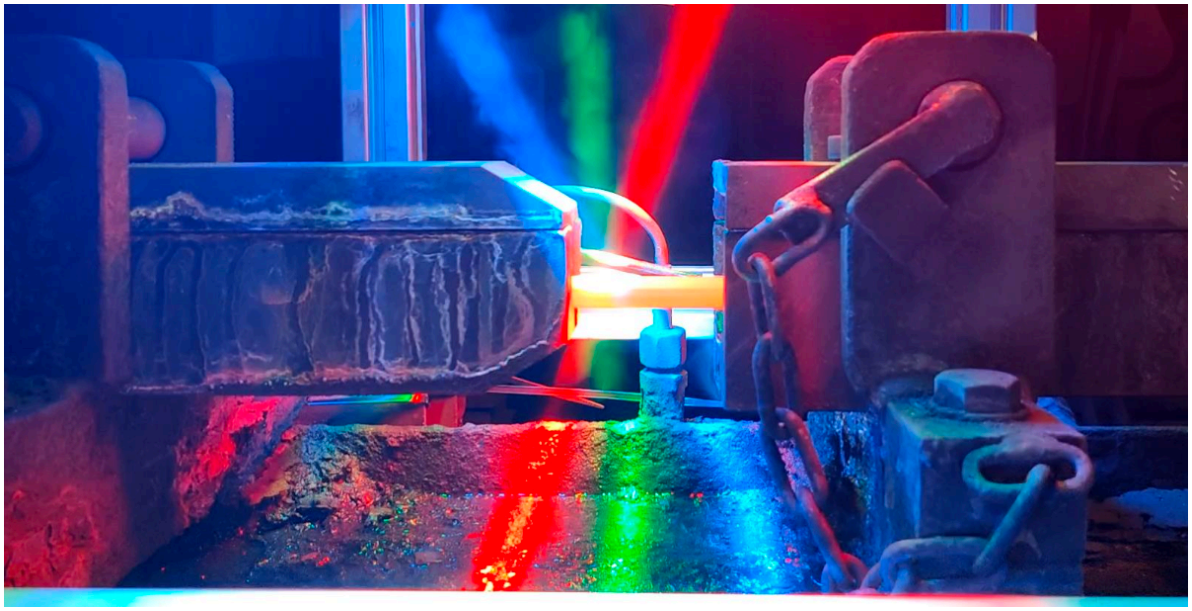


Self-Supervised Vision Transformers for Steel Surface Defect Detection

An Empirical Investigation of Fine-Tuning Strategies and Data Efficiency

Master's thesis in Data Science and AI

Nora Hemmingsson
Alexander Olsson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Master's thesis 2026

Self-Supervised Vision Transformers for Steel Surface Defect Detection

An Empirical Investigation of Fine-Tuning Strategies and Data Efficiency

Nora Hemmingsson
Alexander Olsson

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2026

Self-Supervised Vision Transformers for Steel Surface Defect Detection
An Empirical Investigation of Fine-Tuning Strategies and Data Efficiency
Nora Hemmingsson
Alexander Olsson

Supervisor: Selpi, Department of Computer Science and Engineering
Advisor: Peter Lundin, Swerim
Examiner: Dag Wedelin, Department of Computer Science and Engineering

Master's Thesis 2026
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Camera setup used in the data collection phase. Source: Swerim

Typeset in LaTeX
Gothenburg, Sweden 2026

Self-Supervised Vision Transformers for Steel Surface Defect Detection
An Empirical Investigation of Fine-Tuning Strategies and Data Efficiency
Nora Hemmingsson
Alexander Olsson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Industrial defect classification is a critical task in quality control, where accurate detection of surface defects is essential for ensuring product reliability. However, obtaining large amounts of labeled data is often costly and time-consuming, motivating the use of self-supervised learning (SSL) to leverage unlabeled data. This thesis investigates the effectiveness of SSL for defect classification using Vision Transformer-based methods, with a focus on Masked Autoencoders (MAE) and Distillation with No labels (DINO). The study evaluates the performance of these methods under different conditions, including fine-tuning vs linear probing, ImageNet initialization vs training from scratch and varying amounts of labeled data. A comprehensive experimental setup is used to assess both overall performance and label efficiency and results are compared to a supervised You Only Look Once (YOLO) baseline.

The results show that both MAE and DINO learn transferable representations that achieve high classification performance after fine-tuning. DINO consistently outperforms MAE, indicating that distillation-based approaches produce more discriminative features for this task. Fine-tuning significantly improves performance compared to linear probing, highlighting the importance of adapting the full model to the downstream task. Additionally, ImageNet initialization provides a strong advantage over training from scratch, demonstrating the importance of large-scale pretraining. Under limited labeled data condition during fine-tuning stage, both methods remain effective, achieving competitive performance even at low label fractions such as 1 % or 5%. However, performance improves steadily as more labeled data becomes available. Analysis of the results reveals that most misclassifications occur classifying non-defective samples in defect classes. However, the confusion between the defect classes is minimal which indicates that the key challenge is to avoid the false positives, i.e. identifying non-defective samples as defective.

Overall, the finding demonstrate that self-supervised learning is a viable and scalable approach for industrial defect classification, particularly in scenarios where labeled data is scarce. While fully supervised methods still achieve the highest performance when sufficient labeled data is available, SSL provides a strong alternative with reduce reliance on annotations.

Keywords: Self-supervised learning (SSL), industrial defect classification, computer vision, Vision Transformer, MAE, DINO, label efficiency, transfer learning.

Acknowledgements

We would like to express our sincere gratitude to all of those who have supported and helped us during this thesis project. We are especially grateful to our supervisor, Selpi, whose expertise and encouragement have been essential in shaping the direction and quality of this work. We would also like to thank our examiner, Dag Wedelin, for his valuable guidance and insightful feedback.

We would also like to extend our sincere thanks to our collaborators at Swerim. In particular, we are grateful to our supervisor, Peter Lundin, for sharing his extensive domain knowledge and providing valuable insights into the industrial context of this project. We would also like to thank Kevin Cheuk for his support with quantitative aspects of the work and for his guidance in refining our modeling approaches.

Lastly, we would like to thank the Department of Computer Science and Engineering at Chalmers University of Technology for providing the necessary computational resources for our research.

Nora Hemmingsson
Alexander Olsson
Gothenburg, 2026-05-12

Contents

List of Figures

List of Tables

1

Introduction

Industrial surface inspection plays a critical role in modern manufacturing, particularly in the production of metallic components where surface defects may compromise structural integrity, safety and product reliability. Even small irregularities such as cracks, scratches or material overfill can lead to mechanical failure or costly recalls if not detected in time. Damage to products increases production costs and results in substantial resource waste. It also causes companies to lose valuable economic opportunities that could be highly cost-effective if robust machine learning-based defect detection systems were implemented [1]. Automated visual inspection systems based on machine learning have therefore become increasingly important for ensuring consistent quality control in industrial environments.

1.1 Background

Traditional machine vision systems relies heavily on handcrafted features and rule-based approaches, which often lack robustness under varying lighting conditions, surface textures and defect types. With the rise of deep learning, convolutional neural networks (CNNs) have significantly improved visual inspection performance by automatically learning discriminative representations from data. However, many deep learning-based defect detection systems rely on supervised learning, requiring large amounts of annotated training data. In industrial settings, labeled data is often limited, expensive to obtain and highly imbalanced [2]. Defect categories may appear rarely and manual annotation requires domain expertise and substantial time investment. As highlighted in recent industrial anomaly detection studies [3], the imbalance between abundant normal samples and scarce anomalous examples makes supervised training impractical in many real-world settings. Similar challenges are emphasized in self-supervised defect detection research [4], where the dependency on labeled data is identified as a key limitation for scalable industrial deployment.

At the same time, industrial environments often generate large volumes of unlabeled image data during routine production. This creates an opportunity to leverage SSL, a method in which the models learn meaningful visual representations from unlabeled data by solving pretext tasks, such as image reconstruction or contrastive representation learning [5]. Instead of relying on human-provided annotations, SSL methods exploit intrinsic structure in the data itself. After pretraining on unlabeled images, the learning representations can be transferred to downstream tasks, such

as classification, which typically requires fewer labeled examples.

Recent advances in SSL have demonstrated that representations learned without manual labels can match or even surpass supervised pretraining. Multiple surveys, including Zhang et al. [6], Gui et al. [7], and Jaiswal et al. [8], highlight SSL as a compelling alternative to traditional supervised learning. Since the learning is derived directly from the input data, SSL is fundamentally more scalable and broadly applicable across domains such as computer vision, natural language processing, tabular data and scientific application. Methods based on masked image modeling, such as MAE [9] and teacher-student distillation frameworks such as DINO [10] have shown strong performance when combined with Vision Transformer (ViT) architecture. Transformer backbones provide enhanced global context modeling compared to traditional CNNs, which is beneficial for capturing complex defect patterns. Recent work on Transformer-based anomaly detection [11] demonstrates that combining reconstruction-based and classification-based objectives can significantly improve performance. Furthermore, improvements to MAE for industrial anomaly detection [3] show that architectural adaptations and masking strategies can enhance generalization even on small-scale industrial datasets.

This thesis is conducted in collaboration with Swerim, a Swedish research institute specializing in industrial materials and manufacturing processes. The dataset used in this study reflects real production conditions, where labeled defect data is limited and class distributions are highly imbalanced. These characteristics make the task representative of practical industrial inspection systems, where robust learning under limited supervision is essential for reliable quality control.

Prior exploratory work at Swerim [12] has investigated the feasibility of applying machine learning to automated steel surface defect detection. In that study, a CNN was trained to perform binary defect detection on images of steel products collected from industrial inspection systems. The work demonstrated that machine learning can support real-time defect detection in industrial environments. However, it also highlighted several practical challenges, such as limited labeled data, strong class imbalance between defect and non-defect samples and variations in imaging conditions across production lines. These limitations suggest that more advanced learning approaches may be required to fully leverage the large amount of unlabeled industrial data available in production environments, which thus motivates the exploration of SSL methods in this thesis.

1.2 Problem Formulation and Purpose

Despite the above mentioned advances in the area, several open questions remain. Although recent studies have demonstrated the potential of SSL for industrial anomaly detection, most of the work focuses on specific architectures or training strategies. For example, contrastive frameworks such as SimSiam [13] have been evaluated for steel surface defect detection, while MAE-based approaches have been adapted for industrial anomaly localization [3]. However, systematic comparisons across different SSL families under realistic industrial data constraints remain limited. Furthermore,

most empirical evaluations of SSL have been conducted on large-scale natural image datasets. There remains somewhat of a limited understanding of how different SSL families behave under industrial conditions characterized by texture-dominated images, subtle defect patterns and strong class imbalance.

This thesis investigates whether SSL can improve multi-class metallic surface defect classification compared to conventional supervised baselines. The study is conducted using an industrial dataset provided by Swerim, consisting of metallic surface images with five classes, including one dominant non-defect category and several minority defect types. The dataset exhibits substantial class imbalance and subtle inter-class variations, making it representative of realistic industrial inspection challenges.

More specifically, the thesis aims to address the following research questions:

1. Can self-supervised learning on unlabeled data improve defect classification performance compared to using standard supervised baselines?
2. How do generative and distillation-based SSL approaches compare in terms of performance under industrial specific conditions?
3. To what extent do self-supervised learning approaches remain effective for industrial defect classification when labeled data is scarce?

To answer these questions, the work follows a structured experimental pipeline consisting of data preprocessing, self-supervised pretraining, supervised evaluation and controlled ablation studies. Additional experiments investigate label efficiency by limiting the amount of labeled training data in the fine-tuning stage and thereby evaluating the practical value of SSL in low-annotation scenarios.

By comparing supervised and self-supervised approaches under identical architectural and training conditions, this thesis aims to clarify the role of SSL methods in industrial defect classification. The results contribute both practical insights for industrial deployment and a deeper understanding of how modern SSL frameworks behave in texture-dominated, domain-specific environments.

1.3 Project Scope

This thesis focuses on the application of SSL for image classification of metallic surface defects. The scope is limited to classification tasks and therefore object detection and segmentation are not considered. Furthermore, the study is restricted to the dataset provided by Swerim, which means that the findings are specific to this industrial setting and may not generalize directly to other domains. Also, the thesis does not include either deployment or integration into any of the Swerim customers' production systems.

The experimental study primarily investigates two self-supervised learning methods, one generative SSL method (MAE) and one non-contrastive SSL method (DINO), which represent two distinct families of SSL approaches. While other methods such as SimSiam, BYOL and SimCLR [6] are relevant within the broader SSL literature, they are not explored in this project. The selection of MAE and DINO was motivated

by their strong empirical performance in earlier literature and their complementary learning methods.

The scope is further constrained by the number of model configurations that can be evaluated. Only a limited set of backbone architectures, training strategies and data augmentation techniques are considered. In particular, SSL methods are known to be sensitive to augmentation choices and overly aggressive transformations may obscure subtle defect patterns. Thus, the augmentation strategies are carefully selected but not exhaustively explored. The dataset used in this project exhibits significant class imbalance which introduces additional challenges. Minority defect classes contain relatively few samples which can lead to higher variance in model performance and reduced reliability in evaluation metrics for these classes.

Finally, computational limitations impose some practical constraints on the experimental design. Some SSL methods require large batch sizes or long training schedules that are not feasible with the available GPU resources. Because of this, the project cannot perform extensive hyperparameter searches or evaluate a wide range of model variants.

2

Theoretical Background

This chapter provides the theoretical background necessary for understanding the methods and experimental design used in this thesis. It begins with an overview of supervised learning and transfer learning, which serve as important baselines in modern computer vision. The chapter then introduce SSL with a focus on its core principles and different methodological families. Finally, the fundamentals of Transformer-based architectures are presented, followed by descriptions of the specific SSL approaches, DINO and MAE, that are investigated in this thesis work.

2.1 Supervised Learning

Supervised learning is one of the dominant paradigms in modern computer vision and forms the foundation for many image classification systems. In supervised learning, a model is trained on labeled data pairs $\{(x_i, y_i)\}_{i=1}^{N}$, where x_i denotes an input image and $y_i \in \{1, \ldots, k\}$ represents its corresponding class label. The objective is to learn a function $f_\theta(x)$, parameterized by θ, that maps images to class probabilities [14].

Training is typically formulated as a minimization problem where the model parameters are optimized to minimize the average loss over the training dataset:

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{i=1}^{N} \ell(f_\theta(x_i), y_i), \tag{2.1}$$

where $\ell(\cdot)$ denotes a suitable loss function. For multi-class image classification, the most common choice is the cross entropy-loss:

$$\mathcal{L}_{CE} = -\sum_{c=1}^{k} y_c \log(p_c), \tag{2.2}$$

where p_c denotes the predicted probability for class c and y_c is the one-hot encoded ground truth label. Minimizing this loss encourages the model to assign high probability to the correct class while suppressing incorrect predictions [15].

2.2 Transfer Learning

Although supervised learning has achieved remarkable performance in large-scale benchmarks, it typically requires substantial amounts of labeled data to generalize

well. In industrial inspection scenarios, collecting large quantities of labeled defect images is expensive and time-consuming. Moreover, certain defect categories may be rare, resulting in limited supervision. To address data scarcity, transfer learning is commonly employed. In this setting, a model is first pretrained on a large-scale dataset such as ImageNet and subsequently fine-tuned on a smaller target dataset. The pretrained model provides generic visual representations, including edge detectors, texture filters and mid-level semantic features, which can accelerate convergence and improve performance on the downstream task [16].

Formally, transfer learning initializes the parameters θ with pretrained weights θ_{pre} instead of random initialization and then optimizes

$$\theta^* = \arg\min_{\theta} \mathcal{L}_{target}(\theta \mid \theta_{pre}), \tag{2.3}$$

on the target dataset.

While transfer learning has become standard practice in computer vision, its effectiveness depends on the similarity between the source and target domains.

2.3 Self-Supervised Learning (SSL)

SSL has become one of the most important developments in modern machine learning over the past decade. Its popularity is closely linked to the increasing availability of large-scale unlabeled datasets and the practical limitation of collecting high quality labeled data [6]. SSL offers a compelling alternative to traditional supervised learning by removing the requirement for manually annotated labels. Instead, the learning is derived directly from the input data which makes SSL fundamentally more scalable and broadly applicable across domains such as computer vision, natural language processing, tabular data and scientific application.

While supervised learning depends on explicit labels provided by humans, SSL takes advantage of what Zhang et al.[6] describes as intrinsic supervisory signals contained in the data itself. The underlying assumption is that data contains structural patterns, such as relationships and dependencies, that can be used in practice to learn meaningful representations even without human intervention. Because there are no labels in this data, the information from these structural patterns is what creates a signal from the data itself. This perspective align well with LeCun's [17] argument that predictive or self-supervised learning is the closest analog we have to how humans and animals learn. The argument suggests that humans and animals learn by constantly predicting or inferring missing information in their environment while being guided by internal goals rather than external labels.

SSL can therefore be found in between supervised and unsupervised learning. Like unsupervised learning, it makes use of unlabeled data. But like supervised learning, the training process is driven by clearly defined objectives to see how well the model matches the signal. These objectives are generated through pretext tasks, which are artificial or supporting tasks that are created by transforming the input data in a

systematic way. Solving these tasks forces the model to extract semantic information that later become useful for downstream applications. According to Gui et al. [7], the design of these pretext tasks are not random but must be constructed to encourage the model to capture properties that extend beyond the synthetic task.

