta distribution which would be used to randomly choose α if the power-law mapping had led to a lower bound of 0.4.

Overall Augmentation Pipeline Outlined below is the augmentation pipeline. Figure 21 provides example views that would be returned for a given image. Most joint embedding frameworks use two augmented views (hence the term "Siamese networks"), but others use more than two.

- Perform one of the following, with equal probability of choosing either:
  - Randomized die noise
  - DPW transformation
- 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

Images are resized to $3 \times 224 \times 224$ for compatibility with most popular backbone architectures. Although

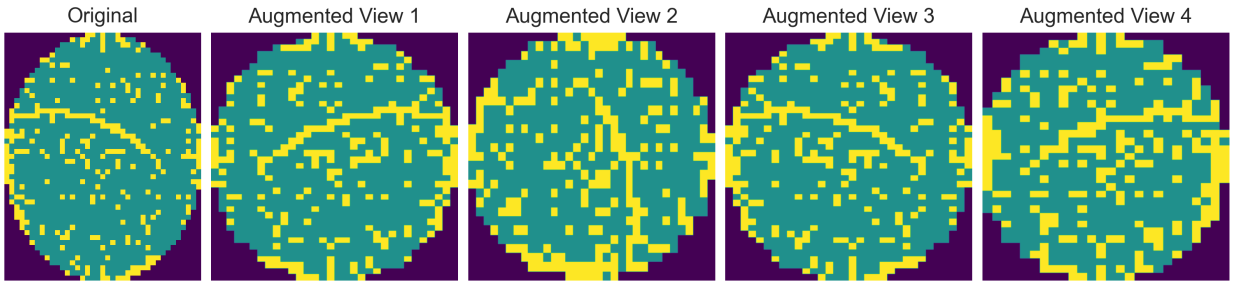


Figure 21: Examples of augmented views returned by the augmentation pipeline.

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, MSN, and PMSN. 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.

3.3 Experimental Setup

The training dataset is used for self-supervised pretraining, and the validation dataset is used for a weighted k-NN evaluation following the evaluation procedure outlined in [57]. The testing dataset is not used for k-NN evaluation because the evaluation proceeds in an online fashion. That is, k-NN classification performance is measured at *every* epoch. Each training epoch proceeds in a self-supervised fashion (no labels are used). At the end of each training epoch, all of the training samples are passed through the model's backbone network in inference mode, creating a feature bank. The ground truth labels for the training data are attached to their respective features. Then, the validation epoch proceeds by passing validation samples through the model's backbone. For each validation feature, the k-NN classifier uses the k nearest features from the training feature bank to vote for the label of the validation sample as illustrated in Figure 22.

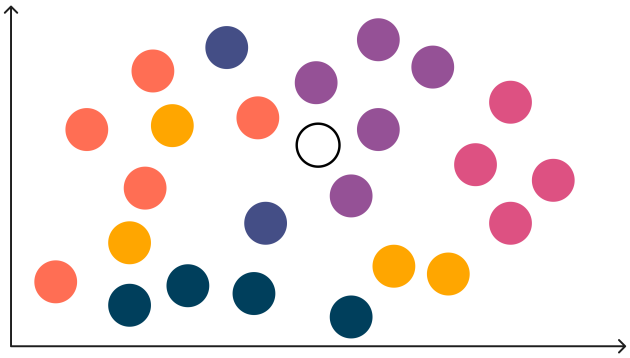


Figure 22: Schematic of k-NN voting in the feature space. The unseen validation sample (white) is compared with its k nearest neighbors in the feature space. With $k = 5$, this would be predicted as a purple sample.

For this dataset, a sweep of different k values shows that $k = 5$ nearest neighbors works best for evaluation. This may seem like a very small k. For reference, the original implementation of DINO uses $k = 20$ on ImageNet evaluation [53]. However, the WM-811K has several minority classes which are difficult to predict via k-NN classification with high values of k. Multiclass accuracy and F1 scores are used as evaluation metrics here. Both are macro-averaged due to class imbalance. When computing multiclass classification metrics, micro-averaging gives equal weighting to every sample, while macro-averaging gives equal weighting to every class.

The SSL frameworks employed include a variety of contrastive and non-contrastive joint embedding methods and two masked image modeling techniques as shown below.

- **Joint Embedding**
  - **Contrastive Learning**: SimCLR, MoCo, DCLW
  - **Clustering**: SwAV
  - **Distillation**: BYOL, SimSiam, FastSiam, DINO
  - **Redundancy Reduction**: Barlow Twins, VICReg
- **Masked Image Modeling**: MAE, SimMIM

With the exception of DCLW and FastSiam, all of these frameworks have been explained in section 2. DCLW is a slight modification to DCL that introduces a weighting function in the DCL loss. This gives a larger weight to "hard positives," which are positive samples that are far from the anchor. Empirically, this weighting improves performance and increases representation quality [49]. FastSiam is a modification of SimSiam that claims to offer faster convergence by using more than two image views. Specifically, four augmented views are used for joint embedding to stabilize training [58]. As shown in Figure 23, the loss is computed using the prediction p_1 of the first view and the element-wise mean of the embeddings of the remaining views z_2, z_3, z_4.

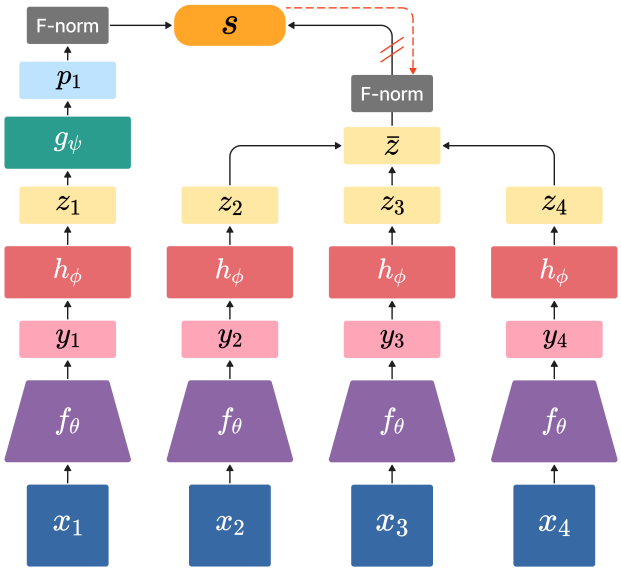


Figure 23: Conceptual diagram of FastSiam.

ResNet-18 [39] is used as the encoder's backbone network for all of the joint embedding models. Although

more modernized CNNs such as RegNet [59] and ConvNeXt [60], [61] have recently been used for SSL, in practice, ResNets are still the standard model for this use case. Additionally, they can be quite effective with today's best practices for neural network training [62]. ViT-S/16 [36] is used as the backbone for MSN and PMSN and tested as an additional backbone for DINO. Both MAE and SimMIM use the much larger ViT-B/32 [36] as the backbone encoder. MSN and PMSN use masking ratios of 0.15 as they use smaller networks, while MAE and SimMIM use a more aggressive masking ratio of 0.75 since larger networks generally scale better with higher masking ratios [34], [35]. Further implementation details, such as model-specific optimizers and projector head dimensions, can be found in Appendix B.

Hardware Setup All models were implemented in PyTorch Lightning [63] using the `lightly` Python package for SSL [64]. Although SSL is often most effective when using large batch sizes with huge models trained on multiple GPUs, it is unreasonable to assume that most of the research community has access to these resources. This work investigates whether SSL can scale down to realistic, single-GPU setups on uncurated datasets like WM-811K. To this end, training employs consumer-grade hardware with fairly modest amounts of VRAM. Specifically, all models are trained for 150 epochs using 16-bit Automatic Mixed Precision with a batch size of 64 on a system with an RTX 3080 Ti (12GB VRAM), Ryzen 5 5600X, and 32GB of system memory. Although mixed precision training may lead to slightly reduced model accuracy and possible numerical instabilities, it greatly reduces memory requirements and training time.

4 Results and Discussion

4.1 k-NN Evaluation Results

Table 2: Reported metrics are the maximum macro-averaged accuracy and F1 score achieved by the k-NN classifier during 150 epochs of training. Models denoted † did not learn effectively, so the reported metrics for these are from the end of training, not the maximum metrics. "Views" refers to the number of augmented views used per training sample, which includes the image crops used in SwAV, DINO, and MSN/PMSN.

Technique	Framework	Views	Backbone	# params ($\times 10^6$)	Max Accuracy	Max F1	Time (hrs)	Memory (GB)
Contrastive Learning	SimCLR	2	ResNet-18	11.5	70.1	71.0	3.53	1.75
	MoCo	2	ResNet-18	12.5	73.6	75.0	4.42	2.08
	DCLW	2	ResNet-18	11.5	73.2	75.0	3.60	1.75
Clustering	SwAV	8	ResNet-18	12.9	74.2	75.8	11.96	2.97
Self-Distillation	BYOL	2	ResNet-18	16.4	76.7	78.2	3.62	2.11
	SimSiam†	2	ResNet-18	22.7	31.2	31.2	3.50	1.90
	FastSiam†	4	ResNet-18	22.7	33.5	34.5	5.99	3.40
	DINO	8	ResNet-18	17.5	70.0	71.5	8.48	3.07
	DINO	8	ViT-S/16	27.7	69.0	70.7	19.87	10.25
Redundancy Reduction	BarlowTwins	2	ResNet-18	20.6	72.3	74.8	8.43	2.04
	VICReg	2	ResNet-18	20.6	72.5	73.6	4.15	1.92
Mask Denoising	MSN	12	ViT-S/16	27.8	71.6	72.7	11.26	7.29
	PMSN	12	ViT-S/16	27.8	71.1	72.2	11.05	7.29
Generative	MAE	1	ViT-B/32	93.4	69.7	69.4	3.47	2.3
	SimMIM†	1	ViT-B/32	90.6	18.5	15.9	4.09	2.95

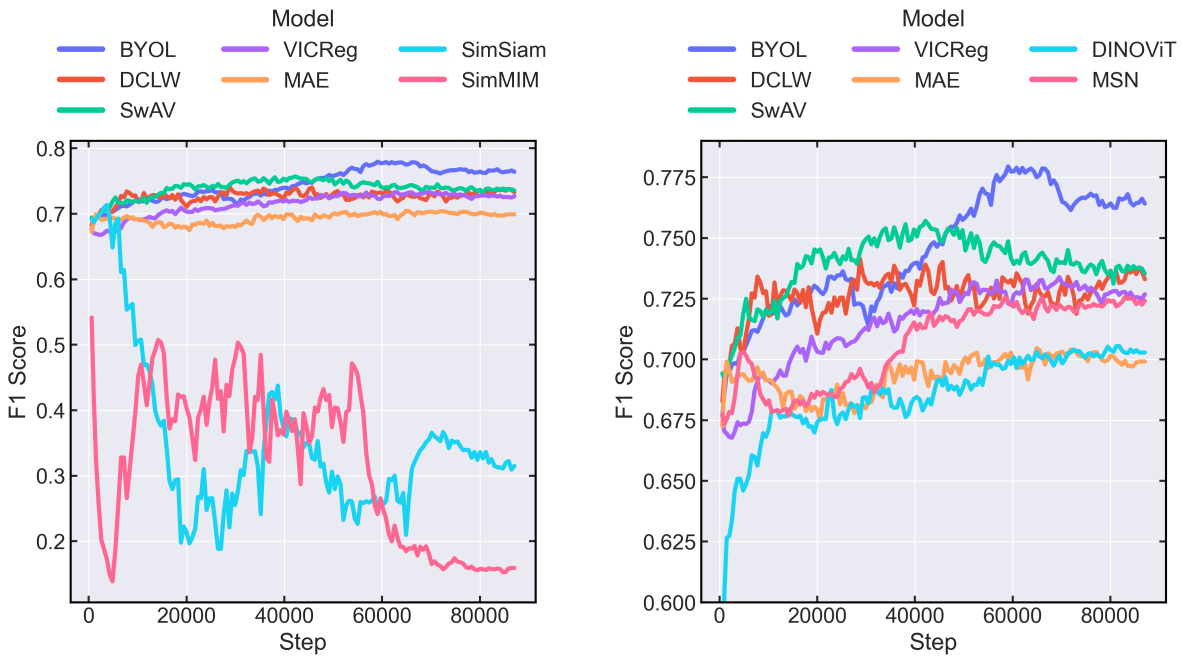


Figure 24: F1-score of selected models during training.[1] For clarity, an exponential moving average is shown. (a) includes the worst-performing frameworks, SimSiam and SimMIM, while (b) is a focused view of the other models. DINOViT refers to a ViT-S/16 model trained using the DINO framework.

Nearly every framework shows strong k-NN evaluation performance on the WM-811K benchmark. Generally, most models reach macro accuracies and F1 scores between 70% to 75%. To contextualize these results, a ResNet-18 trained using the PIRL framework by Kahng et al. reaches 69.3% accuracy ***after*** the entire network is fine-tuned with 1% of the labels [22]. Additionally, Hu et al.'s SimCLR implementation achieves 74.64% accuracy ***after*** fitting a linear classifier with 1% of the labels [23]. Evaluation using any sort of supervision is omitted from this study because linear evaluation and fine-tuning performance are almost always lower-bounded by k-NN performance; supervision should never lead to worse performance than using no labels at all. In other words, these k-NN classification results reflect the *worst-case* performance of these models, yet they still consistently match or outperform the current state of the art with ***no supervision***.

SimSiam, FastSiam, and SimMIM are the only frameworks that struggle on the wafer map k-NN benchmark. The main difference between SimMIM and MAE is that the former uses a simple linear layer as a decoder, while the latter uses a lightweight ViT. Using just a linear layer is likely insufficient to effectively reconstruct images from encoded representations. It is worth noting that the reference code implementation of SimMIM does not use a linear layer as the decoder; instead, a single convolutional layer followed by pixel shuffling is used. Strangely, this is not discussed in the original publication [38]. Nevertheless, adding a higher-capacity decoder may bring the performance of SimMIM closer to MAE on wafer map data.

SimSiam, and by extension FastSiam, consistently suffered from representation collapse. In SSL, simply tracking the training loss is insufficient to determine whether pretraining has been effective. Trivial solutions

[1] Interactive plots for all models showing their accuracy, F1-score, loss, and representation standard deviations are hosted on Tensorboard at this link.

maximally satisfy many loss functions in joint embedding frameworks, namely negative cosine similarity which is used in BYOL and SimSiam. In addition to tracking the loss to monitor instabilities in training, the standard deviations of the L2-normalized features can be monitored [41]. In representation collapse, the features are the same regardless of the inputs, so a standard deviation of zero over all samples in a batch indicates collapse. Figure 25 shows that training loss can be deceiving, but tracking the standard deviations of features clearly reveals representation collapse in SimSiam.

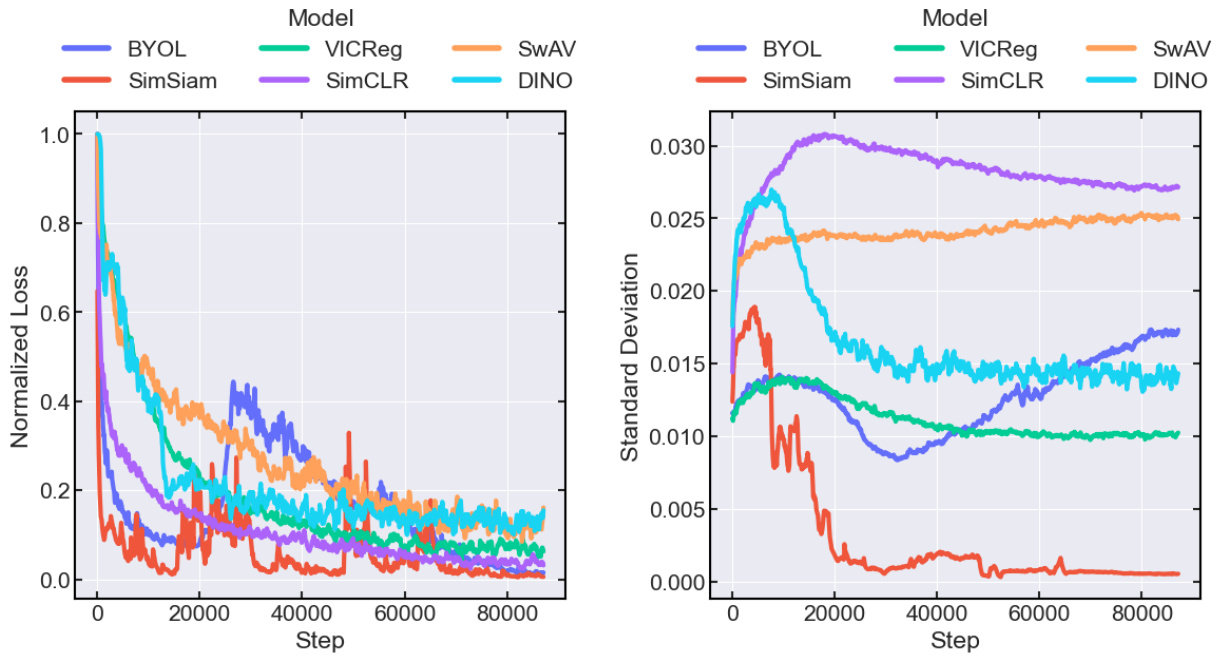


Figure 25: (a) shows the training loss of different models over the course of the benchmark. FastSiam shows nearly identical behavior to SimSiam, so it is omitted. Because each framework uses different loss functions, loss values are min-max normalized for the sake of comparison. (b) shows the standard deviation of the L2-normalized features during training. A value near zero indicates collapse. For both plots, an exponential moving average is shown for clarity.

The only difference between BYOL and SimSiam is that the latter removes the momentum encoder. BYOL reaches the highest performance on the k-NN benchmark, yet this small change leads to representation collapse for SimSiam and FastSiam. With the exact same network architecture and hyperparameter setup, these frameworks do not collapse on datasets of digital images such as CIFAR-10 and Imagenette [64], [65]. Natural images have a continuous range of pixel values from 0 to 255, but wafer maps only have three discrete pixel values: $(0, 128, 255)$. Thus, it is likely that the SimSiam architecture leads to feature collapse due to the discrete nature of pixel values. In other words, the convolutional layers in SimSiam's backbone network likely produce dead activation maps across the channels. This is verified in subsection 4.2

On popular benchmark datasets such as CIFAR-10 and ImageNet where categories are not ambiguous, most SSL frameworks' k-NN evaluation performance generally increases as models are trained for longer [64]. Surprisingly, this is not observed here. Performance for most models seems to plateau near 70% to 75% accuracy and F1 score. This suggests that the categories found through SSL may not be exactly the same as the manually created categories of the WM-811K dataset. Confusion matrices like the one shown in

Figure 26 show which classes are mislabeled most often. Perfect classification leads to values of 1.0 along the main diagonal and 0 everywhere else. For most models, the categories mislabeled most often by the k-NN classifier were "Edge-Loc," "Loc," and "Scratch." Localized and edge-localized defects are very ambiguous classes. Many wafer maps from these classes are arguably categorized better as center, donut, or scratch-type. Interestingly, this is very similar to how the scratch-type defects are misclassified by the k-NN classifiers. The row for the scratch-type label in Figure 26 shows that SwAV features for wafer maps of the scratch category are very close to center, edge-loc, loc, and none-type features. The confusion matrices for all models can be found in Appendix C. Aside from SimMIM, SimSiam, and FastSiam which had unstable training, all frameworks have very similar predictions. Because SSL leads to bottom-up categorization based on data-driven association, this strongly suggests that the apparent misclassifications are due to SSL-created categories being compared to ambiguous ground truths rather than issues in pretraining.

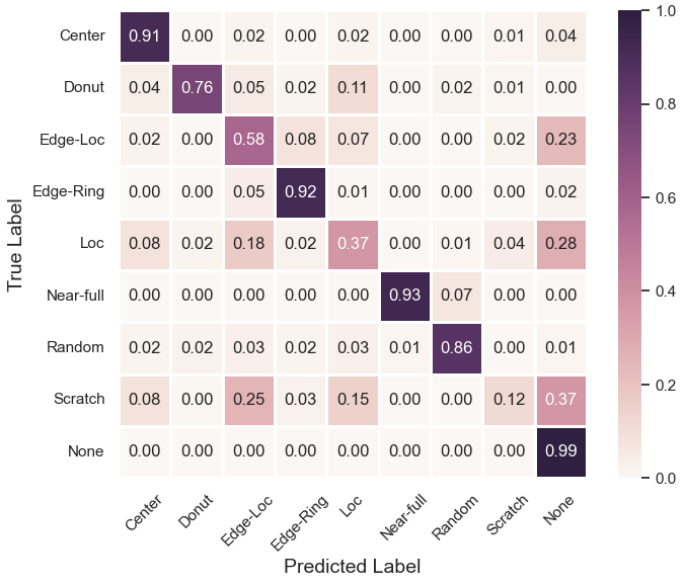


Figure 26: Confusion matrix of SwAV computed at the end of the k-NN benchmark. Values are normalized by the rows (ground truths).

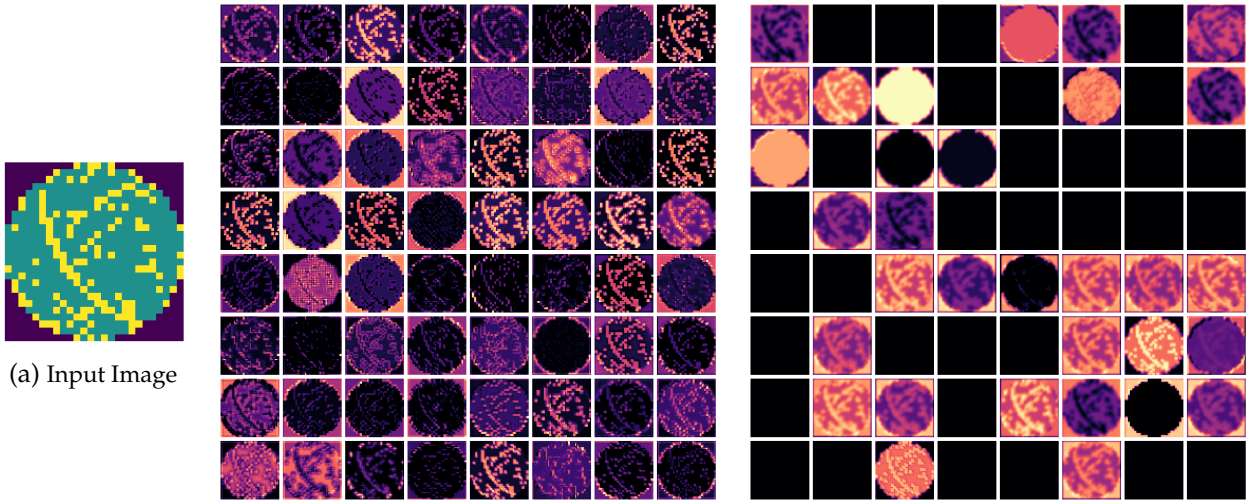
Since the WM-811K dataset has issues in labeling, it is difficult to directly compare SSL frameworks using quantitative analysis alone. Generally, the k-NN benchmarks show that most frameworks perform similarly for classification tasks. DCLW seems to be more stable in training than SimCLR and MoCo, making it a better choice if contrastive learning is desired. Comparing contrastive and non-contrastive joint embedding techniques with these benchmark results is difficult, but they seem to perform comparably. Theoretically, non-contrastive techniques should scale better, and redundancy reduction techniques like VICReg should most consistently avoid representation collapse. However, it has also been shown by Garrido et al. that careful network design choices and hyperparameter tuning can close the gap between contrastive and non-contrastive SSL [66]. This seems to be true for WM-811K. The top-performing models from every family of joint embedding techniques perform quite similarly. Even MSN and PMSN have comparable results despite the latter being theoretically better suited for class-imbalanced data. Performance gaps are observed between these two methods in the high-data regime [19], so it is likely that WM-811K is not large enough or does not have enough semantic concepts for the heavy class imbalance to really hinder effective SSL. On the other hand, masked image modeling generally seems to lead to slightly lower classification performance.

This observation aligns with benchmark results on natural images; MAE generally requires more fine-tuning than joint embedding methods to reach competitive performance on downstream tasks [35].

4.2 Visual Explanations of What Self-Supervised Models "See"

Deep learning is notorious for producing black-box models that are hard to interpret. If self-supervised models are never explicitly shown what to look for in images, what exactly are they "looking for"? Visualizations of the inner workings of backbone models can help provide qualitative insights into the learning behavior of self-supervised models on wafer map data.

Feature Activation Visualizations Qualitative analysis of a ResNet-18's feature maps can show whether the backbone network used in joint embedding is learning discriminative features during self-supervised pretraining. As previously mentioned, SimSiam and FastSiam suffer from representation collapse that may be linked to feature collapse due to the discrete nature of wafer map pixel values. Figure 27 compares the activation maps for BYOL and SimSiam. The output of a ResNet-18's first convolutional layer is 64 feature maps, and each activation map is visualized in small squares to show what parts of an input image activate neurons in that layer. SimSiam suffers from dead neurons in just the first convolutional layer, while BYOL learns discriminative features. Deeper layers of the ResNet show similar behavior for BYOL and even more pronounced feature collapse for SimSiam and FastSiam.

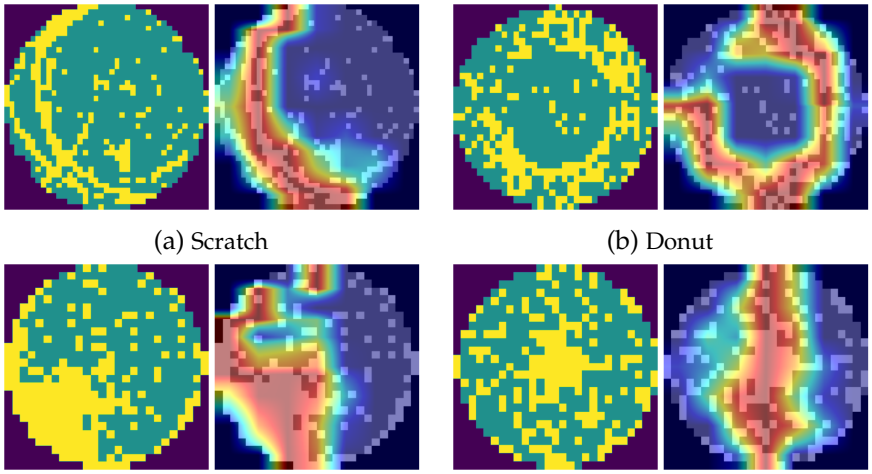


(a) Input Image (b) BYOL (c) SimSiam

Figure 27: Activation maps from the backbone's first convolutional layer in BYOL and SimSiam.

Grad-CAM Visualizations Feature activations can help show how whether a model learns discriminative features during training, but they do not provide high-level overviews of which parts of an image are most salient in making predictions. Grad-CAM is a popular tool that can be used in this regard, and it works by using the gradients of any target concept to show which regions of an image are most important in predicting specific classes [67]. However, SSL does not involve using knowledge of class concepts to produce a supervisory feedback signal. Eigen-CAM can be used instead as it does not rely on gradient backpropagation or class relevance scores [68]. This method works by computing and visualizing the

principal components of learned representations from convolutional layers. Figure 28 shows Eigen-CAM visualizations for BYOL, which illustrates that the backbone "sees" the most salient parts of the wafer map.



(a) Scratch (b) Donut

(c) Edge-Loc (d) Center

Figure 28: Eigen-CAM visualizations of the activations maps of a ResNet-18 on different failure patterns.

Self-Attention Maps ViTs like the ones used in DINO, MSN, and PMSN have no notion of feature maps as CNNs do, but the self-attention maps can be visualized to qualitatively analyze what parts of an input image a ViT pays attention to. Figure 29 shows the activation maps from the six attention heads of a ViT-S/16 trained using the DINO framework. Each attention head focuses on a slightly different aspect of the wafer. The first three attention heads seem to attend to different segments of the scratch and failing die, while the fifth and sixth attention heads attend strongly to yielding die. The fourth attention head seems to attend to the extreme outer edge and even the area outside the wafer map. Image embeddings from a ViT are given by the CLS token, a vector that includes information from all the attention heads. Thus, the feature vector for a wafer map incorporates information from different regions of the wafer.

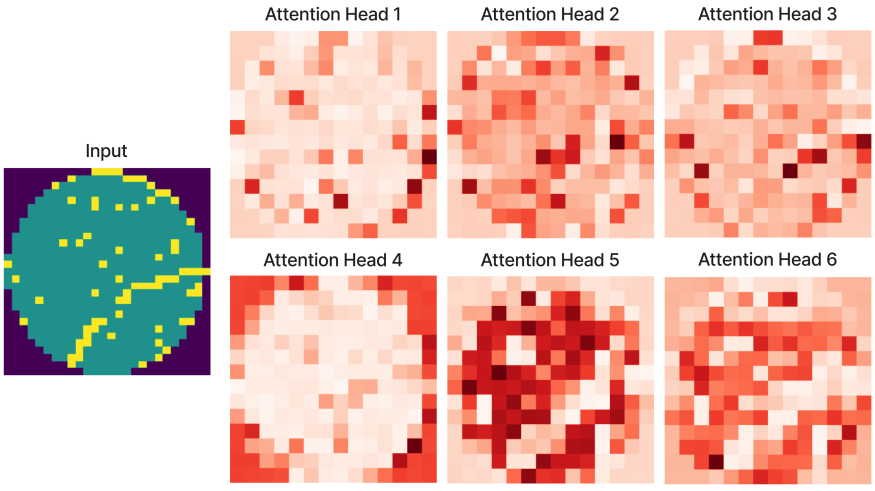
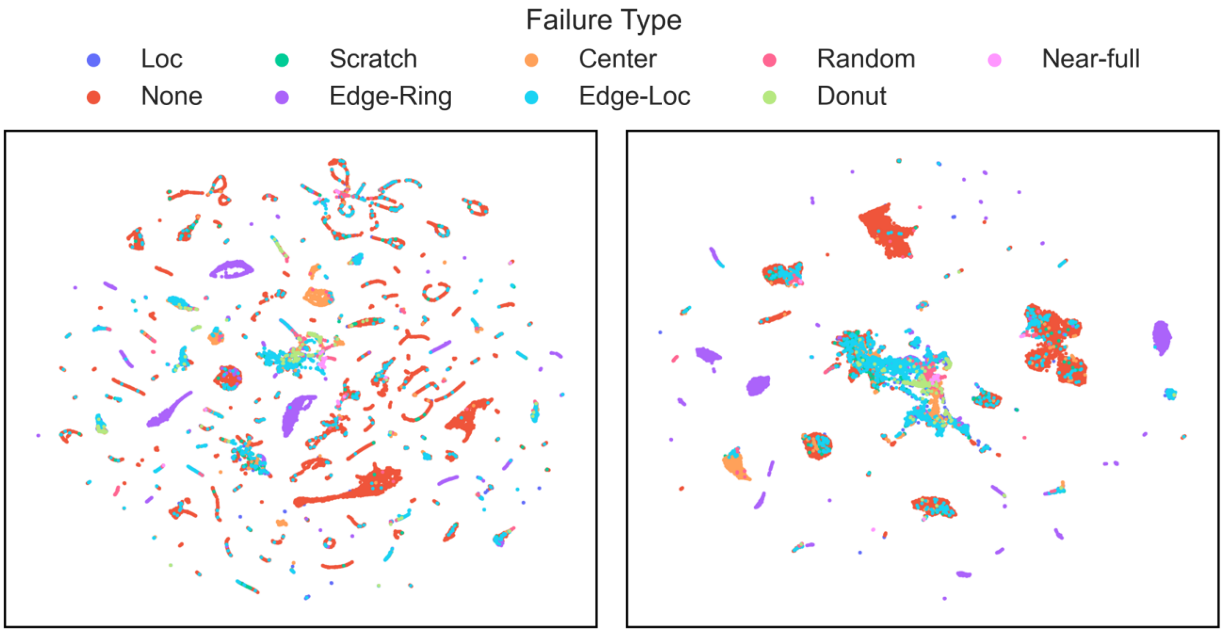


Figure 29: Attention maps for a scratched wafer from a ViT-S/16 trained using the DINO framework. Darker reds indicate higher attention. Each square in the attention maps represents a 16×16 patch of the input.