2.3.1 Core Idea

The central idea of SSL is that the data contains sufficient internal structure so it can generate its own supervisory signal. Instead of relying on human provided labels, SSL creates training objectives by transforming the input data in systematic ways and asking the model to either predict, reconstruct or relate these transformed versions. By solving these artificially constructed tasks, the model learns representation that capture the essential semantic and structural information. This information can later be used for real downstream problems.

Across several surveys, this philosophy is consistently mentioned. Zhang et al. [6] describe SSL as a process where models "construct supervision from structured transformations of the input" while Gui et al. [7] and Jaiswal et al. [8] underline that the success from SSL comes from exploiting relationships, dependencies and mutual information that is naturally present in the data. Uelwer et al. [18] discuss where these relationships arise from in the data. The results of the study showed that, in visual domains, these relationships come from spatial coherence, object structure and appearance similarities.

2.3.2 Pretext Tasks in Early SSL

Early research in SSL often relied on pretext tasks, which are artificial tasks generated automatically from the data without manual labels. These are complementary tasks that are created directly from the input data without the need for manmade annotated labels. Instead of predicting an external target, the model is asked to solve a task generated automatically through transformations such as masking, cropping or contextual rearrangement. As described in several surveys [6], [7], [8], pretext tasks are the mechanism that allows SSL to extract supervisory information from raw, unlabeled data. The purpose of pretext tasks is not to solve a meaningful real-world task but to push the model to uncover the concealed structure of the data and produce internal representations that can later support the downstream applications.

Pretext tasks are closely tied to the idea of the self-supervised signal and the training objective that was mentioned earlier. The transformation applied to the input determines what the model must predict, reconstruct or compare and this is what then defines the self-supervised signal. The training objective then evaluates how well the model matches that signal, for example by penalizing reconstruction errors or encouraging two augmented views of the same instance to produce similar representations. In this way, the pretext task determines what information the model should learn from, while the objective determines how this learning is done. This relationship is discussed across SSL literature, for instance by Albelwi et al. [19]

and Gui et al. [7] who emphasize that effective pretext tasks must produce training signals that are rich enough to guide meaningful representation learning.

The design of these pretrained tasks has evolved significantly over the past decade. Although pretext tasks played a major role in early SSL research, they are now often viewed as one of several approaches within the broader SSL framework. More recent methods instead focus directly on learning invariant and informative representations through objectives that compare different views of the data or predict masked content. Studies such as Uelwer et al. [18] show that these modern formulations usually capture deeper semantic and contextual information compared to the earlier tasks. Despite the variety of methods, all pretext tasks share the same goal which is to expose the structural patterns within the data in a way that leads to robust and transferable representations once the pretrained model is used for downstream tasks.

2.3.3 Families of SSL Methods

SSL methods can be categorized into several major families depending on the design of their pretext tasks and learning objectives. According to a recent study by Zhang et al.[6], SSL approaches can be grouped into six primary categories: context-based methods, generative learning-based methods, contrastive learning-based methods, clustering-based methods, graph-based methods and hybrid methods.

Generative pretext tasks form SSL as a reconstruction or prediction problem. The core idea is to mask, corrupt or remove parts of the input and train the model to recover the missing information. These kind of methods draws strong inspiration from classical generative models. A prominent example is masked prediction, where portions of the input are hidden and the model is tasked with predicting the missing content. In natural language processing, this paradigm is exemplified by masked language modeling, while in vision it has led to masked image modeling approaches that operate on image patches or tokens. Autoencoding-based methods, including denoising autoencoders and variational autoencoders, also belong to this family, as they aim to reconstruct the original input from a compressed latent representation. Generative pretext tasks encourage models to capture global structure and fine-grained details of the data distribution. However, reconstruction objectives may overemphasize low-level information and pixel-wise accuracy which does not always correlate with improved downstream performance [20].

Contrastive pretext tasks define representation learning as a discrimination problem between similar (positive) and dissimilar (negative) samples. The central assumption is that different augmented views of the same data instance should yield similar representations, while representations of different instances should be distinct. In discrimination frameworks, positive pairs are generated through data augmentation, and contrastive losses such as InfoNCE [21] are used to maximize agreement between them. Contrastive learning is particularly effective at learning invariant and discriminative representations, making it well suited for classification tasks [6]. Even though it includes many influential SSL methods, it nevertheless introduces challenges related to negative sample selection, batch size dependence and potential representation collapse [20].

Non-contrastive methods on the other hand define representation learning by enforcing consistency between different augmented views of the same data instance without relying on negative samples. Instead of contrasting positive and negative pairs, these approaches encourage the model to produce similar representations for different transformations of the same input. This is typically achieved through architectural asymmetry, momentum encoders or some additional normalization mechanisms. This SSL family has been shown to rely solely on positive sample pairs, while still maintaining effective representation learning through carefully designed training strategies [22]. It offers several advantages compared to contrastive approaches. By removing the dependency on negative samples, these methods avoid challenges related to large batch sizes, memory banks and the careful selection of negative examples. This makes them more computationally efficient and easier to scale. At the same time, they have been shown to learn semantic representations that transfer effectively to downstream tasks such as classification and detection. However, a key challenge is preventing representation collapse meaning that the model outputs identical representations for all inputs.

2.3.4 Downstream Tasks in SSL

SSL methods are typically evaluated by transferring the learned representations to downstream tasks. After pretraining on unlabeled data, the learned encoder is applied to a supervised task such as image classification, object detection or semantic segmentation. The performance on the downstream task serves as an indicator of the quality and generalization capability of the learned representations.

Two common evaluation protocols that are widely used in the SSL literature are linear probing and fine-tuning [23] [24]. In linear probing, the parameters of the pretrained backbone network are frozen and only a linear classification layer is trained on top of the learned representations. This protocol evaluates the quality of the representations themselves, as the backbone features must already be linearly separable for the downstream classes.

In contrast, fine-tuning allows the entire network, including the pretrained backbone, to be updated during training on the downstream task [25]. This protocol evaluates the adaptability of the pretrained representations and often leads to higher performance when sufficient labeled data is available. As mentioned above, both evaluation strategies are commonly used in self-supervised learning research to provide complementary insights into representation quality and transferability.

2.4 Transformers and Vision Transformers

While SSL defines the training objectives used to learn representations from unlabeled data, the structure and inductive biases of the underlying model architecture play a crucial role in determining the quality of the learned representations. In recent years, Transformer-based architectures have emerged as an alternative to CNNs for visual representation learning. Unlike RNNs or CNNs, Transformers rely entirely on attention mechanisms to model dependencies between input elements which enables

flexible and scalable representation learning across long ranges [26]. Their success in language modeling has motivated their adaptation to computer vision, leading to the development of Vision Transformers (ViTs) [27].

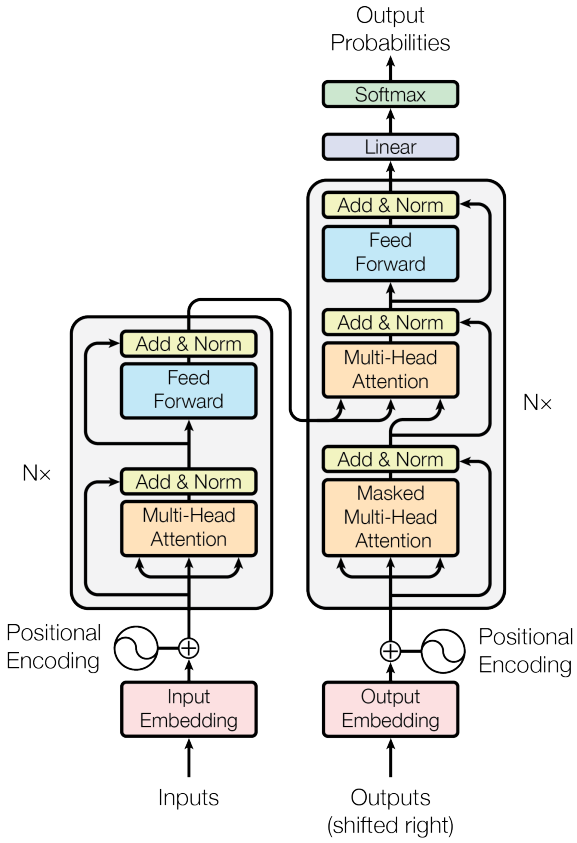


Figure 2.1: Overview of the Transformer encoder architecture based on self-attention [26].

The core component of a Transformer is the self-attention mechanism which allows each element of an input sequence to attend to all other elements. Given an input sequence represented as a set of feature vectors, self-attention computes context-aware representations by modeling pairwise interactions between elements. Formally, given input embeddings projected into queries Q, keys K and values V, self-attention is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V, \tag{2.4}$$

where d denotes the dimensionality of the key vectors. This formulation enables the model to assign different importance weights to different input elements when computing representations [28].

Transformers are composed of stacked layers combining self-attention with position-wise feed-forward networks, residual connections and layer normalization. In practice, multi-head attention is employed which allows the model to attend to information from multiple representation subspaces in parallel. Each attention head

captures different types of relationships and their outputs are concatenated and linearly transformed. A defining characteristic of Transformers is the absence of strong locality assumptions. Unlike CNNs, Transformers do not encode spatial priors such as translation equivariance or local connectivity. They instead rely on the learning objective and data to infer structure [26].

ViTs adapt the Transformer architecture to image data by representing images as sequences of patch embeddings. The input image is divided into a fixed grid of non-overlapping patches where each of them is flattened and linearly projected into a vector embedding. These embeddings form the input token sequence to the Transformer encoder. To preserve spatial information, learnable positional embeddings are added to the patch embeddings. In addition, a learnable class token (CLS) is typically added to the sequence and its output representation is used as a global image descriptor for downstream tasks. The Transformer encoder processes this sequence through successive layers of self-attention and feed-forward networks. Since self-attention operates globally, each patch representation can directly attend to all other patches which enables the model to capture long-range dependencies.

Compared to CNNs, ViTs exhibit weaker built-in inductive biases, which refers to the assumptions a learning algorithm makes about the structure of the data. Convolutional networks encode strong assumptions about locality, translation invariance and hierarchal feature composition through their architecture. ViTs, by contrast, make fewer structural assumptions and thus rely more heavily on data and training objectives to learn meaningful representations. As a result, ViTs typically require large-scale pretraining or carefully designed objectives to achieve competitive performance [29]. SSL has emerged as a particularly effective paradigm in this context, as it allows ViTs to leverage large amounts of unlabeled data while learning invariances and semantic structure through carefully designed pretext tasks [30].

The interaction between Transformers and SSL objectives has revealed properties that do not emerge as clearly in supervised settings. By enforcing consistency across multiple augmented views of the same image, self-supervised methods encourage ViTs to learn representations that encode both global context and object-level structure. This synergy has made them a natural backbone for modern SSL frameworks. In particular, methods based on self-distillation benefit from the expressive capacity and global receptive field of Transformers, resulting in representations that transfer effectively to a wide range of downstream tasks.

2.5 Distillation with No Labels (DINO)

Building on the Transformer architecture, recent SSL methods have demonstrated particularly strong performance when ViTs are used as backbone models. While earlier families of pretext tasks rely on explicit reconstruction objectives or instance-level contrastive comparisons, recent advances in SSL have explored alternative mechanisms for enforcing representation consistency across views. Classical contrastive approaches, such as SimCLR [23] and MoCo [31], formulate representation learning using positive and negative pairs. This often requires large batch sizes or

memory banks to ensure sufficient negative samples. More recent methods challenge this requirement by demonstrating that meaningful representations can be learned without explicit negatives, as shown in self-distillation and asymmetric architectures such as BYOL [32].

The DINO framework [10] consists of two networks sharing the same architecture, a student network which is optimized by gradient descent and a teacher network whose parameters are updated as an exponential moving average of the student's parameters. Given an input image, multiple augmented views are generated using a combination of global and local crops, color jittering, blurring and other stochastic transformations. More formally, let g_{θ_s} and g_{θ_t} denote the student and teacher networks, respectively. Each network consists of a backbone feature extractor followed by a projection head and outputs a K-dimensional vector. Given an unlabeled image x, a set of augmented views is generated using a stochastic augmentation pipeline:

$$\mathcal{V}(x) = \{x^{(1)}, x^{(2)}, \ldots, x^{(V)}\}. \tag{2.5}$$

In the standard configuration of DINO, shown in Fig. 2.2, this set contains two global views and several local views generated by a multi-crop strategy [33]. All views are processed by the student network, while only the global views are processed by the teacher network. This design encourages local-to-global consistency in the learned representation.

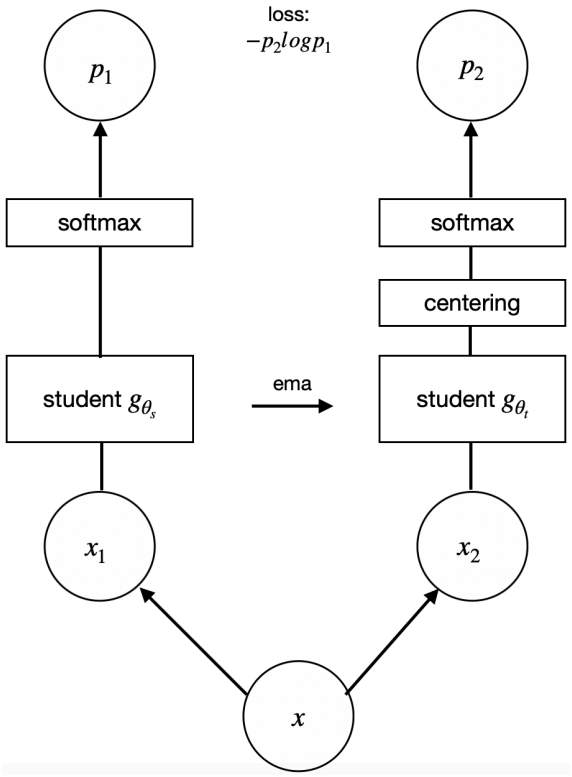


Figure 2.2: Illustration of DINO using a single pair of views (x_1, x_2) for simplicity purposes.

Both networks produce non-normalized outputs which are then converted into prob-

ability distributions using a softmax function with temperature scaling. The student output distribution is defined as follows.

$$P_s(x)(i) = \frac{\exp\left(g_{\theta_s}(x)(i)/\tau_s\right)}{\sum_{k=1}^{K} \exp\left(g_{\theta_s}(x)(k)/\tau_s\right)}, \tag{2.6}$$

where $\tau_s > 0$ is the student temperature parameter.

For the teacher network, a centering operation is applied prior to normalization to prevent representation collapse:

$$P_t(x)(i) = \frac{\exp\left((g_{\theta_t}(x)(i) - c(i))/\tau_t\right)}{\sum_{k=1}^{K} \exp\left((g_{\theta_t}(x)(k) - c(k))/\tau_t\right)}, \tag{2.7}$$

where τ_t is the teacher temperature and $c \in \mathbb{R}^K$ is a running center vector.

The center is updated using an exponential moving average over the batch-wise mean of the teacher outputs:

$$c \leftarrow mc + (1-m)\frac{1}{B}\sum_{i=1}^{B} g_{\theta_t}(x_i), \tag{2.8}$$

where $m \in [0, 1)$ is a momentum coefficient and B is the batch size.

DINO minimizes the cross-entropy between the teacher and student output distributions computed on different views of the same image. For a teacher view x and a student view x', the loss is defined as

$$\mathcal{L}_{\text{DINO}} = H\Big(P_t(x), P_s(x')\Big) = -\sum_{i=1}^{K} P_t(x)(i) \log P_s(x')(i). \tag{2.9}$$

The full objective aggregates this loss across all valid pairs of teacher and student views:

$$\min_{\theta_s} \sum_{x \in \{x_1^g, x_2^g\}} \sum_{\substack{x' \in \mathcal{V}(x) \\ x' \neq x}} H\Big(P_t(x), P_s(x')\Big). \tag{2.10}$$

Gradients are propagated only through the student network, while the teacher parameters are updated using an exponential moving average of the student parameters.

$$\theta_t \leftarrow \lambda\theta_t + (1-\lambda)\theta_s, \tag{2.11}$$

where λ follows a cosine schedule and is typically close to one throughout training.

The term representational collapse used above refers to a degenerate solution in SSL where the model maps all inputs to near-identical representations which results in a loss of discriminative information. Collapse can manifest itself in two distinct forms. In uniform collapse, the network produces a uniform output distribution for all inputs which results in high-entropy but uninformative representations. In contrast, dominant-dimension collapse occurs when the network assigns nearly all probability mass to a single output dimension for every input, yielding low-entropy but equally

degenerate representations. Unlike many SSL methods, DINO does not rely on negative samples, batch normalization [34] or predictor networks [32] to prevent collapse. Instead, collapse is avoided through the interaction of centering and output sharpening. Centering prevents domination by a single output dimension while a low teacher temperature τ_t sharpens the distribution and discourages convergence to a uniform solution. The combination of these two mechanisms is sufficient to ensure stable training [10].

2.6 Masked Autoencoders (MAE)

Masked Autoencoders (MAE) are a class of SSL methods based on masked image modeling, where representation learning is formulated as a reconstruction problem [9]. In contrast to non-contrastive or distillation-based approaches such as DINO, MAE belongs to the family of generative pretext tasks and learns representations by reconstructing missing portions of the input image from partially observed signal. The MAE framework was introduced to address two key challenges in applying ViTs to SSL. The first challenge is the high computational cost of dense image reconstructions and the second challenge is the limited inductive bias of Transformer architecture. By combining aggressive masking with an asymmetric encoder-decoder design, MAE achieves efficient and scalable self-supervised pretraining of ViTs.