4.3 Analysis of Self-Supervised Representation Spaces

Explainability visualizations can show that a model can effectively "see" the important parts of a single wafer map, but they cannot be used to explain differences in how various SSL frameworks represent entire datasets. This is a critical component in determining whether a specific framework can effectively represent a wafer map as a vector, and it is perhaps even more important than quantitative benchmark comparisons such as k-NN evaluation given that WM-811K has ambiguous categories. UMAP [69] and DensMAP [70] dimensionality reduction can be used to this end to create visualizations of the representation space for each framework. Only a high-level overview of the most important results will be provided here, and DensMAP visualizations are omitted because they can be subjectively harder to interpret. UMAP and DensMAP visualizations for all frameworks are provided in Appendix D. Note that dimensionality reduction inherently involves some loss of information, and UMAP can sometimes lead to spurious clustering of the data.

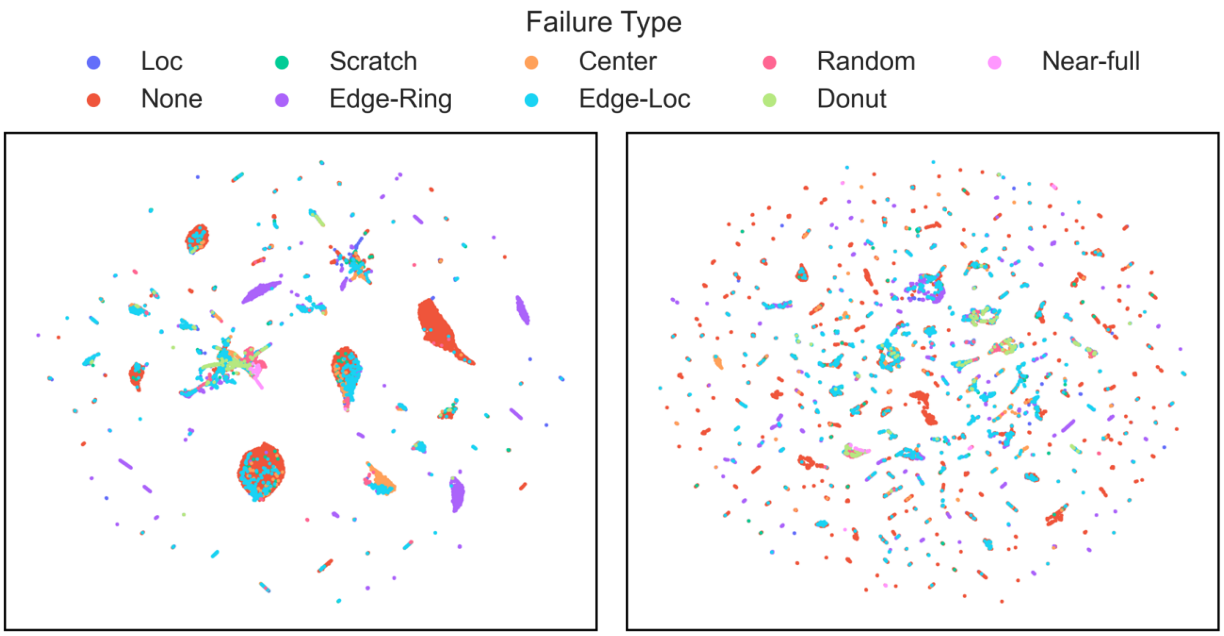


(a) SimCLR

(b) DCLW

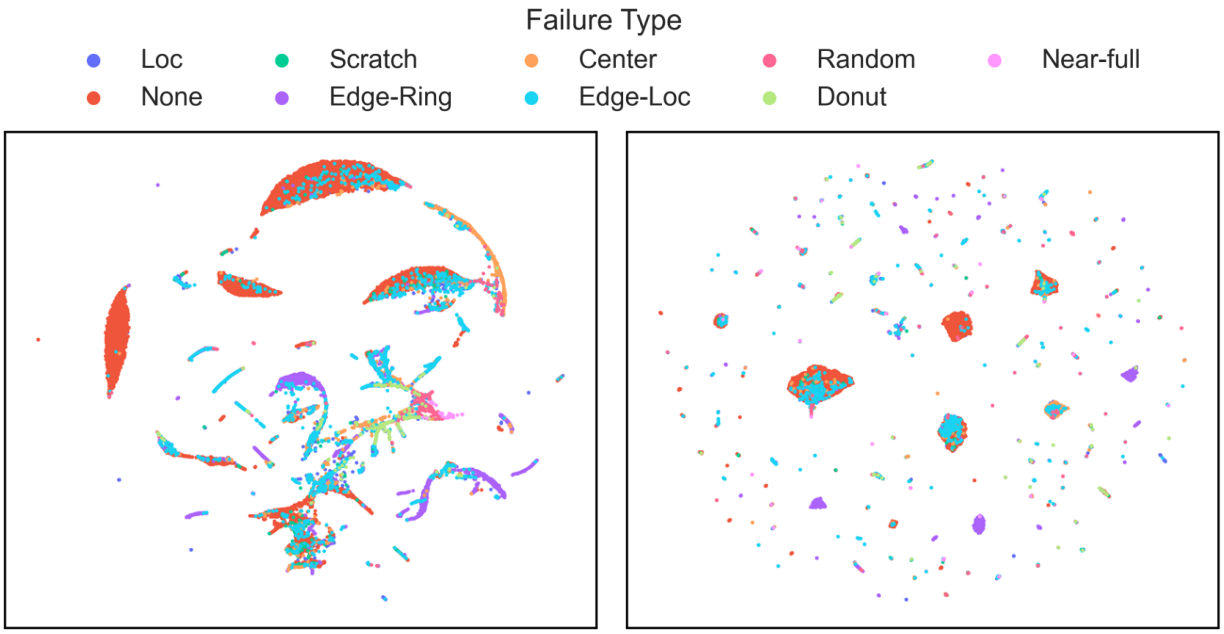
(c) Barlow Twins

(d) VICReg



(e) BYOL (f) SimSiam

(g) DINO (ViT-S/16) (h) SwAV



(i) PMSN (j) MAE

Figure 30: UMAP embeddings of the feature spaces of selected SSL frameworks.

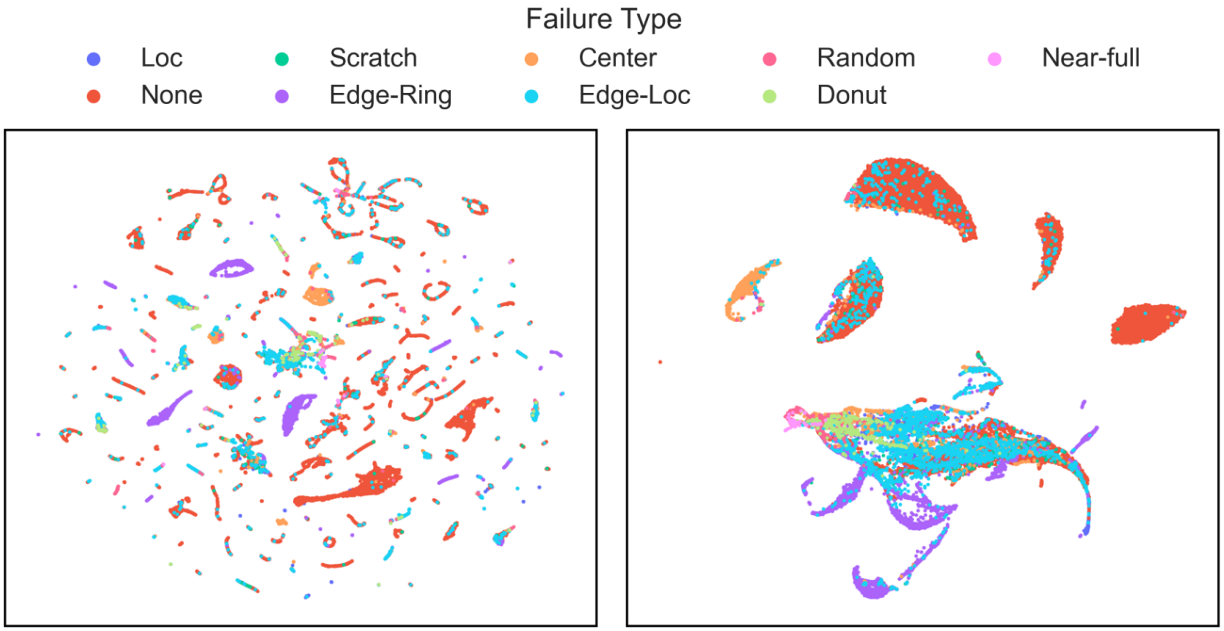
Joint Embedding vs. Generative SSL Masked image modeling generally leads to sparser representation spaces than joint embedding techniques. Frameworks such as MAE are designed to reconstruct images at a pixel level, which involves low-level information from the images that the model essentially "overfits" to. On the other hand, joint embedding techniques form predictions at an embedding level, so they are less prone to pay too much attention to small pixel-level details in images. As shown in Figure 30j, the UMAP embedding of MAE features generally seems sparser than that of the other frameworks.

Sparsity in Joint Embedding Feature Spaces Surprisingly, several joint embedding methods also display a rather sparse representation space. For example, SimCLR, VICReg, and BYOL have several small "islands" in their UMAP embeddings as shown in Figure 30a, Figure 30d, and Figure 30e, respectively. Within specific joint embedding techniques, DCLW seems slightly less prone to creating sparse feature spaces than other contrastive methods such as SimCLR as shown in Figure 30b. The same behavior is observed with Barlow Twins. Compared to the other redundancy reduction technique, VICReg, its feature space has far fewer subclusters as shown in Figure 30c. In the case of DCLW, the weighted decoupled contrastive learning loss may be less prone to overfit to pixel-level details than the InfoNCE loss, but this cannot be confirmed without an in-depth analysis that is beyond the scope of this work. On the other hand, the behavior of Barlow Twins is much simpler to explain. Barlow Twins is the only joint embedding method in this work that does not explicitly involve similarity maximization. Therefore, it should be less likely to create several small clusters in its feature space because embeddings are never directly "pulled together." In contrast, VICReg does include a similarity maximization term in its loss. Specifically, the invariance term is the mean squared error between embeddings of augmented image views, and the minimization of this term will bring samples closer together in the feature space.

Across different joint embedding techniques, DINO, SwAV, and PMSN/MSN consistently avoid creating

sparse representation spaces as shown in Figure 30g, Figure 30h, and Figure 30i respectively. Unlike the other frameworks, these use multiple image crops for joint embedding. DINO and SwAV use the multi-crop augmentation policy, while MSN and PMSN use multiple masking strategies in the form of random and focal masks. Both augmentation techniques provide the model with global and local information about the image, which likely makes it much more difficult for the model to overfit to pixel-level details.

Cropping vs Multi-Crop To determine whether cropping indeed prevents overfitting to pixel-level details, a few joint embedding models were benchmarked again with randomized cropping added to the augmentation pipeline. Specifically, a `RandomResizedCrop` transform was added to the end of the augmentation policy wherein an image would have a 50% chance of being cropped to a random scale within the bounds $(0.4, 1.0)$ with respect to the original image size. The resulting feature spaces are shown in Figure 31.

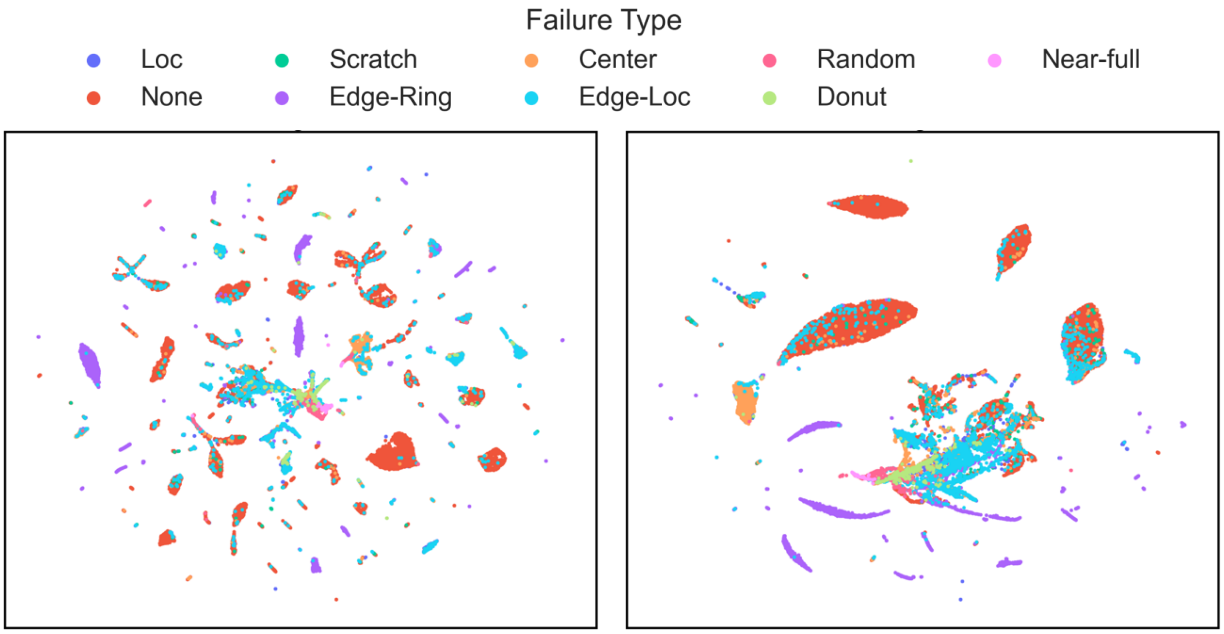


(a) SimCLR (no cropping)

(b) SimCLR (with cropping)

(c) BYOL (no cropping)

(d) BYOL (with cropping)



(e) VICReg (no cropping)

(f) VICReg (with cropping)

Figure 31: Comparison of feature spaces with and without image cropping in the augmentation policy.

Randomized cropping seems to completely eliminate the issue of sparse representation spaces, confirming that this issue is caused by pixel-level overfitting. However, the representation spaces for these joint embedding techniques seem qualitatively less well-separated than those of DINO, SwAV, MSN, and PMSN. Indeed, these models' k-NN evaluation performance is noticeably degraded by adding randomized cropping, as shown in Figure 32.

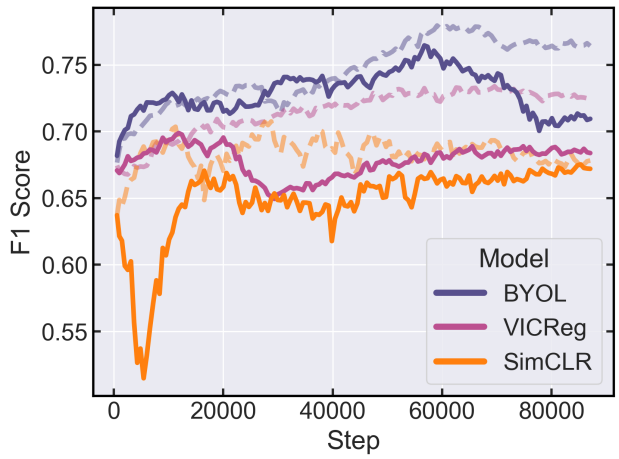


Figure 32: k-NN F1 performance with and without randomized cropping. Dotted lines indicate original k-NN performance without cropping, while solid lines indicate performance with cropping.

BYOL seems less affected by adding the cropping transform than SimCLR and VICReg, and this observation aligns with the fact that BYOL generally requires less strict augmentation policies than other joint embedding techniques due to its momentum encoder [52]. Nonetheless, the degradation in performance indicates that

using *multiple* image crops is key to stable training and well-separated but not overly sparse representation spaces. The issue with using randomized cropping with just two image views is that the cropped view may include irrelevant information about the wafer map's defect pattern. Joint embedding would then lead to similarity maximization between unrelated crops. Even with natural images, aggressive cropping policies can result in cropped views that are not semantically similar. This is especially true for photographs of diverse scenes, and it is a common criticism of many randomized augmentation policies that they are biased for "object-centric" datasets like ImageNet where images only have one main subject [71]. Multi-crop circumvents this issue by using multiple scales of information about the image, reducing the likelihood of a "bad" crop being used for similarity maximization. This explains how frameworks such as SwAV can be scaled to large datasets without the object-centric ImageNet bias; multi-crop prevents over-fitting to pixel-level details while stabilizing. It is worth noting that multi-crop can be adapted for use with joint embedding frameworks that typically only use two image views, but implementing this can be somewhat difficult.

Collapsed Representation Spaces By definition, representation collapse involves identical embeddings for semantically different inputs. Thus, collapsed representations in UMAP embeddings will often appear as large "blobs" without any clear distinctions between classes as shown in Figure 33.

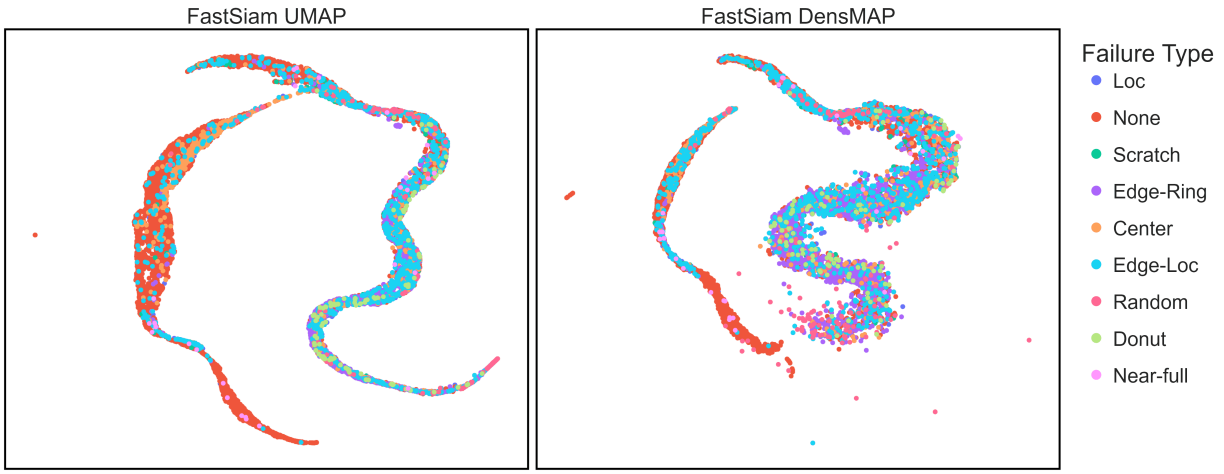


Figure 33: FastSiam trained on a subset of the data leads to a textbook case of representation collapse.

Fully Unsupervised Analysis Requires Interactivity Representation collapse sometimes leads to UMAP visualizations that appear sparse instead of collapsed. When trained on the full training dataset, SimSiam displays this behavior as shown in Figure 30f. When ground truths are available, it can clearly be seen that despite having several small clusters in the SimSiam feature space, the content of these clusters is nonsensical due to significant leakage of semantic concepts across different clusters. The results here include ground truths from the WM-811K dataset to help illustrate feature space separability or a lack thereof. In the real world, these labels will not be available. UMAP visualizations are still an excellent means of analyzing the representation space, but without information about class concepts, these visualizations should be made interactive. Specifically, a data scientist must manually probe the representation space to explore feature separability and cluster semantics. In this regard, it is often useful to create interactive visualizations of

feature spaces with the images overlaid onto the points of a scatter plot. Tensorflow's embedding projector is a good option, as are visualization libraries like Plotly [72] and Bokeh [73] for making scatter plots with hover overlays to display images and other metadata for each sample. Importantly, these qualitative techniques should be accompanied by quantitative analyses such as monitoring the standard deviations of the representations during training. Tracking the standard deviation is a simple, objective, and consistent way of determining representation collapse.

Probing the Semantics of the Representation Space As shown in Figure 34, a simple nearest-neighbor search in the representation space can be used to verify that the features obtained from self-supervised pretraining are semantically meaningful. A nearest-neighbor graph can be created using the representation vectors of any pretrained model to perform image retrieval in this manner. This type of similarity search can help engineers streamline root cause analysis by enabling fast and effective searches through large, unlabeled databases of wafer maps.

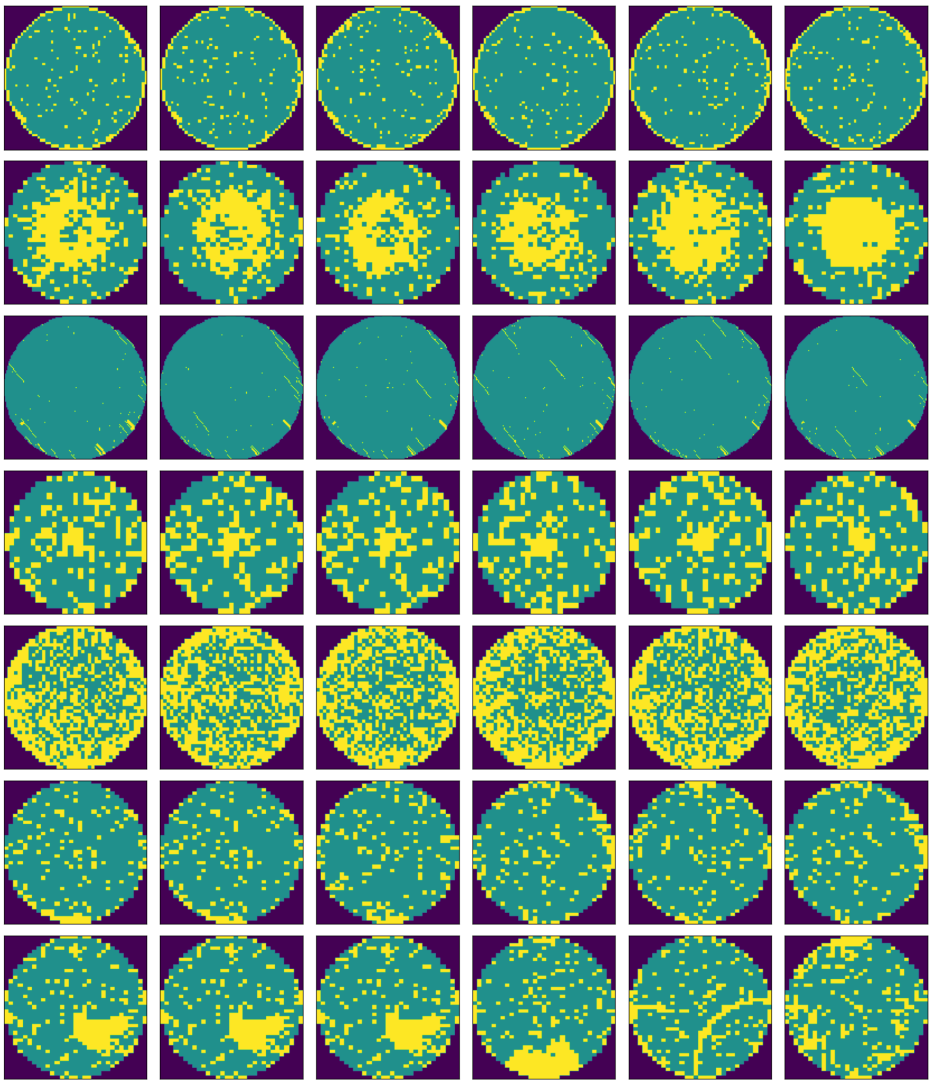


Figure 34: Query wafer maps (far left column) and their five nearest neighbors in the MSN feature space.

5 Future Directions

Realizing the Full Potential of WM-811K The vast majority of WM-811K is unlabeled, yet most machine learning studies using this dataset use just a small labeled fraction of the dataset. SSL can take advantage of the entire WM-811K dataset since self-supervision scales well to large datasets. This is an exciting opportunity for anyone with the necessary compute, since this full wafer map dataset is nearly 75% of the size of ImageNet-1k.

Beyond Binary Wafer Maps SSL leads to effective unsupervised representation learning of binary wafer maps, but multi-bin wafer maps are much more complex. Multi-bin wafer maps have more than three possible pixel values, though this shouldn't be an issue for deep learning architectures designed to work with color images. What makes multi-bin wafer maps interesting is the fact that there are typically multiple defect patterns on a single wafer map, and each defect may be comprised of several multiprobe bins. Currently, there are no publicly available datasets of multi-bin wafer maps. The closest is a synthetic dataset of mixed-type binary wafer map datasets called MixedWM38 created by generative adversarial networks [74]. Unfortunately, this dataset is not very realistic since it seems to suffer from mode collapse. Because there is little diversity among the wafer maps in MixedWM38, it is not a suitable proxy to investigate whether SSL can extend to multi-bin wafer maps. Nevertheless, analysis of multi-bin wafer maps data is an interesting and uncharted territory for SSL on semiconductor data.

Applications in Manufacturing and Non-Static Datasets The most immediate application of this work in manufacturing is creating an image retrieval system for wafer maps. It is worth noting, however, that manufacturing datasets are not static. Engineers may view new yield dashboards daily, and new wafers are constantly being probed in high-volume manufacturing scenarios. This is one of the reasons clustering of self-supervised feature spaces was not performed in this study. In semiconductor manufacturing, clustering on wafer map embeddings is only useful in the short term because today's wafer maps will be useless to an engineer in six months. What is more important is the *ability* to effectively represent wafer maps as vectors, and SSL can certainly do so. Moreover, SSL can better handle distribution shifts that would inevitably occur in real-world manufacturing scenarios compared to supervised learning. An interesting application of this is test-time-training [12], [18], in which a self-supervised model is constantly trained on new data to essentially update its feature space as new data is seen. A wafer map similarity search engine that could adapt to new trends in yield in this manner would be immensely useful to engineers.

Better Representations with Better SSL SSL is a rapidly moving area in machine learning. Several recent developments offer new opportunities to explore. In this work, joint embedding and predictive learning via masked image modeling were discussed as two very different methods of SSL. A recent framework unifies these methods into a joint embedding predictive architecture for images (I-JEPA) that claims to have the strengths of both [75]. Like joint embedding methods, I-JEPA does not overfit to low-level image details as masked autoencoders do. On the other hand, like masked image modeling frameworks, it can learn effective representations from images without the aggressive augmentation policies required by joint embedding methods.

Existing joint embedding methods are also improving. Very recently, the second version of DINO was released as the first "foundation image model," showing strong transfer learning performance across several

tasks and datasets [76]. This monumental progress suggests that SSL may soon eliminate the need for task-specific architectures like classification networks, segmentation networks, depth perception networks, etc., since giant self-supervised pretrained models can provide strong task-agnostic representations of images. Fine-tuning these larger models is out of the question for most of the research community, but steady progress has also been made on making SSL more accessible. Notably, joint embedding frameworks that use actual knowledge distillation (unlike DINO, SimSiam, and BYOL which are *inspired* by distillation) can effectively scale self-supervised pretraining to much smaller and faster backbone networks [77]. Additionally, frameworks such as EMP-SSL claim to offer significantly faster convergence through the use of improved multi-crop augmentations [78]. Because faster and more powerful SSL frameworks will be made available someday, it is important to investigate how well they can apply to domains such as wafer map data.

6 Conclusions

In semiconductor manufacturing, wafer maps are used to summarize which die on the wafer are defective. As image data, wafer maps must first be featurized into vector representations for downstream data mining tasks such as similarity search or classification. Supervised representation learning is untenable in manufacturing settings because wafer map data is often unlabeled. SSL offers a promising alternative without the need for large labeled datasets. Prior work on the WM-811K wafer map dataset has largely focused on fine-tuning classifiers on top of self-supervised pretrained encoders to achieve state-of-the-art classification performance. Here, the limits of SSL on wafer map data were investigated by determining whether the raw representation vectors obtained from self-supervised pretrained encoders are semantically meaningful without any supervised fine-tuning. A k-NN classification benchmark was created to compare the performance of fifteen different SSL frameworks, including several joint embedding frameworks and two masked image modeling frameworks. Extensive analyses showed that two methods do not extend well to wafer map data out-of-the-box: SimMIM and SimSiam. SimMIM likely suffers from a low-capacity decoder, while SimSiam suffers from representation collapse caused by feature activation collapse. Thorough qualitative analyses showed that self-supervised models pay attention to reasonably salient portions of the wafer maps to create image representations. Feature spaces of masked image modeling frameworks are prone to be overly sparse, as are the feature spaces of joint embedding techniques that do not use cropping in their randomized augmentation policies. The most effective SSL frameworks incorporate some type of multi-cropping. For wafer maps, the multi-crop augmentation policy prevents overfitting to pixel-level details while also providing information about global and local regions of defect patterns. The top-performing models such as BYOL, SwAV, DINO, and MSN show that self-supervised wafer map features are strong k-NN classifiers and clearly separable in high-dimensional space with no need for supervised fine-tuning. Similarity search on these self-supervised features shows that wafers are visually consistent when probing small neighborhoods of wafer maps in the feature space. Thus, SSL leads to effective representation learning of wafer maps, which can be immensely useful in manufacturing scenarios as it enables fast and effective searches of wafer maps with similar defect patterns in large, unlabeled datasets.