The central idea of MAE is to randomly remove a large fraction of image patches and train a model to reconstruct the missing content [9]. Given an input image represented as a sequence of patch embeddings, only a small subset of visible patches is processed by the encoder while the decoder is tasked with reconstructing the full image from this partial representation, as seen in Figure 2.3. Unlike non-contrastive methods that rely on comparing multiple augmented views of the same image, MAE constructs its supervisory signal by explicitly removing information and requiring the model to infer the missing structure. This formulation aligns closely with the predictive learning perspective proposed by LeCun et al. [17], where learning is driven by the ability to model and predict unseen parts of sensory input. A key empirical observation from the original MAE paper [9] is that ViTs tolerate extremely high masking rations (up to 75%) without performance degradation. This property is particularly important for computational efficiency and distinguish MAE from earlier masked image modeling approaches that typically relied on lower masking ratios.

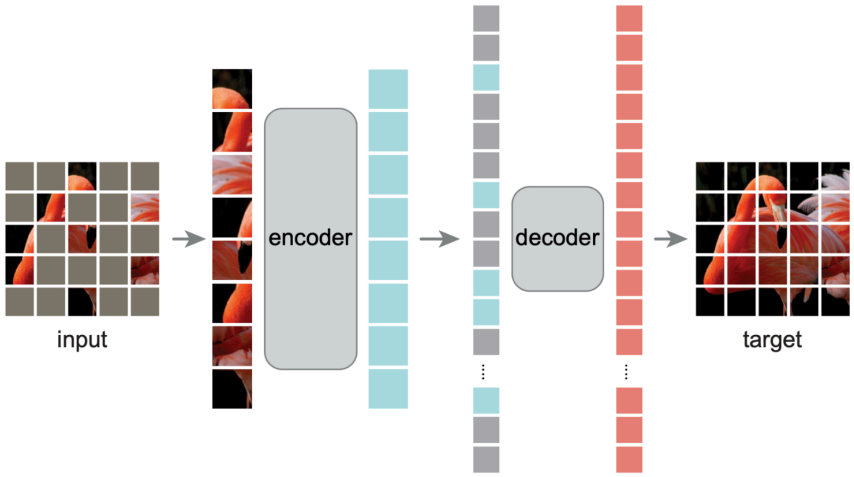


Figure 2.3: Illustration the general MAE architecture. During pre-training, a large random subset of image patches is masked out and the encoder is then applied to the small subset of visible patches [9].

The MAE architecture consists of two components: an encoder and a decoder [9]. The encoder is a standard ViT that processes only the visible patches which are the unmasked ones. Since a large fraction of patches is removed prior to encoding, the computational cost of the encoder is significantly reduced. The decoder is a lightweight Transformer that receives the encoded visible patches along with learnable mask tokens which represents the missing patches. Positional embeddings are added to preserve spatial information. The decoder outputs reconstructed pixel values for all patches and training is performed by minimizing the mean squared error between the reconstructed and original pixel values of the masked patches only. It is important to note that the decoder is discarded after pretraining and only the encoder is retained for downstream tasks. This design choice ensures that the learned representations stays in the encoder which is later fine-tuned or evaluated using linear probing. Formally, let x denote an input image and M the set of masked patch indices. The MAE objective minimizes the the reconstruction loss where $\hat{x}_i$ denotes the reconstructed patch and x_i the original patch, see Equation 2.12.

$$\mathcal{L}_{\text{MAE}} = \frac{1}{|M|} \sum_{i \in M} \|\hat{x}_i - x_i\|^2 , \tag{2.12}$$

Masking plays a central role in MAE and strongly influences the learned representations [9]. The original MAE formulation uses random masking, where the image patches are uniformly sampled without spatial bias. Given a predefined masking ratio, a fixed number of patch indices is randomly selected and removed before being processed by the encoder. This results in visible patches that are spread out across the image, forcing the model to rely on global context rather than using local patterns. Random masking has two important properties [9]. The first one is that because masked patches are distributed across the entire image, the encoder cannot rely on simple local context to reconstruct the missing regions. The second one is that when high masking ratios are used, the model must integrate information from

distant visible patches. This encourages learning representations that understand the whole picture and capture meaningful patterns, not just small or isolated details.

Beyond random masking, alternative masking strategies can be used to introduce different inductive biases into the reconstruction task. One variant is block-wise masking, where rectangular regions of patches are removed. Instead of independently sampling patches, rectangular blocks are randomly positioned on the patch grid until the desired masking ratio is reached. By removing an organized region, block masking encourages the model to learn larger structural patterns rather than relying alone on scattered context.

Another alternative is grid-wise masking, where the patches are removed according to regular lattice pattern [9]. In this case, masked patches are distributed periodically across the image grid often with a small random offset to avoid a fixed alignment. Grid masking introduces structured sparsity and ensures uniform spatial coverage of missing information. Compared to block masking, which creates specific blocked regions, grid masking distributes masked patches more evenly across the image. This forces the model to reconstruct missing information throughout the entire spatial domain rather than focusing on a single region.

Although the original MAE paper [9] primarily evaluates random masking and demonstrate its effectiveness, structured masking strategies may be particularly relevant in industrial defect detection context. In many industrial images, defects occur in specific regions rather than as randomly distributed defects. As a result, block-wise masking may better reflect the structural characteristics of surface defects and localized damage. Grid-wise masking, on the other hand, removes information in a regular grid pattern. This creates an even and controlled way of hiding parts of the data, which can help study how sensitive reconstruction-based learning is to removing structured information.

Important to note is that regardless of the masking strategy, the overall masking ratio is achieved to ensure that exactly a fixed portion of patches is removed. For block-wise or grid-wise masking, an initial spatial pattern is generated and adjusted to match the predefined masking ratio. This guarantees a fair comparison between masking types and as a result, the differences in performance can be be attributed to the characteristics of the masking strategy rather than the differences in the amount of visible input.

In addition to masking, data augmentation plays a role in shaping the learning dynamic of self-supervised methods. Interestingly, MAE differs from contrastive approaches in this regard [9]. Contrastive learning relies heavily on strong augmentations to prevent representation collapse. In contrast, the MAE objective itself provides a sufficiently challenging learning signal through reconstruction. The original MAE study reports that cropping only augmentations (both fixed size and random size) yield strong performance while adding color jittering degrades the results. Moreover, MAE remains functional even when minimal augmentation is applied beyond center cropping. This weak dependence on augmentation highlights a fundamental difference between generative masked modeling and contrastive learning [9]. In MAE, the difficulty of the task is controlled primarily by the masking ratio

rather than by view diversity. This property could be particularly advantageous in industrial defect detection, where aggressive augmentations may disrupt subtle texture patterns or introduce unrealistic transformations.

Lastly, an important consideration in the MAE framework is the choice of initialization. In many computer vision pipelines, models are initialized using ImageNet-pretrained weights [9]. Although this strategy is often effective, several studies question its suitability when the the target domain differs significantly from natural imagery. Raghu et al. [29] show that pretraining on ImageNet can lead to less effective feature reuse when the model is transferred to very different domains. In industrial inspection tasks, this mismatch may cause the model to focus more on high-level object features instead of detailed texture patterns. Recent industrial MAE-based approaches further suggest that pretraining from scratch on domain-specific unlabeled data may mitigate such domain mismatch effects [35].

3

Dataset and Data Preprocessing

This chapter describes the dataset and the preprocessing steps used in the thesis. It begins with an overview of the characteristics and class distribution of the dataset provided by Swerim. The chapter then continues to outline the data splitting strategy employed to ensure fair and reproducible evaluation. Finally, the preprocessing pipeline applied prior to the training is presented.

3.1 Swerim Dataset

The dataset provided by Swerim consists of metallic surface images collected during industrial measurement campaigns at steel manufacturing facilities. The dataset contains a total of 28 932 labeled images of size 512×512 pixels and contains five classes: *Non-defect*, *Cracks*, *Scrap mark*, *Overfill* and *Scrap mark II*. The class distribution is highly imbalanced, with non-defect images dominating the dataset and certain defect categories appearing only in small quantities. The class distribution is shown in Table 3.1. Example images from each class are illustrated in Figure 3.2.

Table 3.1: Labeled dataset distribution across classes.

Non-defect	**Cracks**	**Scrap mark**	**Overfill**	**Scrap mark II**
26258	228	692	950	799

The images were acquired using an industrial line-scan camera system, illustrated in Figure 3.1. Unlike conventional area cameras, a line-camera captures one spatial line at a time. The camera used in this setup has a resolution of 1×4096 pixels which means that each captured line consists of 4096 pixels in width and a single pixel in height. Two-dimensional images are formed by continuously sampling lines as the material moves past the camera.

The line acquisition frequency is 67 kHz, corresponding to approximately 14.77 μs between consecutive lines. The imaged objects move at speeds typically between 3-8 m/s in the studied cases, with other measurement campaigns conducted at speeds up to 30 m/s. Because the material is in production, the surface temperatures range approximately between 800°C and 1100°C.

The dataset thus reflects realistic industrial operating conditions, including high material temperatures, high motion speeds and measurement variability. Images

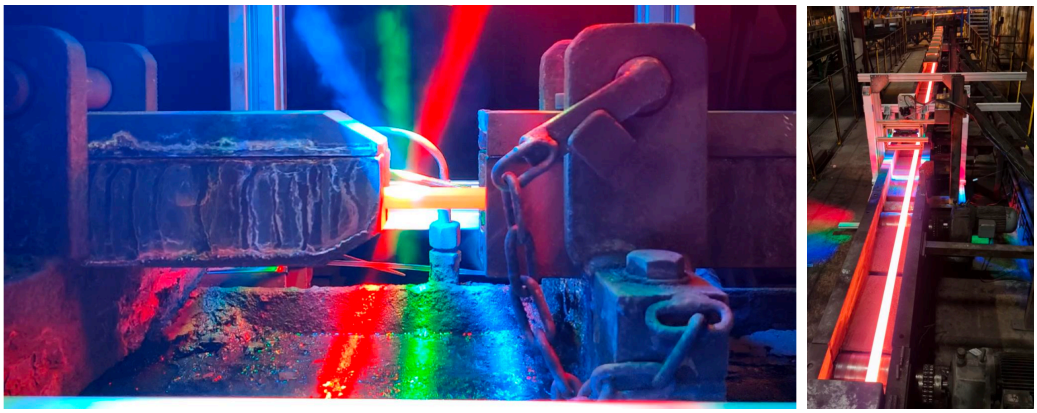


Figure 3.1: Data collection using line-scan camera system combined with LED-lights

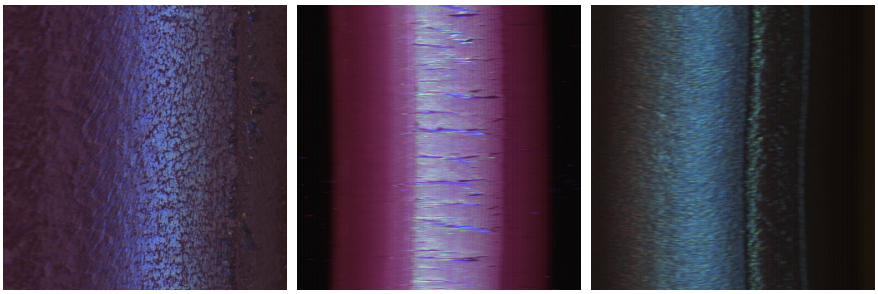
were collected on different material types, including wire, rod and strip products. All images were acquired by Swerim during industrial measurement campaigns and subsequently labeled according to defect type. This acquisition setup implies that spatial resolution along the motion direction is determined by the line frequency and material speed while resolution vertical to motion is determined by the 4096-pixel line sensor. The final 512×512 images used in this thesis are derived from these line-scan measurements.

3.2 Data Splitting

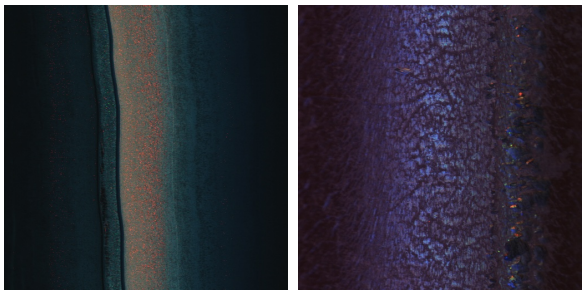
Although there was an initial train/validation/test split provided within the dataset, a new dataset split was created to ensure consistent experimental conditions and balanced class representation. More precisely, the dataset was divided into training (70%), validation (15%) and test(15%) subsets using stratified sampling based on class labels. Stratification was applied to preserve class distributions across all splits. Without stratification, minority defect classes risk being underrepresented from validation and test sets which would lead to unstable training and unreliable performance.

The use of stratified train/validation/test splits is standard practice SSL evaluation, especially with imbalanced datasets, and is recommended to ensure that performance metrics reflect model behavior across all classes. Maintaining representative class distributions across splits is essential for meaningful comparison between supervised baselines and downstream evaluations of self-supervised representations, which have been emphasized in prior work on SSL and industrial defect classification ([4], [23], [32]).

A sufficiently large training set is required to learn robust representations, while a separate validation set is used for model selection and hyperparameter tuning and an independent test set is reserved for final performance evaluation. Split ratios in the range of approximately 60-80 % for training, with the remaining data divided between validation and test, are widely used in supervised and SSL studies and



(a) Non-defect (b) Cracks (c) Scrap mark



(d) Overfill (e) Scrap mark II

Figure 3.2: Example images of the non-defect class and all different defect classes

form the standard protocol for downstream evaluation of learned representations ([14], [23]).

A fixed random seed was used when generating the data splits to ensure reproducibility. The test set was held constant and not used during model development or hyperparameter tuning, in order to avoid information leakage and preserving an unbiased evaluation.

Dataset splits were defined using CSV index files containing image paths and labels rather than relying on directory structure as it was set up before. This allows the same split to be reused across different training setups, including supervised learning and downstream evaluation of self-supervised representations, which is standard practice in recent SSL work ([17], [23]).

3.3 Data Preprocessing

To ensure consistent and reproducible experiments across both supervised and self-supervised learning settings, a standardized data preprocessing pipeline was applied to all images prior to training. The choices for preprocessing were intentionally kept minimal in order to avoid introducing domain-specific assumptions and also to have a fair comparison between models trained from scratch, ImageNet-pretrained models and self-supervised pretrained models. The same preprocessing steps were applied across all experiments, with the exception to data augmentations that are specific to self-supervised learning methods and are therefore introduced separately.

All images were converted to three-channel RGB format to ensure compatibility with

the selected backbone architecture. Although the dataset mainly consists of RGB images, this conversion step was done as a precautionary measure and guarantees a consistent input format in case any single-channel images are encountered. ResNet architectures are designed and evaluated using three-channel RGB images and their ImageNet-pretrained weights assume RGB input [36]. ViT models operate in a similar way on fixed sized patches extracted from RGB images, and pretrained ViT backbones require three-channel input [27]. Using a consistent color space across the dataset also simplifies the data pipeline and reduces unnecessary input variations [14].

All images were also resized to a fixed resolution of 224 x 224 pixels. This resolution is the standard input size for both ResNet and Vision Transformer architectures and allows direct use of ImageNet-pretrained models. ResNet models are evaluated using images resized to 224 x 224 pixels, which has become the standard resolution for convolutional neural networks trained on ImageNet [36]. ViT-B/16 is also designed and pretrained using 224 x 224 pixel inputs [27]. Using a fixed and widely adopted input resolution ensures architectural compatibility and ensures fair comparison between supervised baselines and self-supervised models. In addition, this resolution provides a reasonable trade-off between computational cost and representational capacity.

Furthermore, the images were normalized using the mean and standard deviation of the ImageNet dataset. This normalization is required for the correct use of ImageNet-pretrained weights and is standard practice in computer vision research. ResNet models pretrained on ImageNet assumes that the inputs are normalized using ImageNet statistics [36]. Applying the same normalization across all experiments ensures meaningful comparison between the models trained from scratch, ImageNet-pretrained models and further self-supervised pretrained models. This approach is also followed in SSL frameworks such as SimCLR [23] and BYOL [32], where ImageNet normalization is consistently applied during downstream evaluation.

Data quality has a direct impact on training stability and model performance in deep learning system [14]. In the context of industrial defect detection, Hu et al. [4] emphasize the importance of careful data inspection due to the presence of noise in industrial datasets. Excluding a small number of invalid samples is therefore preferable to risking inconsistent model behavior. Unreadable or corrupted images in our dataset were therefore checked and removed prior to training. The results of our data-cleaning showed however that no images were damaged and therefore no images were removed.

Class labels were preserved exactly as provided in the original dataset. No relabeling, merging or manual modification of the classes was performed. This decision maintains industrial relevance and enables defect-wise performance evaluation. Industrial defect datasets are often highly imbalanced and contain subtle visual differences between defect categories [4]. Rather then addressing this imbalance through label manipulation, class imbalance was handled through stratified data splits and loss weighting. Using minimal and standardized preprocessing has also been shown to be important for reliable evaluation of SSL methods [6].

Lastly, no histogram equalization, denoising, contrast enhancement or manual defect cropping was applied to the images. Such preprocessing operations could introduce domain-specific assumption. SSL methods are designed to learn representations directly from raw sensory data and excessive preprocessing can negatively affect the representation quality. Minimizing manual preprocessing aligns with the principles of representation learning outlined by LeCun et al. [17] and prior work has also shown that overly strong or domain-specific preprocessing choices can be harmful to SSL performance [6].

4

Methodological Framework

In this chapter we walk through the methodological framework used in the thesis. It outlines the model architecture, training strategies and experimental design employed to evaluate SSL for this specific industrial setting. The chapter begins with a description of the selected backbone architectures, followed by details on the downstream classification setup, handling of the class imbalance and training protocols. Finally, the evaluation metrics and experimental procedures are presented.

4.1 Choice of Backbone Architectures

The backbone architecture plays a central role in determining the representational capacity of the model and its ability to capture visual patterns. For this thesis, ViT-B/16 was selected as the primary backbone architecture for both supervised and self-supervised experiments.

Industrial surface defect classification differs from natural image object recognition in several important aspects. Metallic surface images are characterized by subtle texture variations, small structural irregularities and distributed low-contrast defects rather than distinct object-level semantics. Consequently, capturing both fine-grained local details and long-range contextual dependencies was deemed essential. ViTs model images as sequences of non-overlapping patches and apply global self-attention to model relationships between all patches in the image [27]. Unlike CNNs, which rely on local receptive fields and strong inductive biases such as translation equivariance, ViTs directly capture global interactions. This global modeling capability were though to be advantageous for detecting distributed or low-saliency defect patterns that extend across larger spatial regions.

Furthermore, recent literature on industrial defect detection increasingly adopts Transformer-based architectures and reports strong performance improvements over CNN-based counterparts. Chen et al. [37] propose a MAE-based self-supervised anomaly detection framework built on a ViT backbone and demonstrate strong results on three industrial product datasets, MVTec AD, VisA and KolektorSDD2. Their findings suggest that the combination of masked image modeling and global self-attention enables robust multi-class anomaly localization in diverse industrial scenarios. Similarly, Jin et al. [11] introduce a Transformer-based incremental self-supervised anomaly detection method and show that replacing CNN backbones with ViT leads to superior anomaly detection and localization performance on the MVTec

AD dataset. The authors credit this improvement to the strong contextual modeling ability of self-attention mechanism which enhances the models capacity to capture subtle local anomalies while preserving global consistency [11]. These findings indicate that Transformer backbones are not only theoretically appealing but empirically effective for industrial defect detection tasks.

Another key motivation for selecting ViT was its natural compatibility with modern SSL methods. Methods such as MAE and DINO are designed around patch-based representations and benefit from Transformers architectures. In particular, MAE relies on masked patch reconstruction, which aligns directly with the ViT patch-token formulation. Using ViT therefore ensured architectural consistency across supervised and self-supervised pipelines. This also ensured that performance differences could be attributed to initialization strategy and pretraining procedure rather than architectural variation.

4.2 Downstream Classification Head

For downstream defect classification, a linear classification head was attached to the representation produced by the Vision Transformer backbone. ViTs process images as sequences of patch embeddings and introduce a CLS token that aggregates global information about the input image throughout the Transformer layers [27]. The output embedding corresponding to this CLS token from the final Transformer layer is used as the global image representation.

The classification head consists of a single fully connected linear layer that maps the CLS embedding to the five defect classes in the dataset. More formally, let $h \in \mathbb{R}^d$ denote the CLS embedding produced by the encoder, where d is the hidden dimension of the ViT. The classification head computes class logits as

$$z = Wh + b,$$

where $W \in \mathbb{R}^{K \times d}$ and $b \in \mathbb{R}^K$ are learnable parameters and K is the number of classes. The logits are converted into class probabilities using a softmax function

$$p_c = \frac{\exp(z_c)}{\sum_{k=1}^{K} \exp(z_k)}$$

During the downstream training, the parameters of the classification head are learned using a weighted cross-entropy loss.

Two evaluation protocols were used when training the classification head. Linear probing where the pretrained backbone encoder is frozen and only the parameters of the classification head are optimized. This protocol evaluates the quality of the learned representations. The second one is fine-tuning, where both the backbone encoder and the classification head are trained jointly on the labeled dataset, allowing the representation to adapt to the downstream task. This design follows the standard evaluation protocol used in the SSL literature for assessing representation transferability [10], [23].

4.3 Weight Initialization Strategy

For the supervised baselines, two weight initialization strategies were considered. Firstly, random initialization and secondly, supervised ImageNet pretraining. These alternatives represent the standard lower-bound and transfer-learning baselines in modern computer vision and allow the effect of prior visual knowledge to be isolated. In the first setting, the backbone network was initialized with random weights using standard initialization schemes and trained directly on the labeled Swerim dataset. This configuration reflects a scenario in which no prior visual knowledge is available and the model must learn all representations solely from the industrial defect data. It serves as a lower performance bound and provided insights into the intrinsic difficulty of the classification task under limited and imbalanced training data.

In the second setting, the backbone was initialized with weights pretrained in a supervised manner on ImageNet and subsequently fine-tuned on the labeled Swerim dataset. ImageNet pretraining represents the standard transfer learning strategy in computer vision and provides a strong baseline. However, the suitability of ImageNet features for industrial surface inspection was not self-evident. Metallic surface images are dominated by fine-grained textures and subtle structural irregularities, which differ substantially from the object-centric natural images in ImageNet.

Previous work in industrial anomaly detection reports that ImageNet-pretrained MAE models may suffer from domain shift when applied to industrial textures, potentially leading to degraded performance [3]. More broadly, it has been shown that ImageNet pretraining primarily accelerates convergence but does not necessarily improve final downstream performance when sufficient target data and training time are available [35]. These findings motivated an empirical evaluation of whether generic large-scale supervised pretraining provides a measurable advantage for metallic surface detection classification. By comparing random initialization and ImageNet pretraining under identical training protocols, the supervised baselines establish a controlled reference against which the SSL approaches could be evaluated.

4.4 Handling Class Imbalance

The defect dataset we used exhibits an imbalanced class distribution, where certain defect categories are significantly underrepresented compared to others. Such imbalance is common in industrial inspection scenarios, where defective samples occur far less frequently than defect-free or dominant defect types. Similar imbalance characteristics have been reported in strip steel and metal surface inspection datasets [2], [38], where the classifier tends to bias its predictions toward majority classes.

Class imbalance poses a fundamental challenge in defect classification and anomaly detection. As highlighted in the deep anomaly detection literature, anomalies are inherently rare and heterogeneous, resulting in skewed class distributions that can degrade detection recall and generalization performance [39]. Standard training procedures based on unweighted cross-entropy implicitly assume balanced data and therefore favor classes with larger sample sizes.

To mitigate this issue, we adopted a cost-sensitive learning strategy or a inverse frequency class weighting method by incorporating class-dependent weights into the cross-entropy loss function. Loss-adjustment-based approaches, including weighted cross-entropy, have been widely studied for unbalanced classification problems [40]. Compared to data-level resampling methods, loss reweighting preserves the original data distribution while encouraging the model to learn more discriminative representations for underrepresented defect categories. Formally, we let k denote the number of classes, n_c the number of samples in class c, and N the total number of training samples. The weight assigned to class c was defined as:

$$w_c = \frac{N}{k \cdot n_c}. \tag{4.1}$$

This formulation increases the loss contribution of minority classes while proportionally reducing the influence of majority classes. The weight vector was normalized to have unit mean to preserve numerical stability during training. The resulting weighted cross-entropy loss is given by:

$$\mathcal{L}_{CE} = -\sum_{c=1}^{k} w_c \, y_c \log(p_c), \tag{4.2}$$

where y_c denotes the ground-truth label and p_c the predicted class probability. The expression above represents the weighted cross-entropy loss for a single training sample. However, during optimization, the loss was averaged over each mini-batch.

4.5 Training Protocol and Hyperparameters

The choice of training protocol and hyperparameters were important for sufficient and reliable results and is therefore investigated in the following subsection.

4.5.1 Optimizer Choice

All supervised baselines, self-supervised pretraining runs and downstream evaluations were optimized using AdamW [41]. AdamW is a variant of Adam [42] that decouples weight decay from the adaptive gradient update which provides more reliable regularization compared to applying ℓ_2-penalties through the gradient term.

Let $g_t = \nabla_\theta \mathcal{L}(\theta_t)$ denote the gradient at step t. AdamW maintains exponential moving averages of the first and second moments:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{4.3}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t \odot g_t, \tag{4.4}$$

which are bias-corrected as

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{4.5}$$

The AdamW parameter update is then given by

$$\theta_{t+1} = \theta_t - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda\theta \right), \tag{4.6}$$

where η is the learning rate, ϵ is a numerical stability constant and λ is the decoupled weight decay coefficient.

4.5.2 Learning Rate Schedule and Batchsize

To ensure fair comparison between initialization strategies, downstream training configurations were kept consistent across experiments. Identical image preprocessing, dataset splits and optimizer family (AdamW) were used. Finally, the same evaluation protocol (linear probing or fine-tuning) was applied.

For MAE experiments, the learning rate used during self-supervised pretraining was also applied during downstream training. When evaluating different learning rate configurations, the same fixed learning rate was therefore used for both the pretext training stage and the downstream classification stage. The learning rate remained constant throughout training and no explicit learning rate scheduler was applied.

For DINO, the pretraining stage followed the learning rate schedule proposed in the original DINO framework [10]. Specifically, the learning rate was linearly increased during an initial warmup phase and subsequently decayed using cosine annealing over the remaining training iterations. In addition, weight decay was scheduled using cosine annealing between predefined start and end values. In contrast, the downstream classification stage for DINO used a fixed learning rate without scheduling. Different learning rate values were evaluated across experimental runs as part of the hyperparameter tuning procedure, as reported in Table 4.7.

Batch size was constrained by GPU memory and varied in controlled experiments using the values of 16, 32 and 64. For comparisons between initialization strategies, the downstream batch size was otherwise kept fixed to isolate the effect of representation initialization and pretraining procedure.

4.6 Experimental Design

During this project, initial development and debugging of the training pipeline were conducted on local workstations to verify correctness and ensure stable implementation. Due to the computational demands of self-supervised pretraining, all large-scale experiments were executed on Minerva, Chalmers University of Technology's high-performance computing (HPC) cluster. Minerva provides access to GPU accelerated compute nodes suitable for deep learning workloads. Depending on availability and experiment size, the models were run on nodes equipped with NVIDIA L4 (24 GB), NVIDIA L40s (48GB) and NVIDIA H100 GPUs (94 GB). These nodes feature Intel Xeon processors and up to 512 GiB-1 TiB of system memory, running Ubuntu 24.04 LTS. The availability of high-memory GPUs enables larger batch sizes and stable training while resource allocation constraints had an influence on the number

of hyperparameter configurations that were evaluated. The use of HPC resources was essential to complete pretraining and downstream fine-tuning within the thesis timeframe.

To answer the research questions, experiments were organized into a controlled pipeline consisting of (i) supervised baselines, (ii) self-supervised pretraining and (iii) downstream evaluation under identical conditions.

All methods used the same backbone family (ViT) to avoid confounding architectural effects. The following initialization strategies were compared:

- **Supervised from scratch:** random initialization trained directly on labeled data.
- **Supervised ImageNet initialization:** pretrained ImageNet weights followed by downstream training.
- **MAE pretraining:** masked reconstruction pretraining followed by downstream evaluation.
- **DINO pretraining:** teacher-student self-distillation pretraining followed by downstream evaluation.

Self-supervised representations were evaluated using two standard protocols:

- **Fine-tuning (ft):** all backbone parameters and the classification head are trained on labeled data.
- **Linear probing (lin):** the backbone is frozen and only a linear classification head is trained.

Using both protocols separates representational quality from adaptation capacity.

The hyperparameters were tuned using a sequential one-factor-at-a-time procedure. Starting from an initial configuration, one hyperparameter was varied while all others were held constant. The best performing value on the validation set was then adopted as the new default for subsequent tuning stages. The initial configuration for MAE is based on standard hyperparameters reported in prior MAE literature, see Table 4.1. These values serve as the baseline configuration before task-specific tuning. Similar, the initial configuration for DINO is based on standard hyperparameters reported in prior DINO literature, see Table 4.2. The selected configurations after hyperparameter tuning are reported in the Results sections.

Table 4.1: Standard hyperparameters for MAE.

Parameter	Value	Source
Learning rate	1.5e-4	[9]
Epochs	120	[3]
Weight decay	0.05	[9]
Batch size	32	[43]
Masking Ratio	75%	[9]
Masking	Random	[9]
Optimizer	AdamW	[9]

Table 4.2: Standard hyperparameters for DINO.

Parameter	Value	Source
Learning rate - Pretraining	6.25e-5	[10]
Epochs - Pretraining	15	[10]
Weight decay - Pretraining	0.04-0.2	[10]
Batch size	32	[10]
Teacher temperature	0.04-0.05	[10]
Student temperature	0.1	[10]
Online centering, m	0.99	[10]
Teacher momentum (EMA)	0.996	[10]
Output dimension	8192	[10]
Learning rate - Downstream	1.5e-4	[9]
Epochs - Downstream	50	[9]
Weight decay - Downstream	0.05	[9]

To quantify the benefit of self-supervised pretraining under limited supervision, label efficiency was evaluated by training downstream models using stratified subsets of the labeled training data (e.g., 1%, 5%, 10%, 20%). The validation and test sets were kept fixed. Performance trends were compared across initialization strategies under identical downstream training protocols.

Furthermore, in order to qualitatively analyze how the learned feature representations evolve during downstream training, t-distributed stochastic neighbor embedding (t-SNE) [44] was used to project high-dimensional feature embeddings into two dimensions for visualization. For each model, the representation corresponding to the CLS token from the final Transformer layer was extracted from the ViT encoder before the linear classification head. These CLS embeddings were used as global image level features.

The embeddings were extracted from the validation set at selected training epochs and projected into two dimensions using t-SNE. To improve stability and reduce noise, principal component analysis (PCA) was first applied as a preprocessing step before the t-SNE projection. After PCA was applied, the dimensionality was reduced

of the embeddings to 50 dimensions. This preprocessing step reduces noise and improves the stability and computational efficiency of the t-SNE algorithm. The reduced feature representations were then projected to two dimensions using t-SNE and visualized with points colored according to their ground-truth defect class. The t-SNE visualizations were used only for qualitative analysis of class separation and feature organization. Since linear probing keeps the backbone encoder frozen, the CLS embeddings remain unchanged across epochs in that setting. Therefore, t-SNE visualizations across training epochs were only analyzed for the fine-tuning protocol.

5

Evaluation Framework

This chapter presents the evaluation framework used to assess the performance of both the supervised and self-supervised approaches. In particular, a supervised Vision Transformer-based model is presented to provide a reference point for subsequent self-supervised results. Additionally, results from prior work by Swerim using a YOLO-based approach are included to contextualize performance within existing industrial solutions.

5.1 Evaluation Metrics

Due to the strong class imbalance in the dataset, performance was evaluated using metrics that reflect per-class behavior rather than being dominated by the majority class. Balanced accuracy is defined as the mean recall across classes:

$$\mathrm{BA} = \frac{1}{K} \sum_{c=1}^{K} \frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FN}_c}, \tag{5.1}$$

where K is the number of classes and TP_c and FN_c denote true positives and false negatives for class c.

Macro F1-score is computed as the unweighted mean of per-class F1-scores:

$$\mathrm{F1}_{macro} = \frac{1}{K} \sum_{c=1}^{K} \mathrm{F1}_c, \tag{5.2}$$

where

$$\mathrm{F1}_c = \frac{2\,\mathrm{Precision}_c\,\mathrm{Recall}_c}{\mathrm{Precision}_c + \mathrm{Recall}_c}. \tag{5.3}$$

In addition to the primary metrics above, overall accuracy, per-class precision, per-class recall and per-class F1-score were computed for diagnostic analysis. Confusion matrices were used to analyze class-wise error patterns and missclassification behavior.

5.2 Supervised Baseline

To establish a reference point for the self-supervised experiments, a supervised baseline using a ViT backbone was trained for 50 epochs. The architecture used matches the backbone used in the MAE and DINO experiments to ensure a fair comparison between supervised and self-supervised initialization strategies. The model was both trained from scratch and initialized with pretrained ImageNet weights. It was later evaluated under both fine-tuning (ft) and linear probing (lin) protocols. The test results are presented in Table 5.1. The results show both balanced accuracy (Bal Acc) and macro F1-score. The confusion matrices for the supervised baseline can be found in Table 5.2 and Table 5.3.

Table 5.1: Supervised baseline trained on 50 epochs on test data.

	Bal Acc		macro F1	
Baseline	ft	lin	ft	lin
Supervised, Imagenet	96.59	98.86	0.9239	0.9688
Supervised, scratch	94.86	86.40	0.6952	0.5511

Table 5.2: Confusion matrices for the supervised model pretrained on Imagenet weights on the test set under fine-tuning (top) and linear probing (bottom).

Fine-tuning (ft)

True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	3904	4	5	21	5
Scrap Mark II	4	116	0	0	0
Scrap Mark	5	0	98	0	0
Cracks	2	0	0	32	0
Overfill	3	0	0	0	140

Linear probing (lin)

True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	3907	7	12	0	13
Scrap Mark II	1	119	0	0	0
Scrap Mark	2	0	101	0	0
Cracks	0	0	0	34	0
Overfill	3	0	0	0	140

Table 5.3: Confusion matrices for the supervised model trained from scratch on the test set under fine-tuning (top) and linear probing (bottom).