References

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

[2] Y. Wang and D. Ni, "Multi-bin wafer maps defect patterns classification," in *2019 IEEE International Conference on Smart Manufacturing, Industrial & Logistics Engineering (SMILE)*, Apr. 2019, pp. 48–52. DOI: 10.1109/SMILE45626.2019.8965299.

[3] 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.

[4] D. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, Sep. 1999, 1150–1157 vol.2. DOI: 10.1109/ICCV.1999.790410.

[5] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," en, *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004, Number: 2, ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94.

[6] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on statistical learning in computer vision, ECCV*, Issue: 1-22, vol. 1, Prague, 2004, pp. 1–2.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, ISSN: 1063-6919, Jun. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[8] J. Stuckner, B. Harder, and T. M. Smith, "Microstructure segmentation with deep learning encoders pre-trained on a large microscopy dataset," en, *npj Computational Materials*, vol. 8, no. 1, pp. 1–12, Sep. 2022, Number: 1 Publisher: Nature Publishing Group, ISSN: 2057-3960. DOI: 10.1038/s41524-022-00878-5.

[9] M. Huh, P. Agrawal, and A. A. Efros, *What makes ImageNet good for transfer learning?* arXiv:1608.08614 [cs], Dec. 2016. DOI: 10.48550/arXiv.1608.08614.

[10] U. Batool, M. I. Shapiai, M. Tahir, Z. H. Ismail, N. J. Zakaria, and A. Elfakharany, "A Systematic Review of Deep Learning for Silicon Wafer Defect Recognition," *IEEE Access*, vol. 9, pp. 116 572–116 593, 2021, Conference Name: IEEE Access, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3106171.

[11] T. Kim and K. Behdinan, "Advances in machine learning and deep learning applications towards wafer map defect recognition and classification: A review," en, *Journal of Intelligent Manufacturing*, Aug. 2022, ISSN: 1572-8145. DOI: 10.1007/s10845-022-01994-1.

[12] Y. Sun, X. Wang, Z. Liu, J. Miller, A. A. Efros, and M. Hardt, *Test-Time Training with Self-Supervision for Generalization under Distribution Shifts*, arXiv:1909.13231 [cs, stat], Jul. 2020. DOI: 10.48550/arXiv.1909.13231.

[13] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Technical report, University of Toronto, Toronto, Ontario, Tech. Rep. 0, 2009, Backup Publisher: University of Toronto.

[14] A. Torralba, R. Fergus, and W. T. Freeman, "80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008, Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2008.128.

[15] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar, *Do CIFAR-10 Classifiers Generalize to CIFAR-10?* arXiv:1806.00451 [cs, stat], Jun. 2018. DOI: 10.48550/arXiv.1806.00451.

[16] Y. Yang and Z. Xu, *Rethinking the Value of Labels for Improving Class-Imbalanced Learning*, Issue: arXiv:2006.07529 arXiv:2006.07529 [cs, stat], Sep. 2020. DOI: 10.48550/arXiv.2006.07529.

[17] H. Liu, J. Z. HaoChen, A. Gaidon, and T. Ma, *Self-supervised Learning is More Robust to Dataset Imbalance*, Issue: arXiv:2110.05025 arXiv:2110.05025 [cs, stat], May 2022. DOI: 10.48550/arXiv.2110.05025.

[18] Y. Liu, P. Kothari, B. van Delft, B. Bellot-Gurlet, T. Mordan, and A. Alahi, "TTT++: When Does Self-Supervised Test-Time Training Fail or Thrive?" In *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., 2021, pp. 21 808–21 820.

[19] M. Assran, R. Balestriero, Q. Duval, *et al.*, *The Hidden Uniform Cluster Prior in Self-Supervised Learning*, Issue: arXiv:2210.07277 arXiv:2210.07277 [cs], Oct. 2022. DOI: 10.48550/arXiv.2210.07277.

[20] P. Goyal, Q. Duval, I. Seessel, *et al.*, *Vision Models Are More Robust And Fair When Pretrained On Uncurated Images Without Supervision*, Issue: arXiv:2202.08360 arXiv:2202.08360 [cs], Feb. 2022. DOI: 10.48550/arXiv.2202.08360.

[21] S. Mo, J.-C. Su, C.-Y. Ma, *et al.*, "RoPAWS: Robust Semi-supervised Representation Learning from Uncurated Data," en, Feb. 2023.

[22] 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.

[23] 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.

[24] M. Ranzato, Y.-L. Boureau, S. Chopra, and Y. LeCun, "A Unified Energy-Based Framework for Unsupervised Learning," in *Proceedings of the Eleventh International Conference on Artificial Intelligence and Statistics*, M. Meila and X. Shen, Eds., ser. Proceedings of Machine Learning Research, vol. 2, San Juan, Puerto Rico: PMLR, Mar. 2007, pp. 371–379.

[25] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, Issue: arXiv:1810.04805 arXiv:1810.04805 [cs], May 2019. DOI: 10.48550/arXiv.1810.04805.

[26] T. B. Brown, B. Mann, N. Ryder, *et al.*, *Language Models are Few-Shot Learners*, Issue: arXiv:2005.14165 arXiv:2005.14165 [cs], Jul. 2020. DOI: 10.48550/arXiv.2005.14165.

[27] H. Touvron, T. Lavril, G. Izacard, *et al.*, *LLaMA: Open and Efficient Foundation Language Models*, arXiv:2302.13971 [cs], Feb. 2023. DOI: 10.48550/arXiv.2302.13971.

[28] OpenAI, *GPT-4 Technical Report*, arXiv:2303.08774 [cs], Mar. 2023. DOI: 10.48550/arXiv.2303.08774.

[29] S. Gururangan, A. Marasović, S. Swayamdipta, *et al.*, *Don't Stop Pretraining: Adapt Language Models to Domains and Tasks*, arXiv:2004.10964 [cs], May 2020. DOI: 10.48550/arXiv.2004.10964.

[30] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*, ser. ICML '08, New York, NY, USA: Association for Computing Machinery, Jul. 2008, pp. 1096–1103, ISBN: 978-1-60558-205-4. DOI: 10.1145/1390156.1390294.

[31] S. Gidaris, P. Singh, and N. Komodakis, *Unsupervised Representation Learning by Predicting Image Rotations*, Issue: arXiv:1803.07728 arXiv:1803.07728 [cs], Mar. 2018. DOI: 10.48550/arXiv.1803.07728.

[32] M. Noroozi and P. Favaro, *Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles*, Issue: arXiv:1603.09246 arXiv:1603.09246 [cs], Aug. 2017. DOI: 10.48550/arXiv.1603.09246.

[33] R. Zhang, P. Isola, and A. A. Efros, "Colorful Image Colorization," en, in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2016, pp. 649–666, ISBN: 978-3-319-46487-9. DOI: 10.1007/978-3-319-46487-9_40.

[34] M. Assran, M. Caron, I. Misra, *et al.*, *Masked Siamese Networks for Label-Efficient Learning*, Issue: arXiv:2204.07141 arXiv:2204.07141 [cs, eess], Apr. 2022. DOI: 10.48550/arXiv.2204.07141.

[35] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, *Masked Autoencoders Are Scalable Vision Learners*, Issue: arXiv:2111.06377 arXiv:2111.06377 [cs], Dec. 2021. DOI: 10.48550/arXiv.2111.06377.

[36] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*, Issue: arXiv:2010.11929 arXiv:2010.11929 [cs], Jun. 2021. DOI: 10.48550/arXiv.2010.11929.

[37] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[38] Z. Xie, Z. Zhang, Y. Cao, *et al.*, *SimMIM: A Simple Framework for Masked Image Modeling*, Issue: arXiv:2111.09886 arXiv:2111.09886 [cs], Apr. 2022. DOI: 10.48550/arXiv.2111.09886.

[39] K. He, X. Zhang, S. Ren, and J. Sun, *Deep Residual Learning for Image Recognition*, arXiv:1512.03385 [cs], Dec. 2015. DOI: 10.48550/arXiv.1512.03385.

[40] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, *A Simple Framework for Contrastive Learning of Visual Representations*, Issue: arXiv:2002.05709 Number: arXiv:2002.05709 arXiv:2002.05709 [cs, stat], Jun. 2020. DOI: 10.48550/arXiv.2002.05709.

[41] X. Chen and K. He, *Exploring Simple Siamese Representation Learning*, Issue: arXiv:2011.10566 arXiv:2011.10566 [cs], Nov. 2020. DOI: 10.48550/arXiv.2011.10566.

[42] I. Misra and L. van der Maaten, *Self-Supervised Learning of Pretext-Invariant Representations*, Issue: arXiv:1912.01991 arXiv:1912.01991 [cs], Dec. 2019. DOI: 10.48550/arXiv.1912.01991.

[43] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh and M. Titterington, Eds., ser. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 297–304.

[44] A. v. d. Oord, Y. Li, and O. Vinyals, *Representation Learning with Contrastive Predictive Coding*, arXiv:1807.03748 [cs, stat], Jan. 2019. DOI: 10.48550/arXiv.1807.03748.

[45] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton, *Big Self-Supervised Models are Strong Semi-Supervised Learners*, Issue: arXiv:2006.10029 Number: arXiv:2006.10029 arXiv:2006.10029 [cs, stat], Oct. 2020. DOI: 10.48550/arXiv.2006.10029.

[46] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, *Momentum Contrast for Unsupervised Visual Representation Learning*, arXiv:1911.05722 [cs], Mar. 2020. DOI: 10.48550/arXiv.1911.05722.

[47] X. Chen, H. Fan, R. Girshick, and K. He, *Improved Baselines with Momentum Contrastive Learning*, Issue: arXiv:2003.04297 Number: arXiv:2003.04297 arXiv:2003.04297 [cs], Mar. 2020. DOI: 10.48550/arXiv.2003.04297.

[48] X. Chen, S. Xie, and K. He, *An Empirical Study of Training Self-Supervised Vision Transformers*, arXiv:2104.02057 [cs], Aug. 2021. DOI: 10.48550/arXiv.2104.02057.

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

[50] 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 arXiv:2006.09882 [cs], Jan. 2021. DOI: 10.48550/arXiv.2006.09882.

[51] G. Hinton, O. Vinyals, and J. Dean, *Distilling the Knowledge in a Neural Network*, arXiv:1503.02531 [cs, stat], Mar. 2015. DOI: 10.48550/arXiv.1503.02531.

[52] 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 arXiv:2006.07733 [cs, stat], Sep. 2020. DOI: 10.48550/arXiv.2006.07733.

[53] M. Caron, H. Touvron, I. Misra, *et al.*, *Emerging Properties in Self-Supervised Vision Transformers*, Issue: arXiv:2104.14294 Number: arXiv:2104.14294 arXiv:2104.14294 [cs], May 2021. DOI: 10.48550/arXiv.2104.14294.

[54] H. B. Barlow *et al.*, "Possible principles underlying the transformation of sensory messages," *Sensory communication*, vol. 1, no. 01, pp. 217–233, 1961.

[55] J. Zbontar, L. Jing, I. Misra, Y. LeCun, and S. Deny, *Barlow Twins: Self-Supervised Learning via Redundancy Reduction*, Issue: arXiv:2103.03230 arXiv:2103.03230 [cs, q-bio], Jun. 2021. DOI: 10.48550/arXiv.2103.03230.

[56] A. Bardes, J. Ponce, and Y. LeCun, *VICReg: Variance-Invariance-Covariance Regularization for Self-Supervised Learning*, Issue: arXiv:2105.04906 arXiv:2105.04906 [cs], Jan. 2022. DOI: 10.48550/arXiv.2105.04906.

[57] Z. Wu, Y. Xiong, S. Yu, and D. Lin, *Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination*, arXiv:1805.01978 [cs], May 2018. DOI: 10.48550/arXiv.1805.01978.

[58] D. Pototzky, A. Sultan, and L. Schmidt-Thieme, "FastSiam: Resource-Efficient Self-supervised Learning on a Single GPU," en, in *Pattern Recognition*, B. Andres, F. Bernard, D. Cremers, S. Frintrop, B. Goldlücke, and I. Ihrke, Eds., ser. Lecture Notes in Computer Science, Cham: Springer International Publishing, 2022, pp. 53–67, ISBN: 978-3-031-16788-1. DOI: 10.1007/978-3-031-16788-1_4.

[59] I. Radosavovic, R. P. Kosaraju, R. Girshick, K. He, and P. Dollár, *Designing Network Design Spaces*, arXiv:2003.13678 [cs], Mar. 2020. DOI: 10.48550/arXiv.2003.13678.

[60] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, *A ConvNet for the 2020s*, Issue: arXiv:2201.03545 arXiv:2201.03545 [cs], Mar. 2022. DOI: 10.48550/arXiv.2201.03545.

[61] S. Woo, S. Debnath, R. Hu, *et al.*, *ConvNeXt V2: Co-designing and Scaling ConvNets with Masked Autoencoders*, Issue: arXiv:2301.00808 arXiv:2301.00808 [cs], Jan. 2023. DOI: 10.48550/arXiv.2301.00808.

[62] R. Wightman, H. Touvron, and H. Jégou, *ResNet strikes back: An improved training procedure in timm*, arXiv:2110.00476 [cs], Oct. 2021. DOI: 10.48550/arXiv.2110.00476.

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

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

[65] *Imagenette*, original-date: 2019-03-06T01:58:45Z, Dec. 2022.

[66] Q. Garrido, Y. Chen, A. Bardes, L. Najman, and Y. Lecun, *On the duality between contrastive and non-contrastive self-supervised learning*, arXiv:2206.02574 [cs], Oct. 2022. DOI: 10.48550/arXiv.2206.02574.

[67] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," *International Journal of Computer*

Vision, vol. 128, no. 2, pp. 336–359, Feb. 2020, arXiv:1610.02391 [cs], ISSN: 0920-5691, 1573-1405. DOI: 10.1007/s11263-019-01228-7.

[68] M. B. Muhammad and M. Yeasin, "Eigen-CAM: Class Activation Map using Principal Components," in *2020 International Joint Conference on Neural Networks (IJCNN)*, arXiv:2008.00299 [cs], Jul. 2020, pp. 1–7. DOI: 10.1109/IJCNN48605.2020.9206626.

[69] L. McInnes, J. Healy, and J. Melville, *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction*, arXiv:1802.03426 [cs, stat], Sep. 2020. DOI: 10.48550/arXiv.1802.03426.

[70] A. Narayan, B. Berger, and H. Cho, *Density-Preserving Data Visualization Unveils Dynamic Patterns of Single-Cell Transcriptomic Variability*, en, Pages: 2020.05.12.077776 Section: New Results, May 2020. DOI: 10.1101/2020.05.12.077776.

[71] S. Purushwalkam and A. Gupta, *Demystifying Contrastive Self-Supervised Learning: Invariances, Augmentations and Dataset Biases*, arXiv:2007.13916 [cs], Jul. 2020. DOI: 10.48550/arXiv.2007.13916.

[72] P. T. Inc., *Collaborative data science*, Place: Montreal, QC Publisher: Plotly Technologies Inc., 2015.

[73] Bokeh Development Team, "Bokeh: Python library for interactive visualization," manual, 2023.