Fine-tuning (ft)

True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	3536	49	205	85	64
Scrap Mark II	0	120	0	0	0
Scrap Mark	10	0	93	0	0
Cracks	1	0	0	33	0
Overfill	4	0	0	0	139

Linear probing (lin)

True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	3027	50	231	94	537
Scrap Mark II	0	120	0	0	0
Scrap Mark	41	0	62	0	0
Cracks	1	0	0	33	0
Overfill	3	0	0	0	140

5.3 Previous Work With Supervised YOLO Baseline

Prior to this work, Swerim has conducted supervised experiments on the same dataset using a YOLO-based model. For that experiment, the model was initialized with pretrained ImageNet weights and evaluated using a fine-tuning set up. No self-supervised pretraining was applied and the model was trained fully in a supervised manner on labeled data using a separate data split compared to this work. YOLO is a widely used object detection framework that performs detection and classification in a forward pass, enabling efficient and real-time predictions [45]. In this context, the model is adapted for defect classification, where predictions correspond to defect categories rather than bounding box detection.

The performance of the YOLO model on the test set shows that the model achieves a balanced accuracy of 98.98 % and a macro F1-score of 0.9891, which indicates strong and consistent performance across classes. The confusion matrix for the YOLO model is shown in Table 5.4. It can be observed that the model achieves near-perfect classification across all defect classes. The majority of error originate from the "no defect" class being misclassified as defect categories, although these error are relatively few compared to the total number of samples. The defect classes themselves are classified with minimal confusion between classes.

Table 5.4: Confusion matrix for the YOLO model pretrained on Imagenet weights on the original test set under fine-tuning.

Fine-tuning (ft)

True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	5252	2	4	0	3
Scrap Mark II	2	159	0	0	0
Scrap Mark	5	0	131	0	0
Cracks	0	0	0	49	0
Overfill	0	0	0	0	207

These results demonstrate that a fully supervised approach using YOLO can achieve very strong performance on this dataset when sufficient labeled data is available. In particular, the high macro F1-score indicates strong class-wise balance which suggests that the model effectively distinguishes between different defect types. However, it is important to note that this approach relies entirely on labeled data, in contrast to the self-supervised methods explored in this thesis. While YOLO achieves higher performance, it does not address scenarios with limited labeled data, where self-supervised approaches such as MAE and DINO may offer advantages in label efficiency.

6

Results for MAE

In this chapter follows the result of the self-supervised model MAE. The model is initialized both by using ImageNet-pretrained weights and by training from scratch. The model is then evaluated in both fine-tuning (ft) and linear probing (lin) settings. All experiments are conducted using a fixed train/validation/test split, described in Section 3. Performance is measured using balanced validation accuracy (average recall per class) and macro F1-score.

6.1 MAE Model Using Pretrained ImageNet Weights

The results for the MAE model using pretrained ImageNet weights are presented in Table 6.1, where each sub-table corresponds to a specific hyperparameter evaluated during the tuning process. Model selection is primarily based on balanced accuracy and secondarily on macro F1-score.

In Table 6.1(a), the impact of data augmentation strategies is evaluated. The configuration without augmentation ("None") achieves the strongest overall performance in both fine-tuning and linear probing, reaching 98.66% resp 98.69% balanced accuracy. It also yields the highest macro F1-score under fine-tuning (0.9545) and competitive performance under linear probing (0.9006). It can be seen that more aggressive augmentations, particularly random cropping combined with color jitter ("Crop + color jit"), resulted in a noticeable decrease in macro F1-score which indicated reduced stability across classes. Therefore no augmentation is used for the further experiments.

Moving on to further experiments, Table 6.1(b) is the next subsequent experiment which evaluates different masking strategies. Random masking achieves strong and consistent performance across both evaluation protocols with 98.66% balanced accuracy for fine-tuning resp. 98.69% balanced accuracy for linear probing. It can be seen that grid masking produces slightly higher linear probing accuracy (99.02%) but it does not consistently improve macro F1. Given the balanced performance across both metrics, random masking is selected.

Table 6.1: Hyperparameter tuning on the MAE model with pretrained ImageNet weights, where ft denotes fine-tuning and lin denotes linear probing. Bold values indicate the selected hyperparameter configuration.

(a) Data Augmentations

	Bal Acc		macro F1	
Type	ft	lin	ft	lin
None	98.66	98.69	0.9545	0.9006
Crop fix	98.57	98.72	0.9264	0.8677
Crop rand	96.97	97.08	0.9096	0.8015
Crop + color jit	96.39	95.48	0.8427	0.7290

(b) Masking

	Bal Acc		macro F1	
Type	ft	lin	ft	lin
Rand	98.66	98.69	0.9545	0.9006
Block	98.41	98.90	0.9481	0.9098
Grid	98.15	99.02	0.9608	0.9014

(c) Masking Ratio

	Bal Acc		macro F1	
Ratio	ft	lin	ft	lin
40%	98.36	98.55	0.9353	0.8936
75%	98.66	98.69	0.9545	0.9006
90%	98.72	99.20	0.9388	0.8989

(d) Epochs

	Bal Acc		macro F1	
Epochs	ft	lin	ft	lin
25	97.31	98.28	0.9380	0.8400
50	98.72	99.20	0.9388	0.8989
75	97.91	99.16	0.9718	0.9378

(e) Learning Rate

	Bal Acc		macro F1	
LR	ft	lin	ft	lin
1.5e-4	98.72	99.20	0.9388	0.8989
3e-4	97.43	99.01	0.8758	0.9247
5e-4	94.01	94.92	0.7686	0.6355

(f) Weight Decay

	Bal Acc		macro F1	
WD	ft	lin	ft	lin
0.01	98.52	98.82	0.9423	0.9010
0.05	98.72	99.20	0.9388	0.8989
0.1	98.63	98.61	0.9482	0.8930

(g) Batch Size

	Bal Acc		macro F1	
BS	ft	lin	ft	lin
16	98.01	98.70	0.9538	0.9311
32	98.72	99.20	0.9388	0.8989
64	97.47	98.81	0.9552	0.8629

The influence of masking ratio is shown in Table 6.1(c). Increasing the ratio from 40% to 90% leads to gradual increase in balanced accuracy, with 90% achieving the

highest performance (98.72% for fine-tuning, 99.20% for linear probing). Macro F1 remains stable across higher masking ratios and is therefore supporting the selection of 90%. Further on, Table 6.1(d) examines the number of pretext and downstream training epochs. Training for 50 epochs yields the highest balanced accuracy with 98.72% for fine-tuning and 99.20% for linear probing. Increasing the training duration to 75 epochs results in a noticeable improvement in macro F1-score for fine-tuning (0.9718 compared to 0.9388 at 50 epochs), indicating improved precisionrecall balance for certain classes. However, this improvement does not translate into higher balanced accuracy and is accompanied by increased computational cost. Therefore, 50 epochs are selected as the final configuration, providing strong performance while maintaining computational efficiency.

Moving on, the learning rate is compared in Table 6.1(e) and shows that 1.5e-4 achieves the strongest results across both balanced accuracy and macro F1. Higher learning rate significantly reduce the performance, particularly in macro F1, which suggests decreased stability during optimization. Similarly, Table 6.1(f) demonstrates that a weight decay of 0.05 provides the best balance between balanced accuracy and macro F1, outperforming both lower and higher regularization strengths. Finally, Table 6.1(g) evaluates batch size. A batch size of 32 achieves the highest balance accuracy (98.72% for fine-tuning, 99.20% for linear probing) while maintaining stable macro F1 performance. Smaller or larger batch size do not yield consistent improvements and are therefore not selected.

Based on the sequential tuning procedure and the evaluation criteria described above, the final MAE configuration using pretrained ImageNet weights can be seen in Table 6.2.

Table 6.2: Final MAE configuration using pretrained ImageNet weights.

Parameter	Value
Data augmentations	None
Masking	Random
Masking ratio	90%
Epochs	50
Learning rate	1.5e-4
Weight decay	0.05
Batch size	32

6.1.1 Label Efficiency

The label efficiency results for the final MAE configuration evaluated on the validation set is shown in Table 6.3. The model was trained using 1%, 5%, 10%, 20% and 100% of the labeled training data and evaluated using balanced accuracy (Bal Acc) and macro F1-score (macro F1). Results are shown for both fine-tuning and linear probing.

Table 6.3: Label efficiency for the final MAE configuration using pretrained ImageNet weights, taken the average over three runs.

	Bal Acc		macro F1	
Percent	ft	lin	ft	lin
1%	74.41	78.80	0.7905	0.6599
5%	91.51	96.97	0.9175	0.6980
10%	94.65	97.48	0.9325	0.7277
20%	96.77	98.30	0.9278	0.7887
100%	97.80	99.08	0.9472	0.8985

Overall, performance improves consistently as the amount of labeled data increases which is expected. Even with only 1% of labeled data, the model achieves a balanced accuracy of 74.41% (ft) and 78.80% (lin), which indicates that the MAE pretraining provides a strong representation that generalize well under label scarcity. When increasing the labeled data to 5%, performance improves substantially. Linear proving reaches 96.97% balanced accuracy, outperforming fine-tuning (91.51%). This suggests that, in this settings, the pretrained representations are already highly discriminative and benefit from keeping the encoder frozen. From 10% labeled data and above, both approaches achieve very high balanced accuracy (above 94%). Linear probing remains slightly higher in balanced accuracy, reaching 98.30% at 20% of labeled data. However, when considering macro F1-score, fine-tuning consistently outperforms linear probing for all label fractions. This indicates that fine-tuning yields better class-wise balance in precision and recall. These fine-tuning results combined with the the results of linear probing shows a trade-off where linear probing maximizes recall balance across classes while fine-tuning improves the overall precision-recall balance per class. Importantly, the results demonstrate strong label efficiency of the MAE-pretrained model. High balanced accuracy is achieved already with 5-10% of labeled data, which indicates that only a small fraction of annotated samples is required to obtain sufficient performance.

6.1.2 T-SNE for Different Epochs

To analyze the how the learned feature representations evolve during training, the CLS token embeddings from the validations set is visualized using t-SNE. The CLS embeddings is extracted from the final transformer layer and by visualizing the CLS embeddings, insight is given into how the model organizes image-level representation in feature space and also how class separability evolved during training. The t-SNE is visualized during fine-tuning and not linear probing.

The t-SNE projections of the CLS embeddings at epochs 0, 5, 25 and 50 during fine-tuning is shown in Figure 6.1. At epoch 0, the representations already have some structure due to the pretrained initialization. However the majority class (no_defect) dominates the embedding space and minority defect classes are not clearly separated. By further training, it can be seen at epoch 5 that the embedding space begins to reorganize. Some defect classes start forming more compact

regions, which indicates that the model is adapting its feature extractor to the downstream defect classification task. Further along at epoch 25, class separation becomes more pronounced. Minority defect classes such as scrap mark, scrap mark II and overfill appear as more compact and distinguishable clusters. This suggests that the model is learning task-specific representations that more effectively capture differences between classes. Finally, by epoch 50, clusters become further refined and more clearly separated. The samples within each class become more clustered and the classes move further apart, showing that fine-tuning improves how well the model separates them.

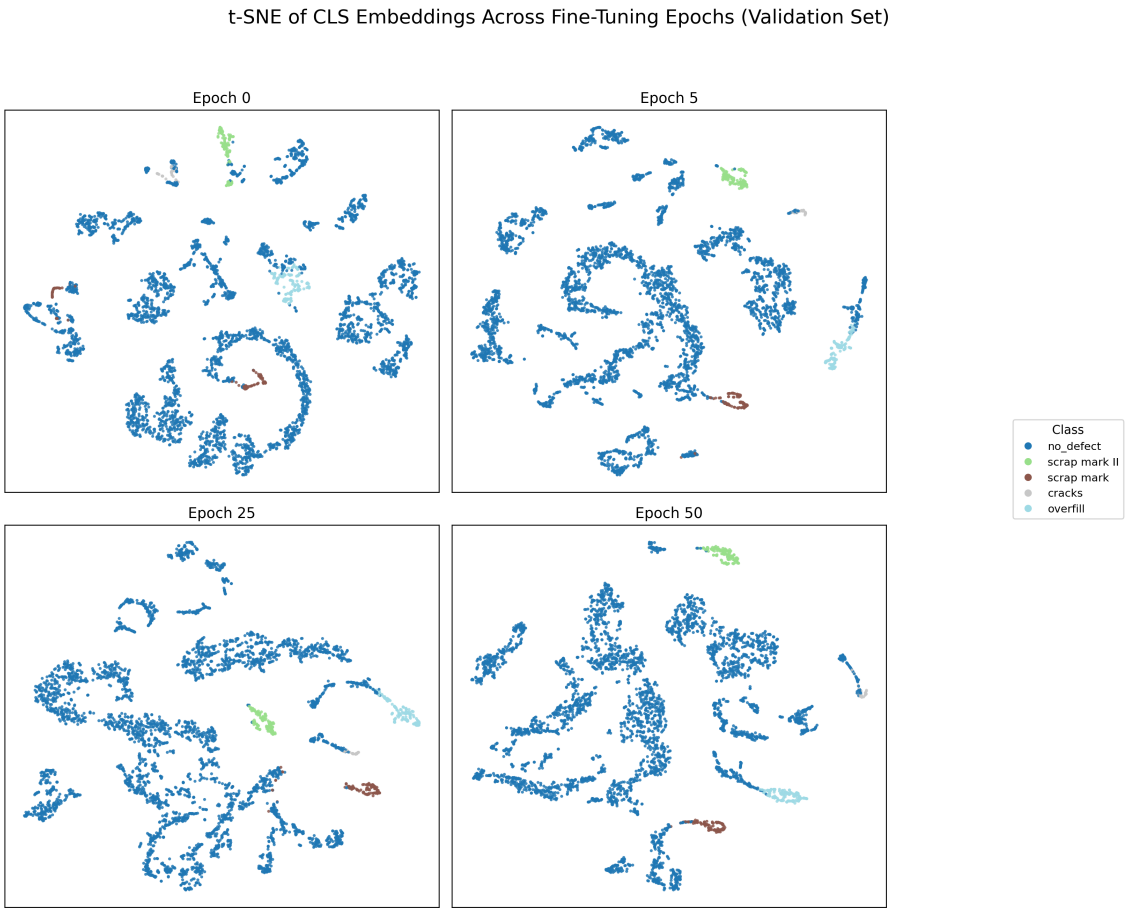


Figure 6.1: t-SNE visualization of the MAE model pretrained on Imagenet weights, trained on the validation set across different epochs.

6.1.3 Final Model Performance

After selecting the final MAE configuration based on validation performance, evaluation is performed on the held-out test set to obtain unbiased estimate of generalization performance. The test set is not used during hyperparameter tuning or model selection. The summarized test results are shown in Table 6.4.

Table 6.4: Test results on the final MAE configuration using pretrained ImageNet weights.

Evaluation metric	ft	lin
Test balanced accuracy (%)	98.89	99.60
Test macro F1-score	0.9623	0.9073

Under linear probing, the final model achieves a test balanced accuracy of 99.60% and a macro F1-score of 0.9073. The per-class F1-score range from 0.77 to 0.99 with the lowest value observed for the "cracks" class (0.773), which is primarily explained by lower precision despite perfect recall. The corresponding confusion matrix is shown in Table 6.5, bottom which indicates that most errors originate from samples in the "no_defect" class being predicted as defects, while the remaining defect classes are classified with near-perfect recall.

Table 6.5: Confusion matrices for the final MAE configuration pretrained on ImageNet weights on the test set under fine-tuning (top) and linear probing (bottom).

Fine-tuning (ft)

True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	3912	6	10	5	6
Scrap Mark II	0	120	0	0	0
Scrap Mark	5	0	98	0	0
Cracks	0	0	0	34	0
Overfill	0	0	0	0	143

Linear probing (lin)

True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	3861	18	23	20	17
Scrap Mark II	0	120	0	0	0
Scrap Mark	0	0	103	0	0
Cracks	0	0	0	34	0
Overfill	0	0	0	0	143

For fine-tuning, the model achieves a test balanced accuracy of 98.89% and a macro F1-score of 0.9623. Per-class F1-score range from 0.93 to 0.996, which indicates strong and consistent performance across all defect categories. Compared to linear probing, fine-tuning substantially improves macro F1-score which suggests more balanced performance across classes. This improvement is particularly visible for the "cracks" class, where the F1-score increases from 0.773 to 0.932 which reflect improved precision while maintaining perfect recall. The confusion matrix (see Table 6.5, top) further shows reduced confusion across categories compared to linear probing.

Overall, the two protocols emphasize different strengths on the test set. Linear probing achieves the higher balanced accuracy, whereas fine-tuning yields a noticeable higher macro F1-score and more consistently high per-class F1 values.

6.1.4 Comparison With Supervised YOLO Baseline

To get a context of the performance of the MAE model initialized with ImageNet weights, the results are compared to a previously developed supervised YOLO-based model evaluated on the same original data split. The test result for both the MAE and the YOLO model can be found in Table 6.6.

Table 6.6: Test results on the final MAE configuration and YOLO configuration using pretrained ImageNet weights on the original data split (fine-tuning).

Evaluation metric	MAE	YOLO
Test balanced accuracy (%)	97.90	98.98
Test macro F1-score	0.9410	0.9891

The YOLO model achieves a test balanced accuracy of 98.98% and a macro F1-score of 0.9891. In comparison, the MAE model reaches a balanced accuracy of 97.90% and a macro F1-score of 0.9410. While the difference in balanced accuracy is relatively small, a more noticeable gap is observed in macro F1-score which indicates that the YOLO model achieves stronger class-wise balance. Further, the confusion matrices highlight these differences. The confusion matrix for the YOLO model can be found in Table 5.4 while the confusion matrix for the MAE model can be found in Table 6.7.