[74] S. Nag, D. Makwana, S. C. T. R, S. Mittal, and C. K. Mohan, "WaferSegClassNet - A light-weight network for classification and segmentation of semiconductor wafer defects," en, *Computers in Industry*, vol. 142, p. 103 720, Nov. 2022, ISSN: 0166-3615. DOI: 10.1016/j.compind.2022.103720.

[75] M. Assran, Q. Duval, I. Misra, *et al.*, *Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture*, arXiv:2301.08243 [cs, eess], Apr. 2023. DOI: 10.48550/arXiv.2301.08243.

[76] M. Oquab, T. Darcet, T. Moutakanni, *et al.*, *DINOv2: Learning Robust Visual Features without Supervision*, arXiv:2304.07193 [cs], Apr. 2023. DOI: 10.48550/arXiv.2304.07193.

[77] Q. Duval, I. Misra, and N. Ballas, *A Simple Recipe for Competitive Low-compute Self supervised Vision Models*, arXiv:2301.09451 [cs], Jan. 2023. DOI: 10.48550/arXiv.2301.09451.

[78] S. Tong, Y. Chen, Y. Ma, and Y. Lecun, *EMP-SSL: Towards Self-Supervised Learning in One Training Epoch*, arXiv:2304.03977 [cs], Apr. 2023. DOI: 10.48550/arXiv.2304.03977.

A DPW Code

```python
def dpw_transform(original_wafer, scale):
    # Calculate the new dimensions of the wafer after scaling down
    h, w = original_wafer.shape
    new_h = int(h * scale)
    new_w = int(w * scale)
    new_dim = (new_h, new_w)

    # Find the indices of the passing/failing die in the original wafer
    passing_indices = np.argwhere(original_wafer == 128)
    failing_indices = np.argwhere(original_wafer == 255)

    # Find relative central coordinates of original wafer's passing/failing die
    pass_coords = (passing_indices + 0.5) / original_wafer.shape
    fail_coords = (failing_indices + 0.5) / original_wafer.shape

    # Calculate central coordinates of the new wafer's passing/failing die
    new_pass_coords = (pass_coords * new_dim).astype(int)
    new_fail_coords = (fail_coords * new_dim).astype(int)

    # Create the (new_h, new_w) wafer map
    new_matrix = np.zeros(new_dim, dtype=int)

    # Assign the passing and failing elements in the new wafer
    new_matrix[new_pass_coords[:, 0], new_pass_coords[:, 1]] = 128
    new_matrix[new_fail_coords[:, 0], new_fail_coords[:, 1]] = 255

    return new_matrix
```

Listing 1: Python code for a vectorized DPW transform using NumPy.

B Model Implementation Details

All models are optimized using a learning rate factor of `batch_size / 256`. We use a batch size of 64, so all base learning rates specified below are multiplied by 0.25.

SimCLR and DCLW For the projection head, we use dimensions $(512, 512, 128)$, corresponding to the feature, hidden, and embedding dimensions, respectively. We use the SGD optimizer with a base learning rate of 6×10^{-2}, a momentum of 0.9, and a weight decay of 5×10^{-4}. We also use a cosine annealing learning rate scheduler.

MoCo For the projection head, we use dimensions $(512, 2048, 128)$, corresponding to the feature, hidden, and embedding dimensions, respectively. We use a memory bank size of 4096 and a temperature of 0.1 for the NTXentLoss. We use the SGD optimizer 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.

SwAV For the projection head, we use dimensions $(512, 2048, 128)$, corresponding to the feature, hidden, and embedding dimensions, respectively. 3000 prototypes are used, and we use 2 high-resolution crops and

6 low-resolution crops. We use the Adam optimizer with a base learning rate of 1×10^{-3} and a weight decay of 1×10^{-6}, using a cosine annealing learning rate scheduler.

SimSiam and FastSiam For the projection head, we use dimensions $(512, 2048, 2048)$, corresponding to the feature, hidden, and embedding dimensions, respectively. We use dimensions $(2048, 512, 2048)$ for the prediction head, corresponding to the embedding, hidden, and prediction dimensions, respectively. We do not use learning rate scaling for these models due to training instability (that is, we don't multiply the base learning rate by the learning rate factor). We use the SGD optimizer 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.

BYOL For the projection head, we use dimensions $(512, 4096, 256)$, corresponding to the feature, hidden, and embedding dimensions, respectively. We use dimensions $(256, 4096, 256)$ for the prediction head, corresponding to the embedding, hidden, and prediction dimensions, respectively. Again, we use the SGD optimizer 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 Both the ResNet-18 and ViT-S/16 variants use the same projection head setup. The student and teacher networks' projection heads use dimensions $(2048, 256, 2048)$, corresponding to the feature, hidden, and embedding dimensions, respectively. The ResNet variant used batch normalization in the projection head, while the ViT variant did not. As with SwAV, we use 2 high-resolution crops and 6 low-resolution crops. For the ResNet variant, we use the SGD optimizer 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. For the ViT variant, we use the AdamW optimizer with a base learning rate of 1.5×10^{-4}, a momentum of 0.9, a weight decay of 0.05, and betas $(0.9, 0.95)$. A cosine warmup scheduler is used with 20 warmup epochs.

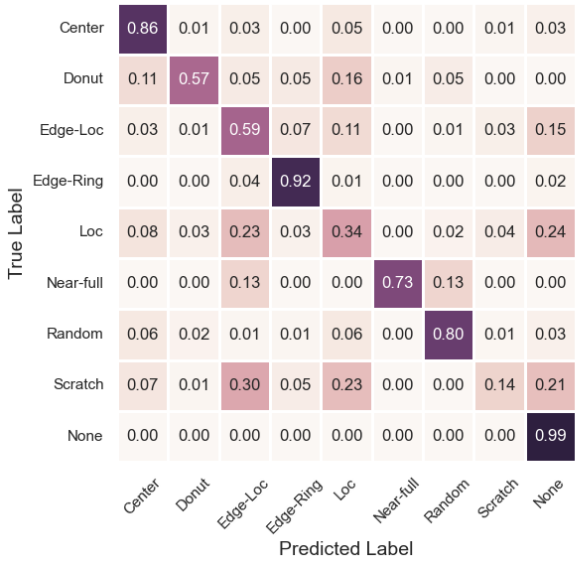
BarlowTwins and VICReg For the projection head, we use dimensions $(512, 2048, 2048)$, corresponding to the feature, hidden, and embedding dimensions, respectively. For the optimizer, we use the LARS optimizer with a base learning rate of 0.2, a momentum of 0.9, and a weight decay of 1.5×10^{-6}. A cosine warmup scheduler is used with 20 warmup epochs. VICReg is trained the same way, except we use a base learning rate of 0.3 instead.

MSN and PMSN For the projection head, we use dimensions $(384, 2048, 256)$, corresponding to the feature, hidden, and embedding dimensions, respectively. We initialize the prototypes using the weights of a linear layer, specifically `nn.Linear(256, 1024, bias=False)`. For the optimizer, we use AdamW with a base learning rate of 1.5×10^{-4}, a weight decay of 0.05, and betas $(0.9, 0.95)$. A cosine warmup scheduler is used with 15 warmup epochs.

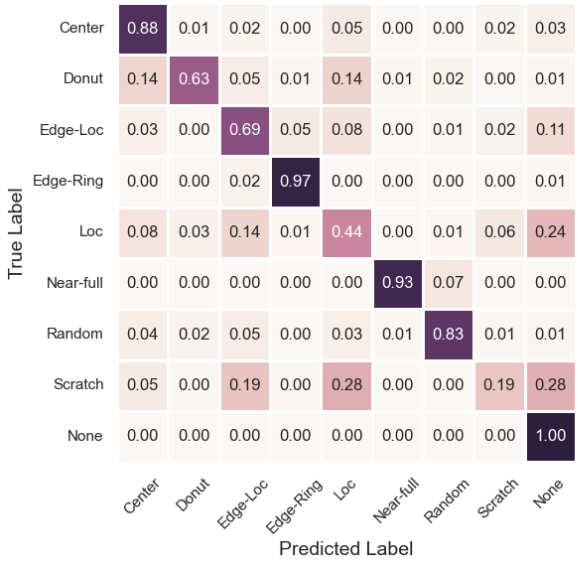
MAE We use ViT-B/32 as the encoder. For the decoder, we use a ViT with a sequence length of 32, 1 layer, 16 heads, hidden dimension of 512, MLP dimension of 512×4, output dimension of $32^2 \times 3$, and no dropout. For the optimizer, we again use AdamW with a base learning rate of 1.5×10^{-4}, a weight decay of 0.05, and betas $(0.9, 0.95)$. A cosine warmup scheduler is used with 20 warmup epochs.

SimMIM We use ViT-B/32 as the encoder and a simple linear layer with dimensions $(768, 32^2 \times 3)$ corresponding to the input and output features, respectively. For the optimizer, we use AdamW with a base learning rate of 8×10^{-4}, a weight decay of 0.05, and betas $(0.9, 0.95)$. A cosine warmup scheduler is used with 20 warmup epochs.