Table 6.7: Confusion matrix for the MAE model pretrained on ImageNet weights on the original test set under fine-tuning.

	Fine-tuning (ft)				
True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	5211	14	7	17	10
Scrap Mark II	2	158	0	0	0
Scrap Mark	3	0	132	0	0
Cracks	2	0	0	46	0
Overfill	4	0	0	0	202

The YOLO model demonstrates near-perfect classification across all defect classes, with very few misclassifications. In contrast, the MAE model shows a higher number of errors, particularly for the "no defect" class being misclassified as defect classes. This indicates that while MAE learns strong representations, it is less effective than the fully supervised approach at separating defect from non-defect samples. The relative small drop in balanced accuracy suggests that the MAE model is able to

learn highly competitive representation, even without supervised pretraining. However, the lower macro F1-score indicates that these representations are less robust in terms of class-wise precision and recall. Overall, the comparison shows that while supervised methods such as YOLO still achieves the strongest performance when sufficient labeled data is available, self-supervised approaches like MAE provide a competitive alternative, particularly in scenarios where labeled data is limited.

6.2 MAE Model Trained From Scratch

The hyperparameter optimization results for the MAE model trained from scratch are presented in Table 6.8, where each sub-table corresponds to a specific hyperparameter evaluated during the tuning process. Model selection is primarily based on balanced accuracy and secondarily on macro F1-score.

In Table 6.8(a), the impact of data augmentation strategies is evaluated. Similar to the pretrained setting, the configuration without augmentation ("None") achieves the strongest overall performance, reaching 92.65% and 94.24% balanced accuracy for fine-tuning and linear probing respectively. More aggressive augmentations, particularly cropping combined with color jitter, lead to a notable drop in both balanced accuracy and macro F1-score, indicating reduced stability across classes. This suggests that, when training from scratch, the model is more sensitive to augmentation and struggles to generalize under strong transformations. Therefore, no augmentation is selected for the final configuration. Table 6.8(b) evaluates different masking strategies. Unlike the pretrained case, grid masking yields the strongest performance, achieving the highest balanced accuracy (95.12% fine-tuning, 95.23% linear probing) and macro F1-score. This indicates that a more structured masking strategy benefits representation learning when no prior knowledge is available, likely by enforcing more consistent spatial reasoning. Consequently, grid masking is selected.

The influence of masking ratio is shown in Table 6.8(c). Increasing the ratio from 40% to 75% improves both balanced accuracy and macro F1-score, while a further increase to 90% leads to a slight decrease in performance. This contrasts with the pretraining setting, where higher masking ratios were beneficial. The results suggest that when training from scratch, excessively high masking removes too much information and hinders learning. Therefore, a masking ratio of 75% is selected. Table 6.8(d) examines the number of training epochs. Training for 50 epochs achieves a strong balance between performance and stability, with competitive accuracy and macro F1-score. Increasing the number of epochs to 75 does not provide consistent improvements and introduces a slight instability, particularly for linear probing. Thus, 50 epochs are selected as the final configuration.

Table 6.8: Hyperparameter tuning on the MAE model trained from scratch.

(a) Data Augmentations

	Bal Acc		macro F1	
Type	ft	lin	ft	lin
None	92.65	94.24	0.7195	0.6578
Crop fix	90.14	94.76	0.6702	0.6528
Crop rand	92.41	94.11	0.6757	0.6293
Crop + color jit	80.10	81.54	0.5195	0.4683

(b) Masking

	Bal Acc		macro F1	
Type	ft	lin	ft	lin
Rand	92.65	94.24	0.7195	0.6578
Block	95.27	93.93	0.6907	0.6403
Grid	95.12	95.23	0.7131	0.6951

(c) Masking Ratio

	Bal Acc		macro F1	
Ratio	ft	lin	ft	lin
40%	95.30	92.02	0.6782	0.6489
75%	95.12	95.23	0.7131	0.6951
90%	91.48	93.73	0.7100	0.6685

(d) Epochs

	Bal Acc		macro F1	
Epochs	ft	lin	ft	lin
25	93.83	92.96	0.7416	0.6032
50	95.12	95.23	0.7131	0.6951
75	94.57	95.11	0.7358	0.6913

(e) Learning Rate

	Bal Acc		macro F1	
LR	ft	lin	ft	lin
1.5e-4	95.12	95.23	0.7131	0.6951
3e-4	92.57	86.44	0.6256	0.5705
5e-4	93.94	70.71	0.6563	0.4965

(f) Weight Decay

	Bal Acc		macro F1	
WD	ft	lin	ft	lin
0.01	90.35	95.02	0.6911	0.6969
0.05	95.12	95.23	0.7131	0.6951
0.1	95.72	92.99	0.7367	0.6410

(g) Batch Size

	Bal Acc		macro F1	
BS	ft	lin	ft	lin
16	93.10	90.60	0.7309	0.6873
32	95.12	95.23	0.7131	0.6951
64	95.93	94.72	0.7704	0.6443

For the learning rate (Table 6.8(e)), it can be seen that 1.5e-4 achieves the best overall performance in both metrics. Higher learning rates significantly degrade performance, particularly for linear probing, indicating unstable optimization. This sensitivity further highlights the increased difficulty of training from scratch. Sim-

ilarly, Table 6.8(f) shows that a weight decay of 0.05 provides the best balance between regularization and performance.

Finally, Table 6.8(g) evaluates batch size. A batch size of 32 achieves strong and consistent results across both balanced accuracy and macro F1-score, while smaller and larger batch sizes do not provide consistent improvements. Therefore, a batch size of 32 is selected. Based on the sequential tuning procedure and the evaluation criteria described above, the final MAE configuration trained from scratch is summarized in Table 6.9.

Table 6.9: Final MAE configuration trained from scratch.

Parameter	Value
Data augmentations	None
Masking	Grid
Masking ratio	75%
Epochs	50
Learning rate	1.5e-4
Weight decay	0.05
Batch size	32

6.2.1 Label Efficiency

The label efficiency results for the MAE model trained from scratch are presented in Table 6.10. Compared to the pretrained setting, performance is consistently lower across all label fractions, highlighting the importance of pretrained representation.

Table 6.10: Label efficiency for the final MAE configuration trained from scratch, taken the average over three runs.

	Bal Acc		macro F1	
Percent	ft	lin	ft	lin
1%	79.66	76.34	0.6997	0.5317
5%	86.29	89.30	0.7401	0.5871
10%	91.78	91.55	0.7195	0.6191
20%	90.11	94.44	0.7273	0.6469
100%	94.27	95.44	0.7190	0.6912

With only 1% labeled data, the model achieves 79.66% balanced accuracy for fine-tuning and 76.34% for linear probing, which indicates limited capability to generalize under extreme data scarcity. As the amount of labeled data increases, performance improves steadily, reaching 94.27% (fine-tuning) and 95.44% (linear probing) balanced accuracy at 100% labeled data. Unlike the pretrained case, linear probing consistently achieves comparable or slightly higher balanced accuracy than fine-tuning across most label fractions. However, fine-tuning generally yields higher macro F1-scores, indicating improved class-wise balance. This suggests that while

linear probing can effectively leverage learned features for overall accuracy, fine-tuning is necessary to improve precision-recall balance across classes. Overall, the results demonstrate that training from scratch results in reduced label efficiency compared to pretrained models, by requiring a larger fraction of labeled data to achieve competitive performance.

6.2.2 T-SNE for Different Epochs

To analyze how feature representations evolve during training, t-SNE visualizations of the CLS token embeddings are shown in Figure 6.2 for epochs 0, 5, 25 and 50.

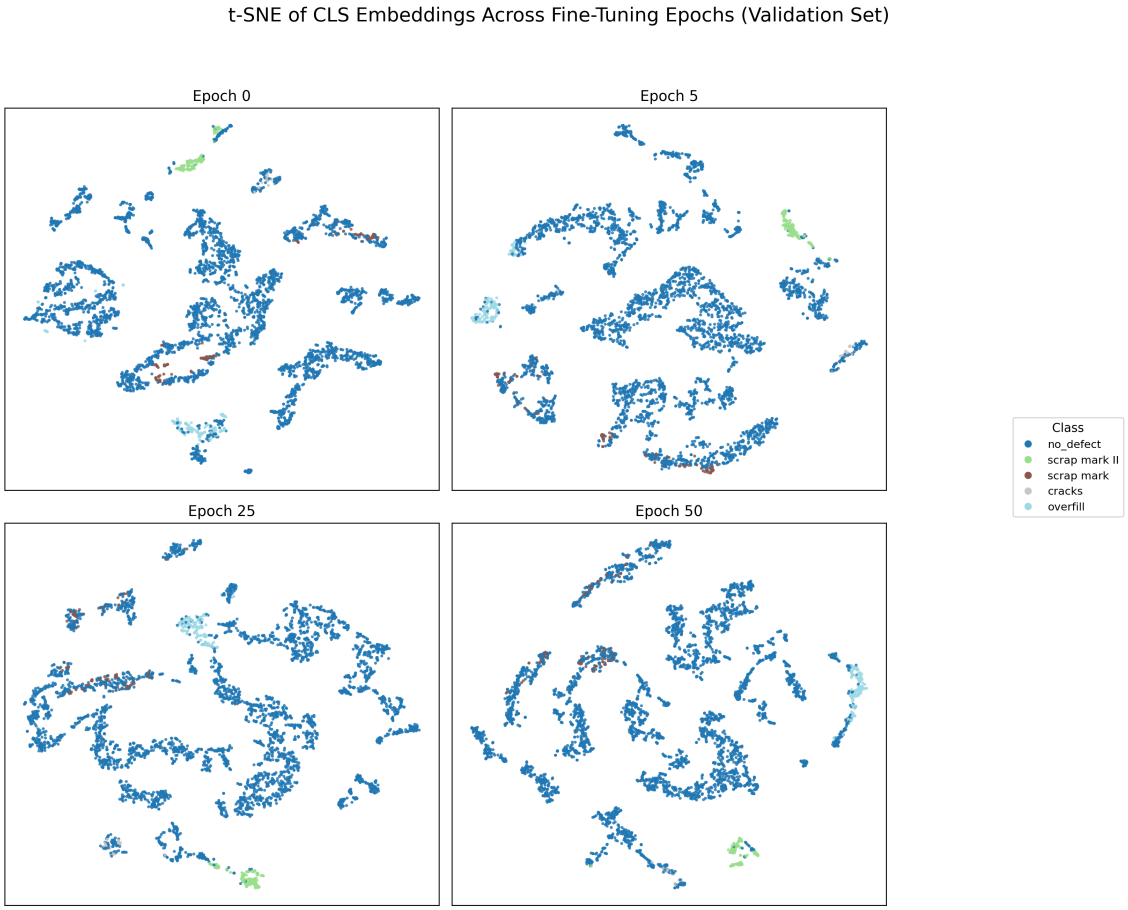


Figure 6.2: t-SNE visualization of the MAE model trained from scratch, trained on the validation set across different epochs.

At epoch 0, the embeddings appear largely unstructured with most samples overlapping and no clear separation between classes. This contrasts with the pretrained model, where some initial structure is already present due to prior knowledge. By epoch 5, early signs of organization emerge with som classes beginning to form loose clusters. However, class separation remains limited and highly overlapping. At epoch 25, the structure becomes more pronounced with clearer grouping of certain classes, although some classes (for example the class "scrap mark") are still not well separated. At epoch 50, clusters become more compact and distinguishable which

indicates that the model has learned more discriminative representations. However, compared to the pretrained setting, the clusters remain less well-separated and more diffuse which shows weaker feature quality. These observations confirm that, without pretrained initialization, the model requires more training to develop meaningful representations and still does not reach the same level of separability.

6.2.3 Final Model Performance

The final model performance on the test set is presented in Table 6.11. The model achieves a balanced accuracy of 93.49% for fine-tuning and 94.51% for linear probing, with corresponding macro F1-scores of 0.6945 and 0.6840. Compared to the pretrained model, this represents a clear performance gap, particularly in macro F1-score which indicates a reduced class-wise balance.

Table 6.11: Test results on the final MAE configuration trained from scratch

Evaluation metric	ft	lin
Test balanced accuracy (%)	93.49	94.51
Test macro F1-score	0.6945	0.6840

The confusion matrices in Table 6.12 show that most errors originate from the "no defect" class being misclassified as defect categories. In contrast, the defect classes themselves are classified with high accuracy and minimal confusion between classes.

Table 6.12: Confusion matrices for the final MAE configuration trained from scratch on the test set under fine-tuning (top) and linear probing (bottom).

Fine-tuning (ft)

True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	3491	5	284	98	61
Scrap Mark II	1	119	0	0	0
Scrap Mark	8	0	95	0	0
Cracks	0	0	0	34	0
Overfill	14	0	4	0	125

Linear probing (lin)

True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	3458	29	292	89	71
Scrap Mark II	0	120	0	0	0
Scrap Mark	15	0	88	0	0
Cracks	0	0	0	34	0
Overfill	1	0	0	0	142

Compared to linear probing, fine-tuning improves macro F1-score which indicates better precision-recall balance across classes. However, linear probing achieves slightly higher balanced accuracy which suggests stronger overall performance. Overall, the results highlight that while the model trained from scratch is capable of learning useful representation, it remains consistently inferior to the pretrained counterpart in terms of both accuracy and class-wise robustness.

7

Results for DINO

In this chapter follows the result of the self-supervised model DINO. The model is initialized both by using ImageNet-pretrained weights and by training from scratch. The model is then evaluated in both fine-tuning (ft) and linear probing (lin) settings. All experiments are conducted using a fixed train/validation/test split, described in Section 3. Performance is measured using balanced validation accuracy (average recall per class) and macro F1-score.

7.1 DINO Model Using Pretrained ImageNet Weights

The hyperparameter tuning results for the DINO model using pretrained ImageNet weights are presented in Table 7.1, where model selection is based primarily on balanced accuracy and secondarily on macro F1-score.

For online centering shown in Table 7.1(a), it can be seen that $m = 0.9$ gives the most consistent performance overall and is therefore selected. Although it can be seen that $m = 0.999$ yields the highest balanced accuracy for fine-tuning but this comes with a clear drop in macro F1-score for linear probing which indicates less stable class-wise performance. Regarding the output dimension shown in Table 7.1(b), the output dimension of 4096 performs best across both fine-tuning and linear probing, giving the highest balance accuracy and the strongest macro F1-score overall. The number of epochs, evaluated in Table 7.1(c), has a clear effect on performance. Increasing the training from 10 to 15 epochs improved both balanced accuracy and macro F1-score, whereas 20 epochs leads to lower results. A similar pattern can be seen in batch size shown in Table 7.1(d), where 32 gives the strongest result across both evaluation protocols except for the macro F1-score for linear probing which is slightly reduced.

For the downstream optimization parameters, the learning rate evaluated in Table 7.1(e) benefits from lower rates. A learning rate of 1.5e-4 shows the best results for fine-tuning while higher learning rates reduce both balanced accuracy and macro F1-score. Linear probing results slightly favored higher learning rates, although the improvement was negligible. Weight decay, shown in Table 7.1(f), gives the best results for balanced accuracy and macro F1-score for fine-tuning at 0.05 and is therefore selected. Linear probing results slightly favored weight decay at 0.01, although the improvement was negligible here as well.

The final DINO configuration using pretrained ImageNet weights is presented in Table 7.2.

Table 7.1: Hyperparameter tuning on the DINO model with pretrained ImageNet weights, where ft denotes fine-tuning and lin denotes linear probing. Bold values indicate the selected hyperparameter configuration.

(a) Online centering

	Bal Acc		macro F1	
m	ft	lin	ft	lin
0.9	97.63	97.73	0.8701	0.7702
0.99	97.18	97.26	0.9152	0.7594
0.999	98.10	96.01	0.8956	0.6875

(b) Output dimension

	Bal Acc		macro F1	
dim	ft	lin	ft	lin
4096	97.89	98.64	0.9273	0.7845
8192	97.63	97.73	0.8701	0.7702
65536	97.46	98.51	0.9235	0.7619

(c) Epochs

	Bal Acc		macro F1	
Epochs	ft	lin	ft	lin
10	96.60	98.21	0.9260	0.7647
15	97.89	98.64	0.9273	0.7845
20	97.46	97.59	0.9174	0.7505

(d) Batch Size

	Bal Acc		macro F1	
BS	ft	lin	ft	lin
16	97.11	98.61	0.9225	0.8207
32	97.89	98.64	0.9273	0.7845
64	97.46	98.25	0.9248	0.7983

(e) Learning Rate - Downstream

	Bal Acc		macro F1	
LR	ft	lin	ft	lin
1.5e-4	97.89	98.64	0.9273	0.7845
3e-4	97.67	98.97	0.8761	0.8211
5e-4	96.26	99.13	0.8110	0.8453

(f) Weight Decay - Downstream

	Bal Acc		macro F1	
WD	ft	lin	ft	lin
0.01	96.91	98.71	0.9246	0.7908
0.05	97.89	98.64	0.9273	0.7845
0.1	96.75	98.58	0.9173	0.7785

Table 7.2: Final DINO configuration using pretrained ImageNet weights.

Parameter	Value
Online centering	0.9
Output dimension	4096
Epochs	15
Batch size	32
Learning rate - Downstream	1.5e-4
Weight decay - Downstream	0.05

7.1.1 Label Efficiency

The label efficiency results for the final DINO configuration evaluated on the validation set are presented in Table 7.3.

Table 7.3: Label efficiency for the final DINO configuration using pretrained ImageNet weights, taken the average over three runs.

	Bal Acc		macro F1	
Percent	ft	lin	ft	lin
1%	80.72	87.43	0.8181	0.4201
5%	90.44	91.75	0.8928	0.4541
10%	91.84	92.87	0.9168	0.4974
20%	95.59	96.91	0.9126	0.6570
100%	97.68	98.60	0.9203	0.7868

Overall, performance improves steadily as the amount of labeled data increases. Even with only 1% of labeled data, the model achieves a balance accuracy of 80.72% for fine-tuning and 87.43% for linear probing. This indicates that the pretrained DINO representations provide strong generalization under extreme label scarcity. Linear probing consistently achieves slightly higher balanced accuracy than fine-tuning across all label fractions. However, a pattern is observed for macro F1-score. With very limited data, linear probing performs substantially worse than fine-tuning, reaching only 0.4201 at 1% labeled data compared to 0.8181 for fine-tuning.

As the amount of labeled data increases, the gap between the two approaches decreases. From 10% labeled data and above, both methods achieve high balanced accuracy by exceeding 91%. At full supervision, the results become very similar in terms of balanced accuracy (97.68% for fine-tuning and 98.60% for linear probing). Nevertheless, fine-tuning consistently provides higher macro F1-score across all label fractions. These results indicate that the pretrained DINO representations are highly label efficient, achieving strong performance even with limited supervision. At the same time, the comparison between fine-tuning and linear probing highlights a similar trade-off to the one observed for MAE which is that linear probing slightly favors overall balanced accuracy, while fine-tuning leads to better class-wise precision-recall balance.

7.1.2 T-SNE for Different Epochs

For the t-SNE analysis, the CLS embeddings are, as described previously, extracted from the final transformer layer and projected into two dimensions to illustrate how the model organizes image representations in feature space. The visualization is again performed for the fine-tuning setting only. The t-SNE projections of the CLS Embeddings at epochs 0, 5, 25 and 50 are shown in Table 7.1.

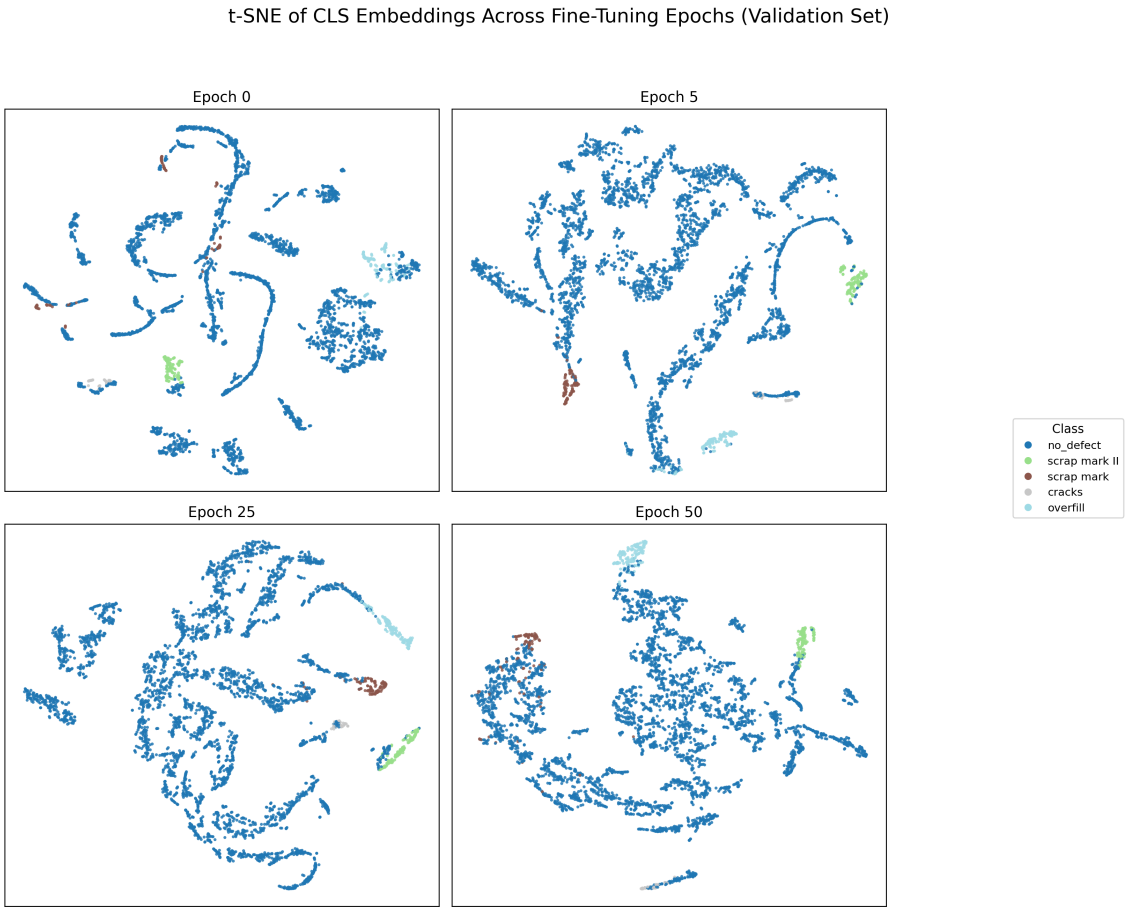


Figure 7.1: t-SNE visualization of the DINO model pretrained on Imagenet weights, trained on the validation set across different epochs.

At epoch 0, the embeddings already exhibit some structure due to the ImageNet pretraining but not as well as the MAE model performed under the same circumstances. It can also be seen that the majority class (no_defect) dominates the feature space and minority defect classes appear scattered and partially overlapping. After a few epochs of fine-tuning, the embedding space begins to reorganize. By epoch 5, some defect classes starts forming more localized regions, indicating that the model is adapting the pretrained representations to the defect detection task. As training continues, this structure becomes gradually clearer. At epoch 25, clusters corresponding to certain defect classes become more compact and begin to separate from the dominant no_defect class. By epoch 50, the clusters are more refined and several classes appear more clearly separated which suggests that fine-tuning improved the separability of the learned representations. However, some overlap between minority classes remains which is expected given the strong class imbalance in the dataset. Overall, the visualization indicates that fine-tuning progressively reshapes the pretrained DINO feature space to better reflect the structure of the defect classification task.

7.1.3 Final Model Performance

After selecting the final DINO configuration based on validation performance, the selected model is evaluated on the held-out test set to estimate generalization performance. The test set is not used during hyperparameter tuning or model selection. The summarized test results are shown in Table 7.4.

Table 7.4: Test results on the final DINO configuration using pretrained ImageNet weights.

Evaluation metric	ft	lin
Test balanced accuracy (%)	99.56	98.63
Test macro F1-score	0.9326	0.7699

Under fine-tuning, the final model achieves a test balanced accuracy of 99.56% and a macro F1-score of 0.9362, indicating strong and consistent performance across defect categories. The per-class results show that most defect classes are detected with near-perfect recall. The corresponding confusion matrix (Table 7.5, top) shows that most errors originate from the majority no_defect class being predicted as defect classes, while the minority defect categories are largely classified correctly.

Table 7.5: Confusion matrices for the final DINO configuration on the test set under fine-tuning (top) and linear probing (bottom).

Fine-tuning (ft)

True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	3891	11	12	17	8
Scrap Mark II	0	120	0	0	0
Scrap Mark	1	0	102	0	0
Cracks	0	0	0	34	0
Overfill	0	0	0	0	143

Linear probing (lin)

True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	3670	40	27	97	105
Scrap Mark II	0	120	0	0	0
Scrap Mark	0	0	103	0	0
Cracks	0	0	0	34	0
Overfill	0	0	0	0	143

In contrast, linear probing results in a noticeably lower macro F1-score of 0.7699, despite achieving a balanced accuracy of 98.63%. The confusion matrix (Table 7.5, bottom) reveals that a large portion of no_defect samples are misclassified as defect

classes, particularly as cracks and overfill. This behavior indicates that the frozen encoder representations are less suited for direct linear separation in this task.

Overall, the results demonstrate that fine-tuning substantially improves the class-wise balance of predictions for the DINO model. While linear probing achieves competitive balanced accuracy, fine-tuning leads to significantly better macro F1-score and more reliable class predictions, highlighting the benefit of adapting the pretrained representations to the downstream defect classification task.

7.1.4 Comparison With Supervised YOLO Baseline

To further evaluate the effectiveness of the pretrained DINO model, its performance is compared to the supervised YOLO baseline on the same original data split. As shown in Table 7.6, the YOLO model achieves a balance accuracy of 98.98% and macro F1-score of 0.9891, while the DINO model reaches 98.89% and 0.9114 respectively. Although the difference in balanced accruacy is marginal, the gap in macro F1-score is more pronounced which indicates that YOLO achieves more consistent class-wise performance. The confusion matrices for YOLO (see Table 5.4) and for DINO (see Table 7.7) provides further insight into this difference.

Table 7.6: Test results on the final DINO configuration and YOLO configuration using pretrained ImageNet weights on original data split (fine-tuning).

Evaluation metric	DINO	YOLO
Test balanced accuracy (%)	98.89	98.98
Test macro F1-score	0.9114	0.9891

Similar to the MAE model, most errors are concentrated in the "no defect" class which is occasionally misclassified as defect categories. In contrast, the defect classes themselves are classified with minimal confusion between the classes. This suggest that the DINO successfully captures discriminative features for defect types but struggles more with distinguishing between defective and non-defective samples.

Table 7.7: Confusion matrix for the DINO model pretrained on ImageNet weights on the original test set under fine-tuning.

	Fine-tuning (ft)				
True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	5161	13	22	27	36
Scrap Mark II	2	158	0	0	0
Scrap Mark	2	0	133	0	0
Cracks	0	0	0	48	0
Overfill	2	0	0	0	204

Compared to the MAE model, DINO achieves slightly higher balanced accuracy

which indicates stronger overall class separation. However, the lower macro F1-score relative to YOLO highlights that this separation is not equally robust across all classes. In particular, error concentrated in the majority class reduce the overall class-wise balance. The performance difference can be attributed to the fundamentally different training paradigms. While YOLO is trained fully supervised and directly optimized for the downstream classification task, DINO relies on self-supervised representation learning followed by fine-tuning. As a result, DINO must adapt general-purpose features to the specific defect classification task, which may limit its ability to fully match the class-wise consistency of a supervised model.

Overall, the results show that the pretrained DINO model achieves performance close to the supervised YOLO baseline in terms of balanced accuracy, while still lagging behind in macro F1-score. This indicates that DINO learns strong and transferable representations, but that fully supervised training remain advantageous for achieving optimal class-wise precision-recall balance.

7.2 DINO Model Trained From Scratch

The hyperparameter optimization results for the DINO model trained from scratch are presented in Table 7.8, where each sub-table corresponds to a specific hyperparameter evaluated during the tuning process. Model selection is primarily based on balanced accuracy and secondarily on macro F1-score.

In Table 7.8(a), the effect of online centering is evaluated. A value of $m = 0.99$ achieves the strongest overall performance across both balanced accuracy and macro F1-score, particularly for linear probing. Lower values such as $m = 0.9$ yield slightly more stable results for fine-tuning but do not reach the same peak performance. This differs from the pretrained setting, where a lower value was preferred, suggesting that stronger centering is beneficial when training from scratch. Table 7.8(b) examines the output dimension. An output dimension of 8192 provides the best balance across both evaluation protocols, achieving the highest balanced accuracy and macro F1-score. Smaller dimensions (4096) underperform slightly, while larger dimensions (up to 65536) do not yield consistent improvements. Compared to the pretrained case, where lower dimensions were sufficient, this indicates that higher capacity is required when no pretrained representations are available.

The number of epochs is evaluated in Table 7.8(c). Training for 15 epochs yields the best overall performance, improving both balanced accuracy and macro F1-score compared to 10 epochs. Increasing training to 20 epochs leads to a slight degradation, particularly in balance accuracy for fine-tuning, indicating potential overfitting. This behavior is consistent with the pretrained setting, although performance is overall lower. In Table 7.8(d), batch size is evaluated. A batch size of 32 achieves the most consistent results across both metrics and evaluation protocols. Smaller and larger batch sizes do not provide consistent improvements, aligning with observations from the pretrained model.

For the downstream optimization parameters, Table 7.8(e) shows that learning rate of 1.5e-4 yields the best performance. Higher learning rates significantly degrade

macro F1-score, particularly for fine-tuning, indicating unstable optimization. Similarly, Table 7.8(f) shows that weight decay of 0.05 provides the best balance between regularization and performance. Based on the sequential tuning procedure and the evaluation criteria described above, the final DINO configuration trained from scratch is summarized in Table 7.9.

Table 7.8: Hyperparameter tuning on DINO model trained from scratch

(a) Online centering

	Bal Acc		macro F1	
m	ft	lin	ft	lin
0.9	95.60	97.62	0.8052	0.7922
0.99	96.46	97.63	0.7694	0.8010
0.999	96.14	76.53	0.7348	0.4982

(b) Output dimension

	Bal Acc		macro F1	
dim	ft	lin	ft	lin
4096	91.66	97.37	0.7382	0.8038
8192	96.46	97.63	0.7694	0.8010
65536	93.71	98.03	0.7734	0.7836

(c) Epochs

	Bal Acc		macro F1	
Epochs	ft	lin	ft	lin
10	94.86	97.23	0.7267	0.7336
15	96.46	97.63	0.7694	0.8010
20	92.06	97.90	0.7956	0.8138

(d) Batch Size

	Bal Acc		macro F1	
BS	ft	lin	ft	lin
16	95.05	97.88	0.7256	0.8085
32	96.46	97.63	0.7694	0.8010
64	95.93	97.19	0.7990	0.7711

(e) Learning Rate - Downstream

	Bal Acc		macro F1	
LR	ft	lin	ft	lin
1.5e-4	96.46	97.63	0.7694	0.8010
3e-4	94.94	98.03	0.6581	0.8534
5e-4	93.85	98.14	0.6829	0.8719

(f) Weight Decay - Downstream

	Bal Acc		macro F1	
WD	ft	lin	ft	lin
0.01	90.37	97.68	0.7378	0.8068
0.05	96.46	97.63	0.7694	0.8010
0.1	95.63	97.55	0.7237	0.7904

Table 7.9: Final DINO configuration trained from scratch

Parameter	Value
Online centering	0.99
Output dimension	8192
Epochs	15
Batch size	32
Learning rate - Downstream	1.5e-4
Weight decay - Downstream	0.05

7.2.1 Label Efficiency

The label efficiency results for the DINO model trained from scratch are presented in Table 7.10. With only 1% labeled data, the model achieves 79.93% balanced accuracy for fine-tuning and 87.28% for linear probing. While linear probing achieves higher balanced accuracy under extreme data scarcity, its macro F1-score is substantially lower (0.5622 compared to 0.6606) which indicates poor class-wise balance. This mirrors the behavior observed in the pretrained setting but is more pronounced here. As the amount of labeled data increases, performance improved steadily. At 100% labeled data, the model reaches 95.29% balanced accuracy for fine-tuning and 97.60% for linear probing. However, macro F1-scores remain noticeably lower than in the pretrained case, particularly for fine-tuning.

Across all label fractions, linear probing consistently achieves higher balanced accuracy, while fine-tuning yields better macro F1-score. This indicates that linear probing captures overall class separation effectively, whereas fine-tuning improved precision-recall balance across classes. Overall, the results demonstrate reduced label efficiency compared to the pretrained DINO model by requiring more labeled data to achieve comparable performance.

Table 7.10: Label efficiency for the final DINO configuration trained from scratch, taken the average over three runs

	Bal Acc		macro F1	
Percent	ft	lin	ft	lin
1%	79.93	87.28	0.6606	0.5622
5%	89.00	94.44	0.7138	0.6288
10%	91.47	95.78	0.6927	0.6459
20%	92.26	96.49	0.7314	0.6984
100%	95.29	97.60	0.7295	0.7981

7.2.2 T-SNE for Different Epochs

To analyze how feature representations evolve during training, t-SNE visualizations of the CLS token embeddings are shown in Figure 7.2 for epochs 0, 5, 25 and 50.