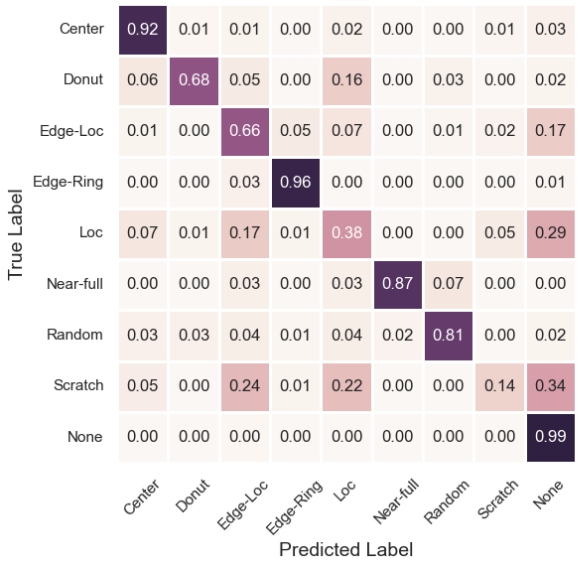
C k-NN Confusion Matrices



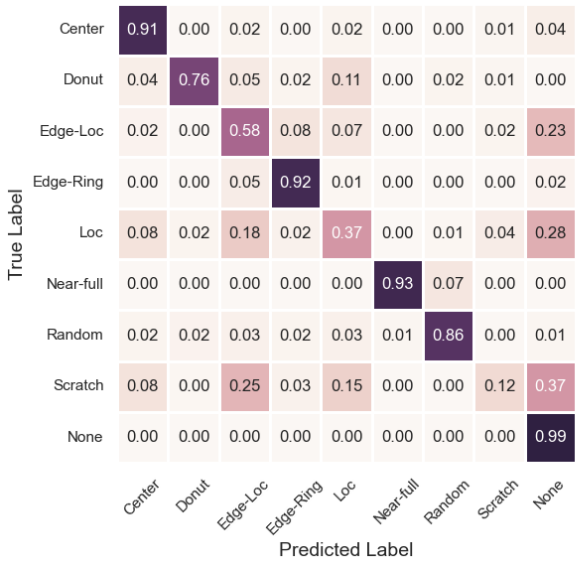
(a) SimCLR



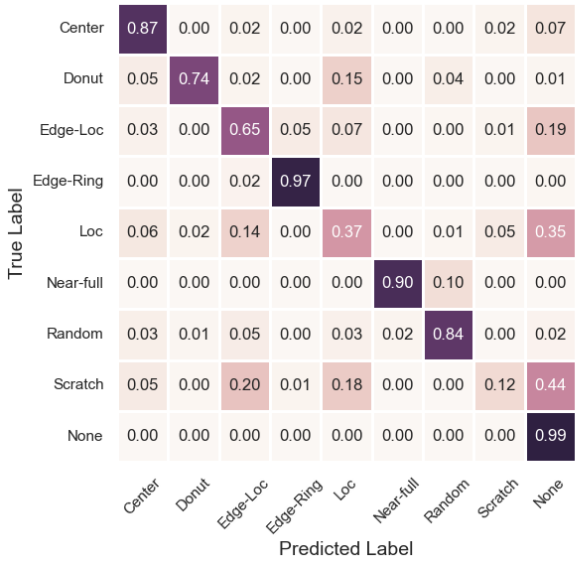
(b) MoCo



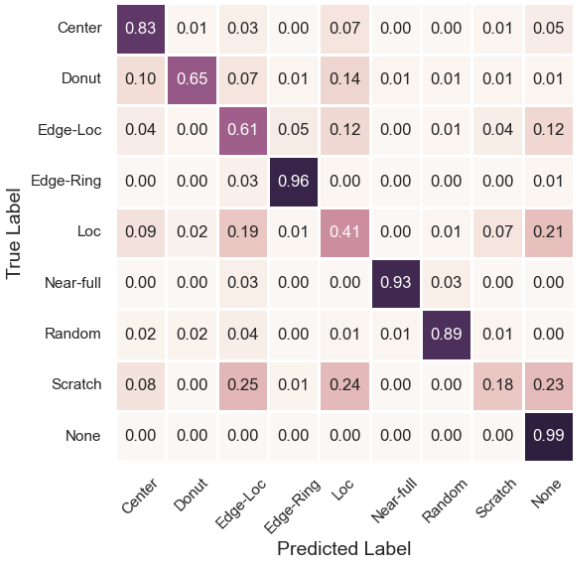
(c) DCLW



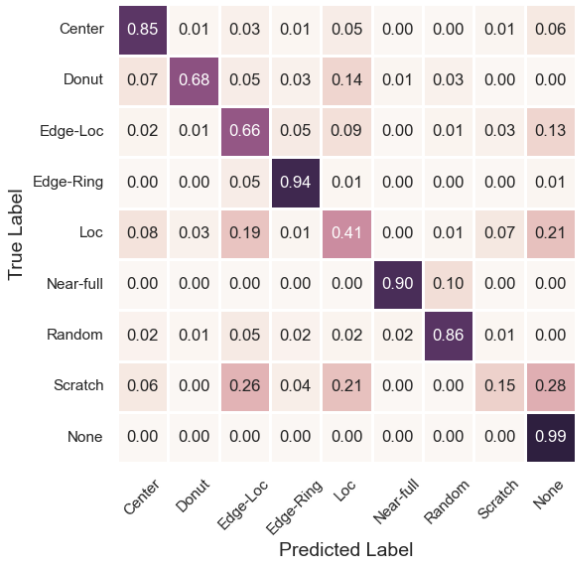
(d) SwAV



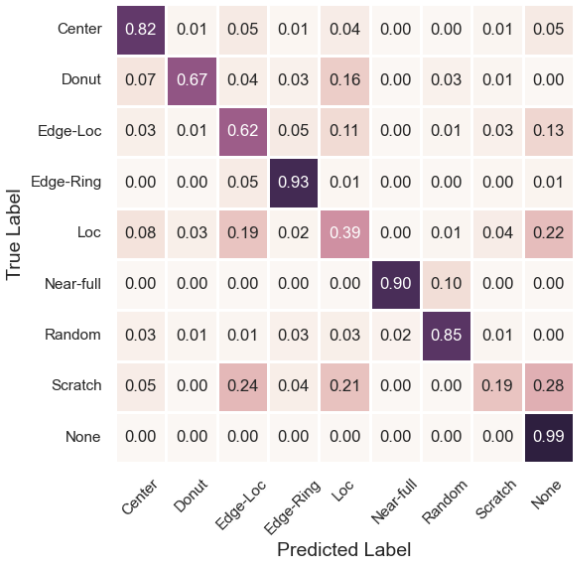
(e) Barlow Twins



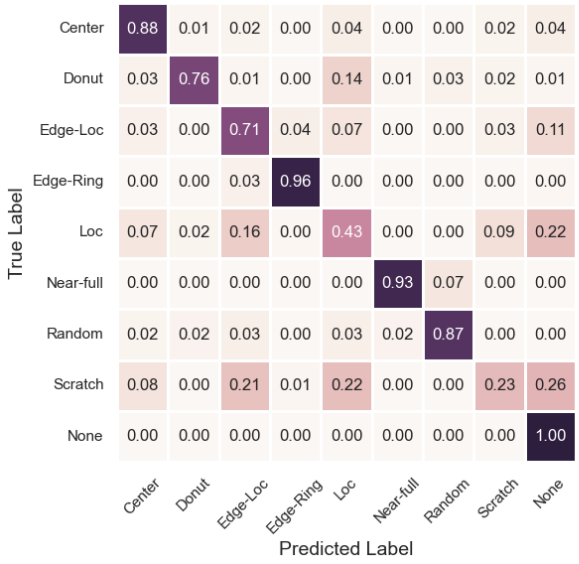
(f) VICReg



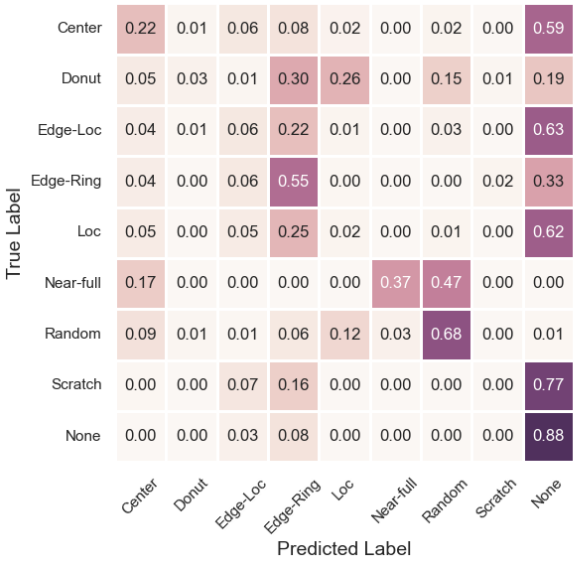
(g) MSN



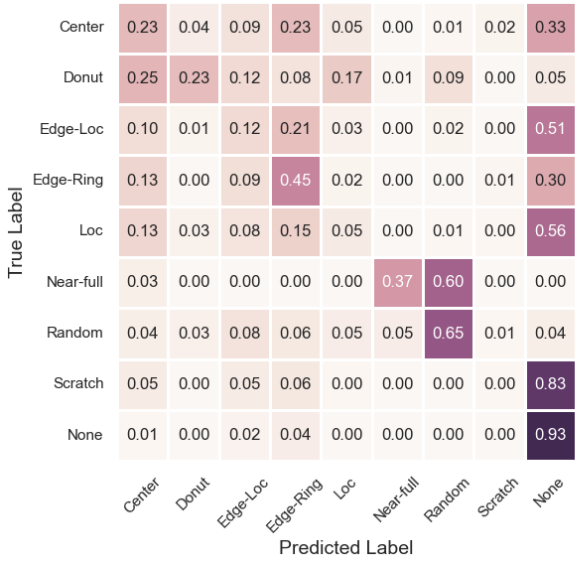
(h) PMSN



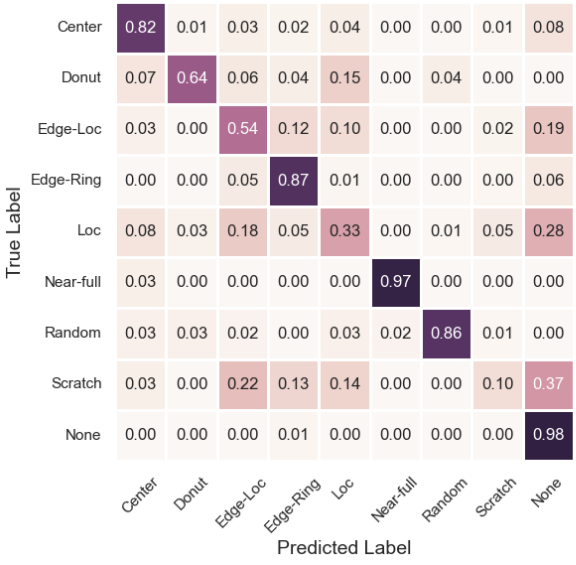
(i) BYOL



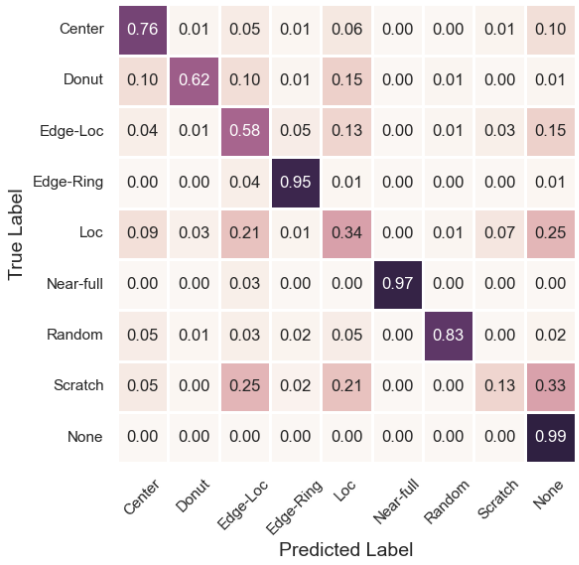
(j) SimSiam



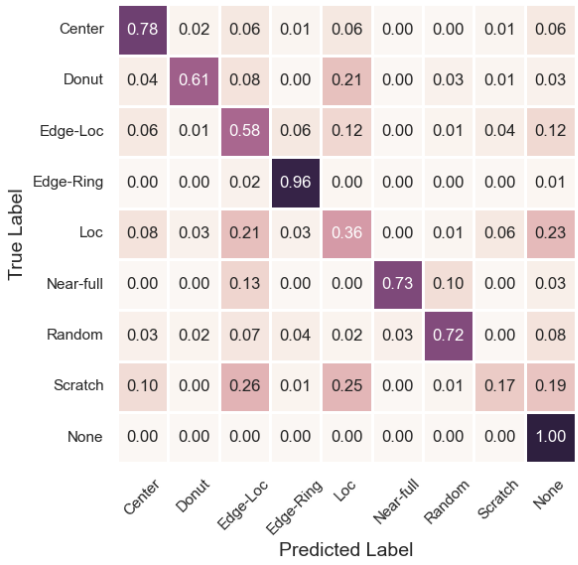
(k) FastSiam



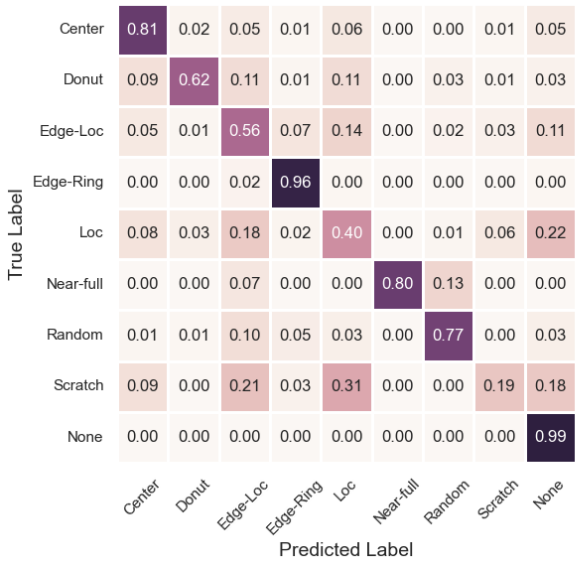
(l) DINO



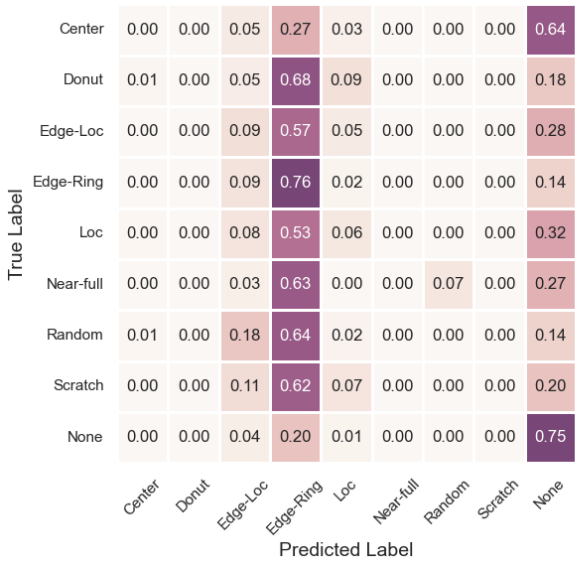
(m) DINO (ViT-S/16)



(n) MAE (minimal transforms)



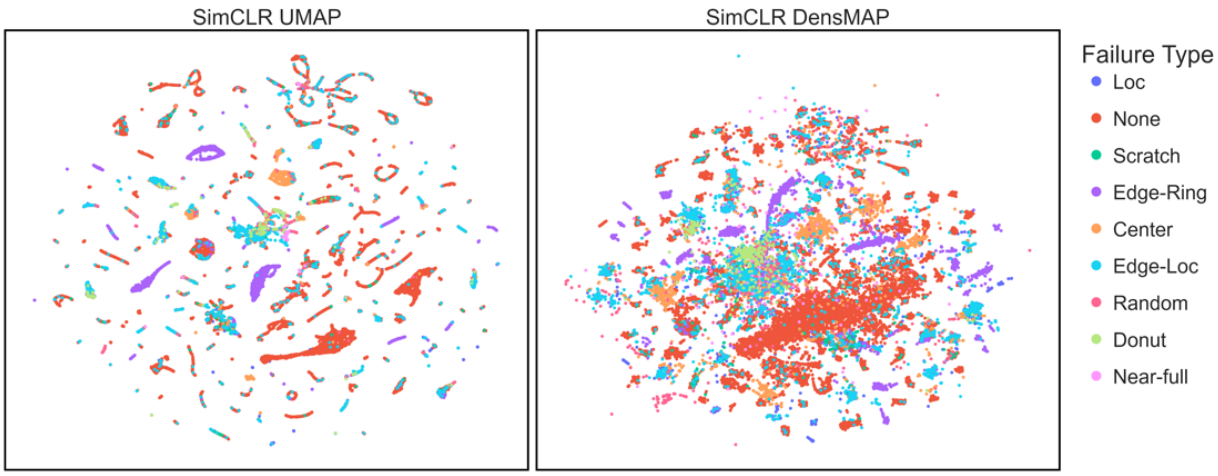
(o) MAE (full transforms)

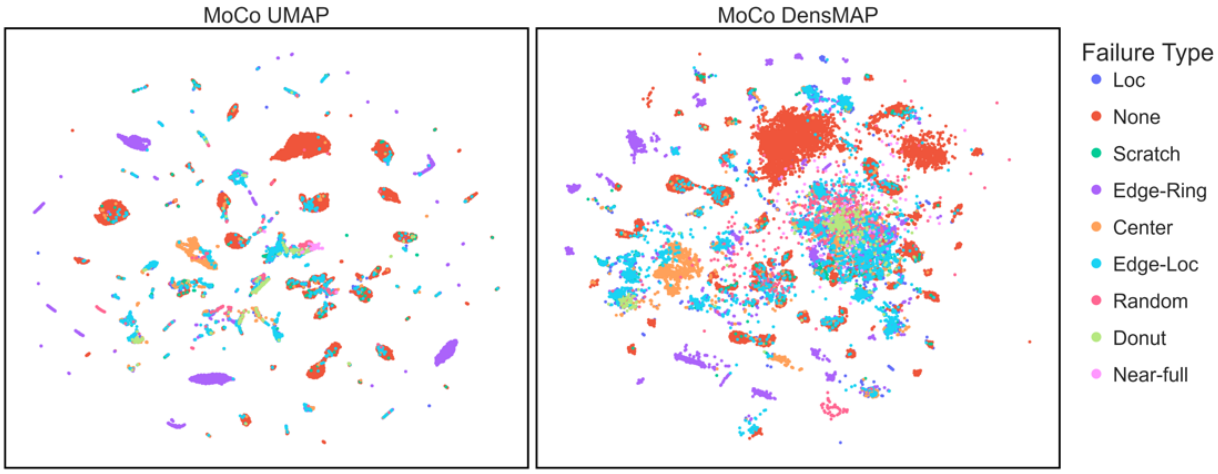


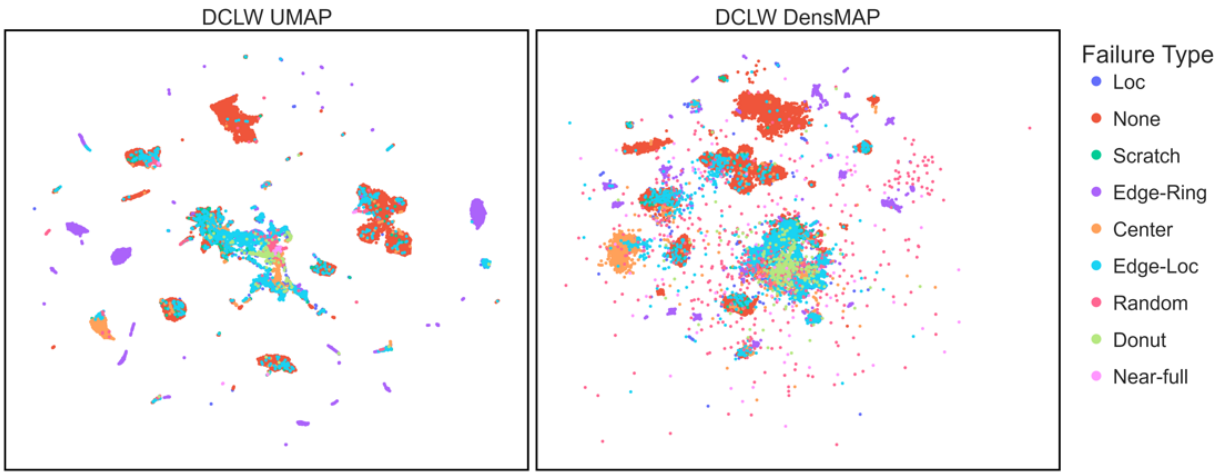
(p) SimMIM

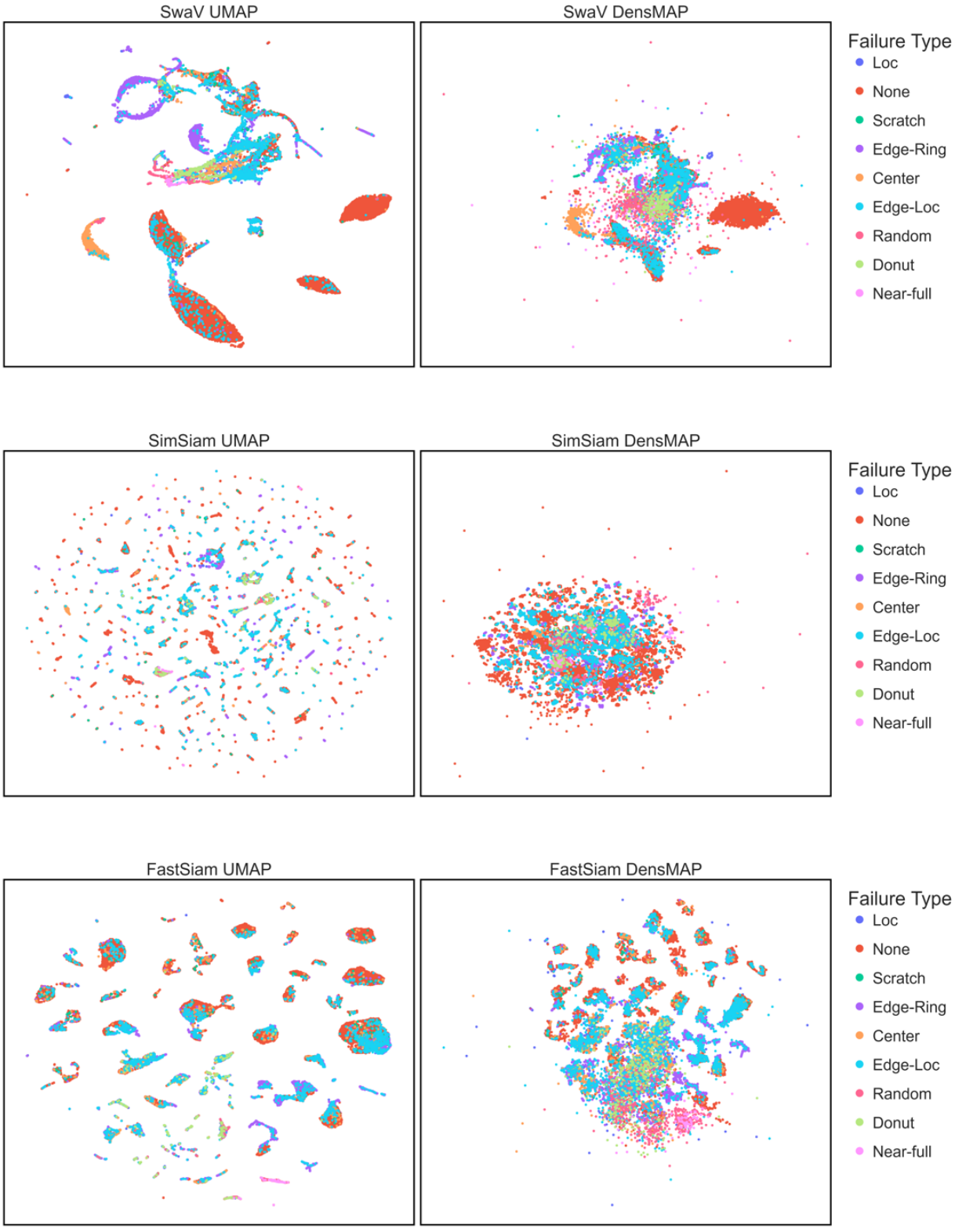
Figure 35: Confusion matrices of all k-NN benchmark models computed at the end of 150 epochs of training.

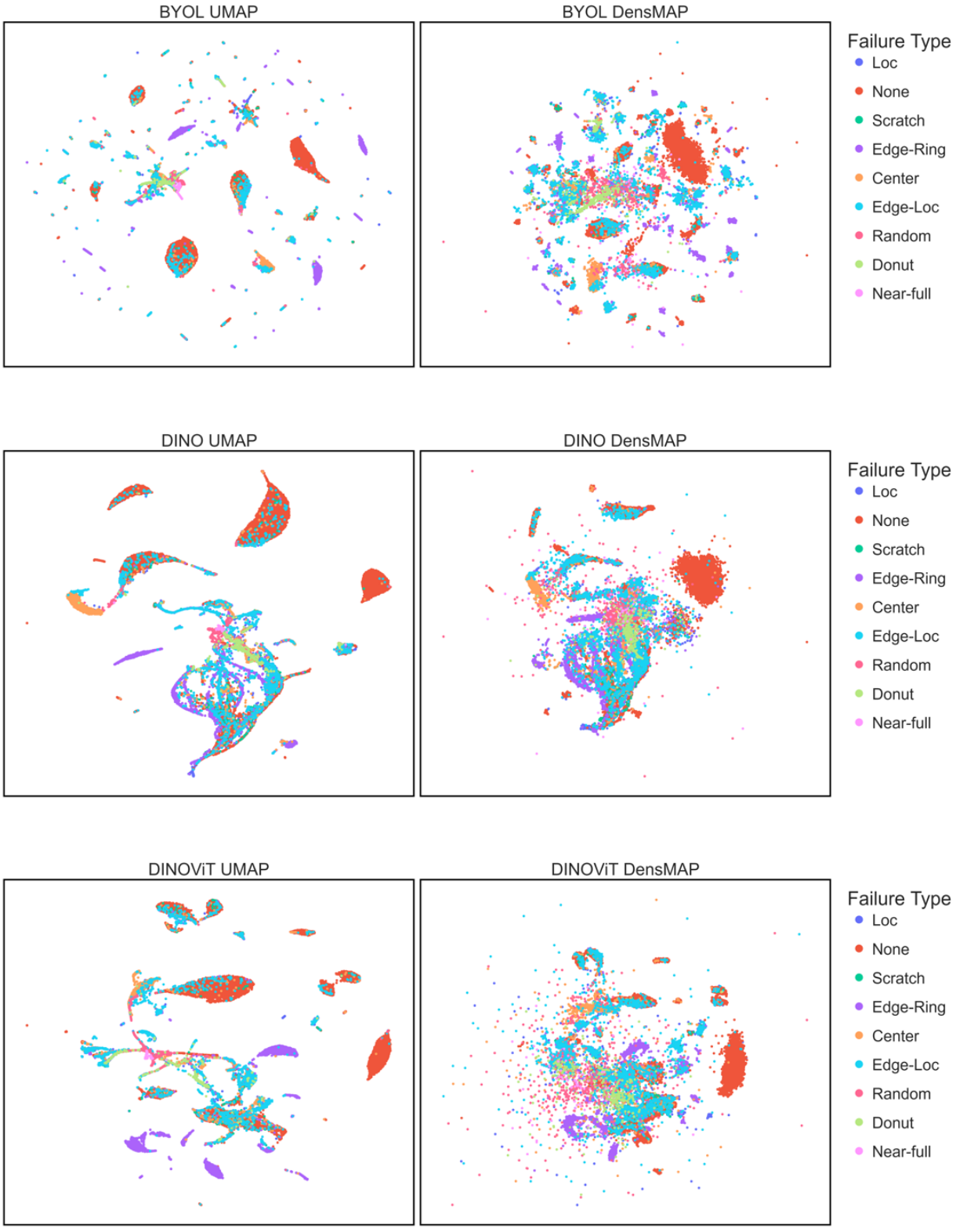
D Visualizations of Feature Spaces

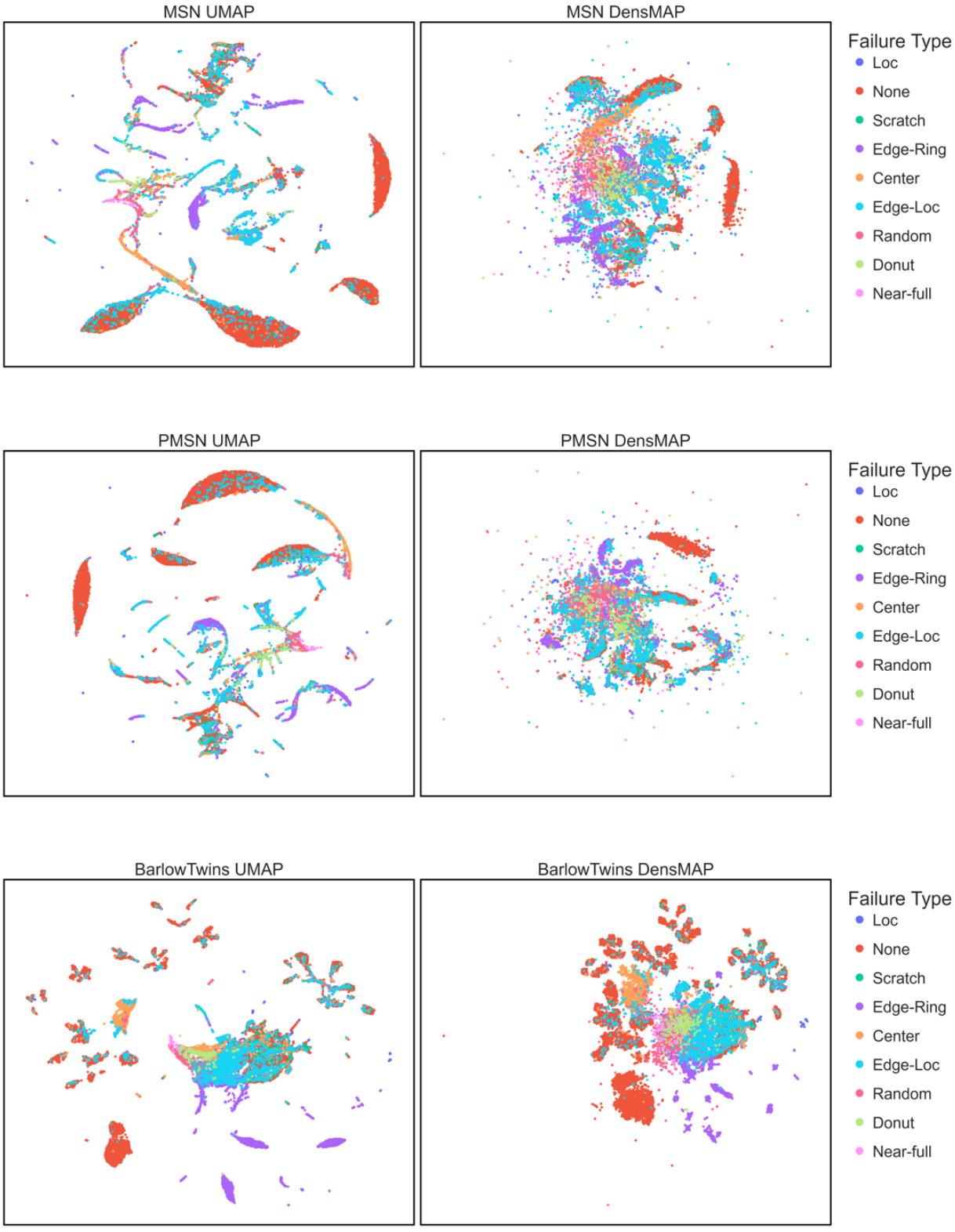







SwaV UMAP
SwaV DensMAP
SimSiam UMAP
SimSiam DensMAP
FastSiam UMAP
FastSiam DensMAP
Failure Type
Loc
None
Scratch
Edge-Ring
Center
Edge-Loc
Random
Donut
Near-full


BYOL UMAP
BYOL DensMAP
DINO UMAP
DINO DensMAP
DINOViT UMAP
DINOViT DensMAP
Failure Type
Loc
None
Scratch
Edge-Ring
Center
Edge-Loc
Random
Donut
Near-full


MSN UMAP
MSN DensMAP
PMSN UMAP
PMSN DensMAP
BarlowTwins UMAP
BarlowTwins DensMAP
Failure Type
Loc
None
Scratch
Edge-Ring
Center
Edge-Loc
Random
Donut
Near-full

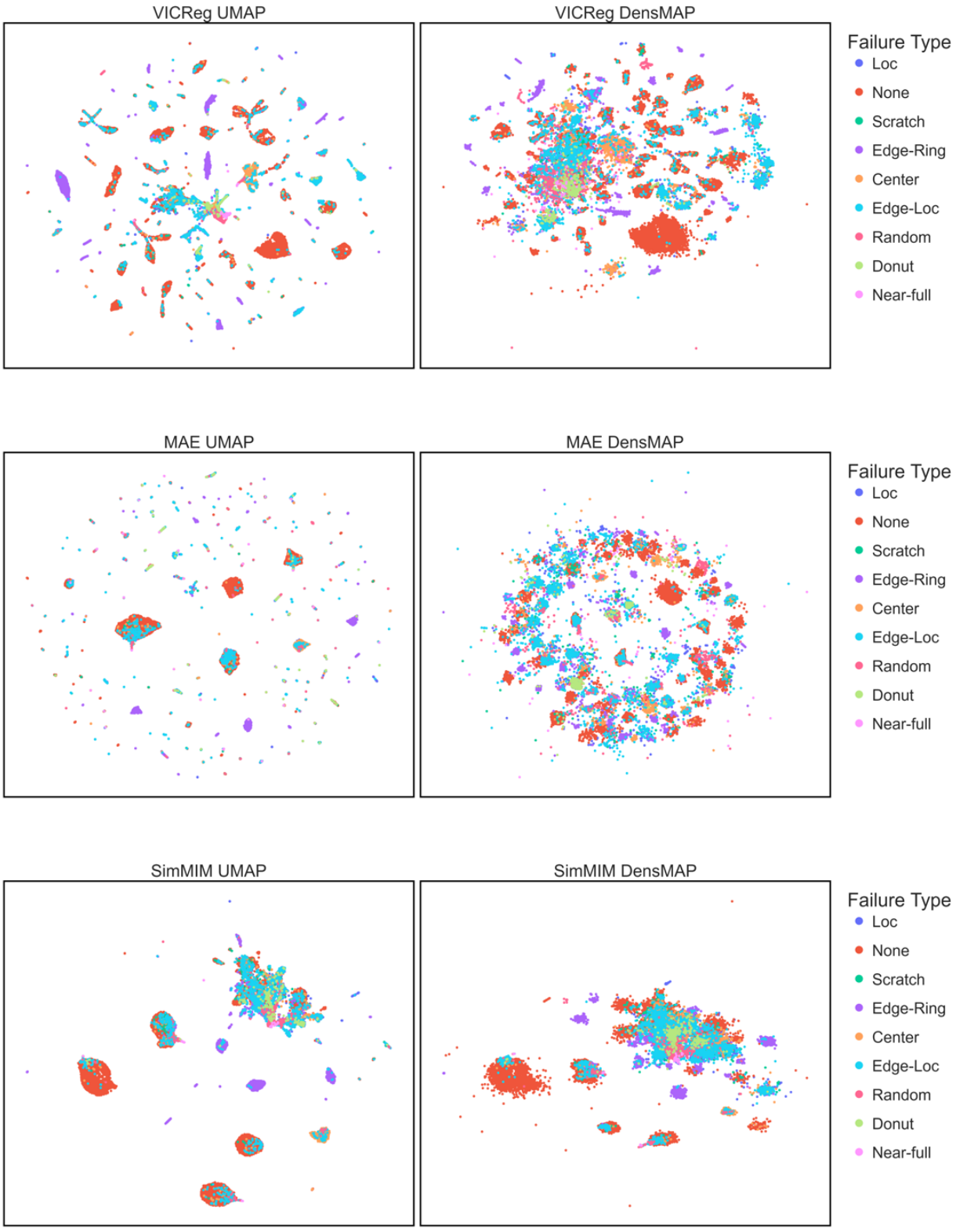


Figure 36: UMAP and DensMAP embeddings of the training dataset for all models. UMAP parameters are default (`n_neighbors=15, min_dist=0.1`). DensMAP reduction uses `dens_lambda=1`, otherwise all parameters are default. Qualitatively, UMAP may be easier to digest, as DensMAP essentially "expands" regions of low density.