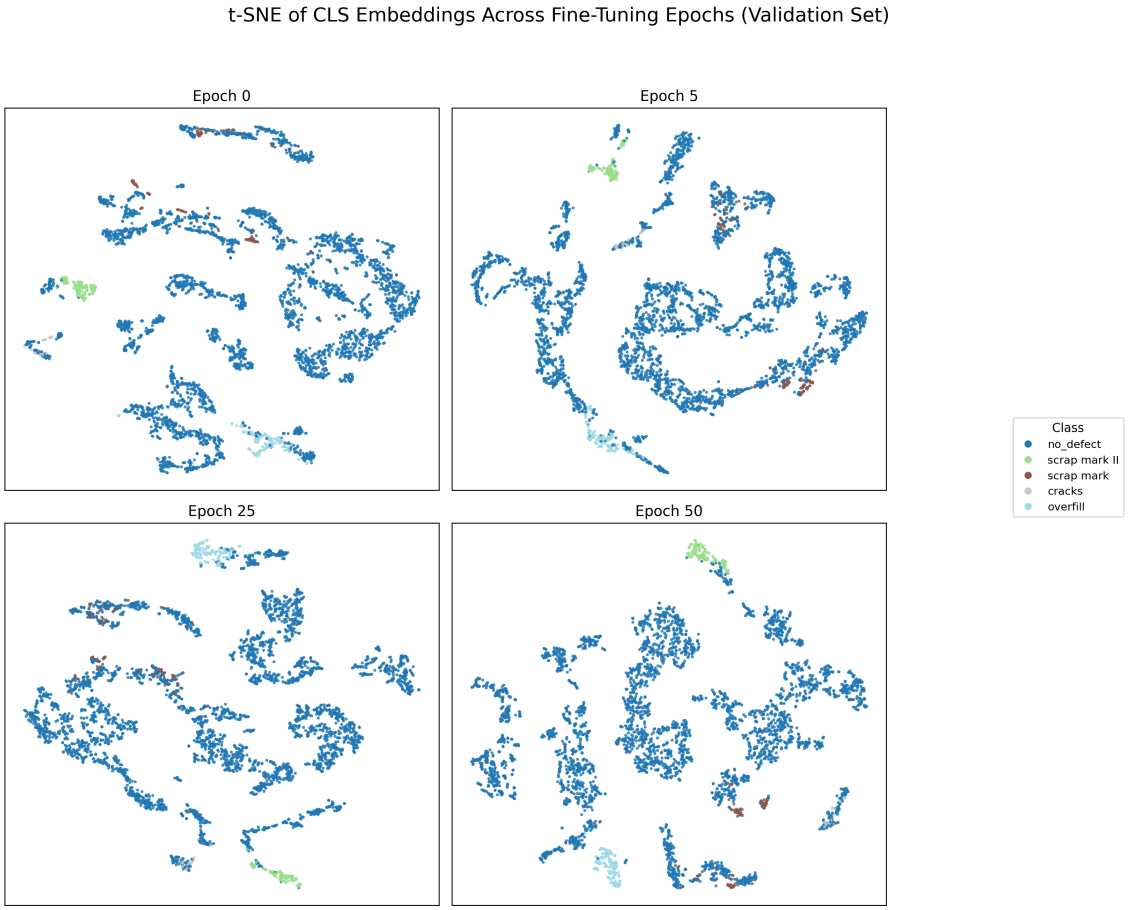


Figure 7.2: t-SNE visualization of the DINO model trained from scratch, trained on the validation set across different epochs.

At epoch 0, the embeddings appear largely unstructured with no clear separation between classes. This contrasts with the pretrained DINO model, where meaningful structure is already present due to prior training on ImageNet. By epoch 5, some early organization emerges but the clusters remain highly overlapping. At epoch 25, the embedding space becomes more structured with certain classes forming more compact regions. However, class separation is still limited compared to the pretrained case. At epoch 50, clusters become more defined and more compact which indicates that the model has learned more discriminative representations. However, the clusters remain less clearly separated than in the pretrained model and minority classes are still less distinguishable. Overall, the visualization shows that training from scratch requires more effort to structure the feature space and still results in weaker separability compared to pretrained representations.

7.2.3 Final Model Performance

The final model performance on the test set is presented in Table 7.9. The model achieves a balance accuracy of 93.38% for fine-tuning and 98.08% for linear probing, with corresponding macro F1-score of 0.6880 and 0.8000. Compared to the pretrained DINO model, this represents a noticeable drop in macro F1-score, particularly for fine-tuning, which indicates reduced class-wise balance.

Table 7.11: Test results on the final DINO configuration trained from scratch

Evaluation metric	ft	lin
Test balanced accuracy (%)	93.38	98.08
Test macro F1-score	0.6880	0.8000

The confusion matrices in Table 7.12 show that most errors originates from the "no defect" class being misclassified as defect categories. In contrast, the defect classes themselves are classified with high accuracy and minimal confusion between classes.

Table 7.12: Confusion matrices for the final DINO configuration trained from scratch on the test set under fine-tuning (top) and linear probing (bottom).

Fine-tuning (ft)

True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	3491	37	272	100	39
Scrap Mark II	0	120	0	0	0
Scrap Mark	21	0	82	0	0
Cracks	0	0	0	34	0
Overfill	2	0	0	0	141

Linear probing (lin)

True/Pred	No Defect	Scrap Mark II	Scrap Mark	Cracks	Overfill
No Defect	3703	23	140	36	37
Scrap Mark II	0	120	0	0	0
Scrap Mark	3	0	100	0	0
Cracks	0	0	0	34	0
Overfill	1	0	0	0	142

Compared to linear probing, fine-tuning improves macro F1-score which indicates better balance between precision and recall across classes. However, linear probing achieves higher balanced accuracy, suggesting stronger overall classification performance. Overall, while the DINO model trained from scratch is capable of learning useful representations, it remains inferior to the pretrained counterpart, particularly in terms of class-wise robustness and representation quality.

8

Discussion & Future Work

This chapter interprets the experimental results presented in Section 6 and Section 7 with respect to the research questions outlined in Section 1.2.

8.1 Discussion

Overall, the results of our thesis demonstrate that SSL-based approaches are highly competitive with, and in some cases superior to, fully supervised training. In particular, models leveraging SSL pretraining consistently achieve strong performance even when trained on limited labeled data which highlights the potential of SSL to reduce annotation requirements in industrial applications. Among the evaluated methods, MAE with ImageNet initialization yields the most consistent results across multiple evaluation metrics.

8.1.1 ImageNet vs. Scratch Initialization

From the results, a somewhat and consistent performance gap is observed between models that were initialized with ImageNet-pretrained weights and those that were trained from scratch. This trend is visible across most evaluated methods including both balanced accuracy and macro F1-score. For both MAE and DINO, ImageNet initialization leads to higher balanced accuracy and macro F1-score compared to scratch training. In contrast, models trained from scratch seem to consistently underperform. This can be seen in Figure 8.1 and 8.2, where the results for both MAE and DINO are presented as plots.

Previous literature seems to provide no general consensus regarding this topic. Some studies show results that align with our results from this thesis while other suggests that the opposite is true. The size of the performance gap suggests that the dataset used in this thesis is not sufficiently large or diverse to support effective representation learning from random initialization. A possible explanation for this behavior is that ImageNet pretraining provides a strong set of low- and mid-level visual features, such as edge detectors, texture representations and shape primitives, which are transferable across domains. Even though the target domain in this work differs from natural images, it is highly likely that these fundamental visual patterns remain relevant for defect detection as well. As a result, the models initialized with pretrained weights start from a more informative feature space and require less task-

specific adaptation. In contrast, the models trained from scratch must learn both general visual representations and task-specific features simultaneously, which might be particularly challenging given the limited size and imbalance of the dataset we use.

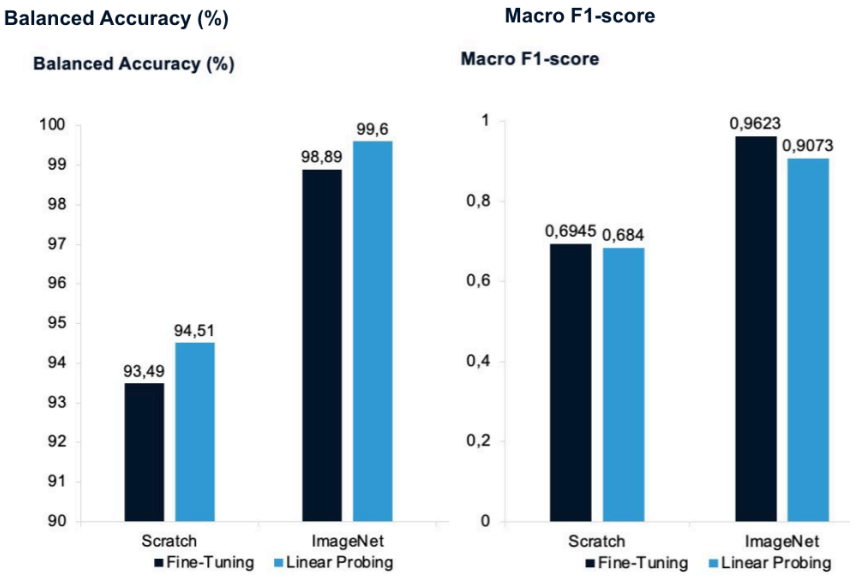


Figure 8.1: Overall test performance of MAE model

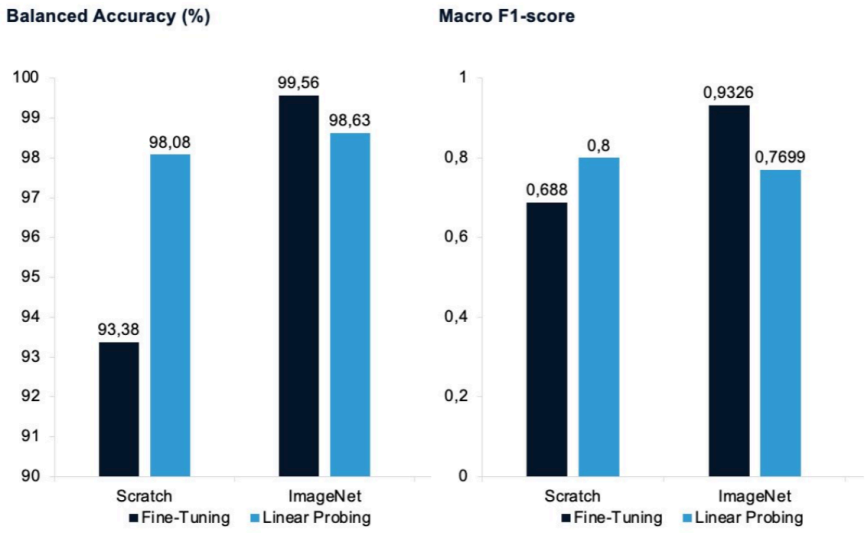


Figure 8.2: Overall test performance of DINO model

8.1.2 Linear Probing vs Fine-Tuning

The results reveal a consistent difference in behavior between linear probing and fine-tuning across all evaluated models. In general, linear probing achieves higher balanced accuracy, while fine-tuning yields higher macro F1-scores. This pattern is observed in both the full dataset results and the label efficiency experiments. This

can be seen in Figure 8.3 and 8.4, where plots over label efficiency are presented for both the MAE and DINO model.

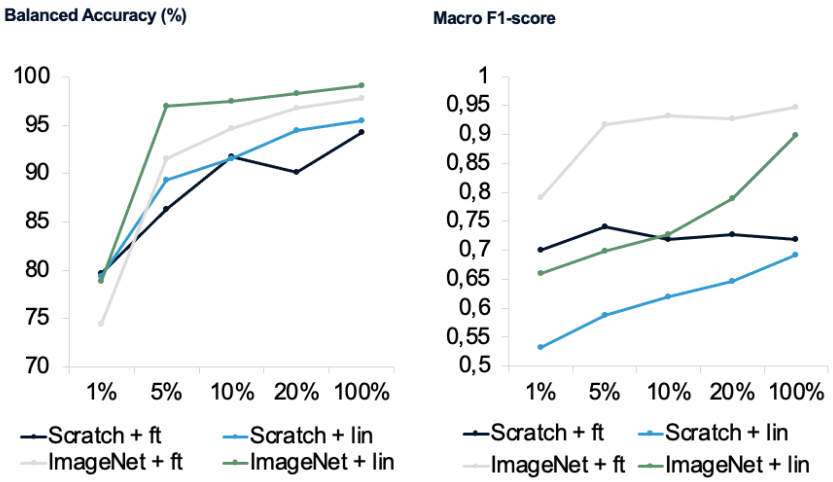


Figure 8.3: Label efficiency for MAE model

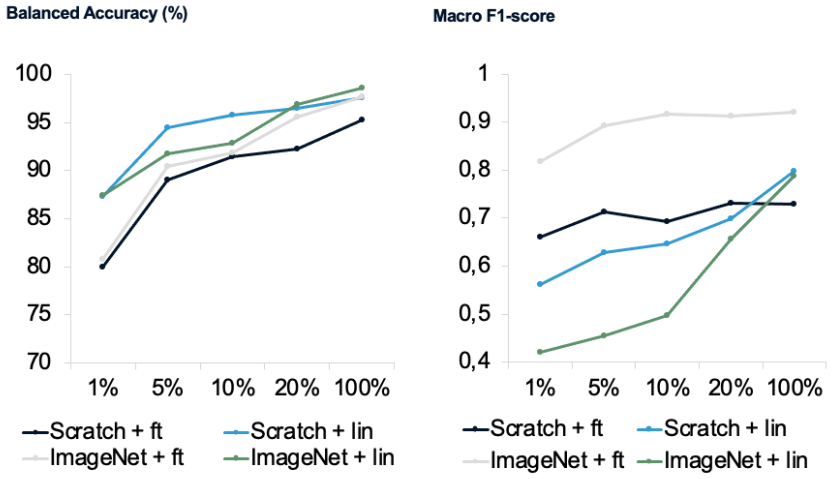


Figure 8.4: Label efficiency for DINO model

The higher balanced accuracy obtained with linear probing suggests that the pre-trained representations are already well-structured and allow for effective class separation using only a linear classifier. However, balanced accuracy alone does not fully capture performance across all classes. Fine-tuning improves macro F1-score, which places greater emphasis on minority classes. This indicates that fine-tuning allows the model to better adapt to class imbalance and capture more subtle differences between defect categories. The improvement in macro F1-score suggests that fine-tuning enhances performance on underrepresented classes, even if this sometimes comes at the cost of slightly reduced balanced accuracy. This trade-off highlights an important consideration for practical applications. Linear probing provides a simple and computationally efficient approach while still achieving strong performance,

whereas fine-tuning offers improved performance on minority classes, which may be critical in defect detection scenarios. So in applications where detecting rare defects is critical, fine-tuning may therefore be the preferred approach despite its higher computational cost.

8.1.3 SSL vs Supervised Learning

One of the main findings of this thesis is that SSL-based methods are able to match and in some cases outperform supervised learning, particularly under limited labeled data conditions. This is clearly demonstrated in the label efficiency results where the SSL methods achieve a relatively high performance even as the amount of labeled data is as low as 5% or 10%. These findings indicate that SSL is particularly advantageous in scenarios where labeled data is limited, which is often the case in industrial applications. By leveraging large amounts of unlabeled data during pretraining, SSL methods can significantly reduce the need for manual annotation.

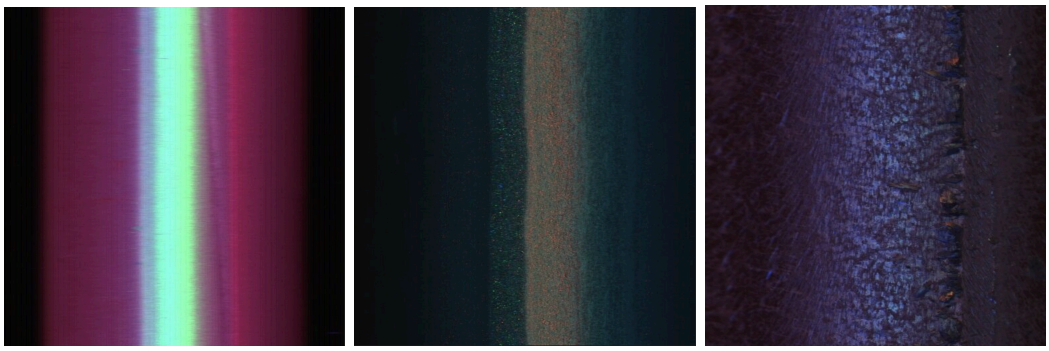
The comparison between YOLO and our SSL-based approaches shows that MAE and DINO generally achieve similar performance in terms of balanced accuracy while SINO shows somewhat lower performance in terms of macro f1-score. This indicates that YOLO performs better on minority classes and provides a more balanced performance across all the defect classes. One possible explanation for this difference could lie in the nature of the models. YOLO is specifically designed for object detection and is optimized for precise localization and classification of defects. This may give it an advantage in capturing fine-grained details and distinguishing between visually similar defect types. In contrast, MAE and DINO are primarily designed for representation learning rather than task-specific optimization.

Overall, the results indicate that fully supervised models still achieve the highest performance when large amounts of labeled data are available. This is expected since supervised learning directly optimizes the downstream objective using explicit annotations and can therefore exploit the full information contained in the labeled dataset. However, the experiments also demonstrate that SSL methods provide a more cost-effective and scalable alternative in industrial environments where annotation is expensive and time-consuming. By using large volumes of unlabeled production data during pretraining, SSL reduces the dependence on manual labeling while still achieving competitive downstream performance. This suggests that SSL may offer a more practical long-term solution for industrial defect classification systems, where unlabeled data is plentiful but high-quality annotations are scarce.

8.1.4 Misclassified Images

With the support of domain specialists at Swerim, all images being misclassified by the models were reviewed in order to identify any patterns in what the models struggled with in the classification task. The analysis of misclassified images reveals several recurring patterns that help explain the limitations observed in the results. Interestingly, the analysis revealed that a significant portion of the observed errors cannot be solely attributed to model limitations. Instead, several underlying factors

related to data quality and image characteristics contribute to the misclassifications. A common issue discovered from the results is the confusion between visually similar defect classes, as well as the tendency to misclassify minority class samples as the dominant no-defect class. These patterns are consistent with the class imbalance that is present in the dataset and are reflected in the differences between balanced accuracy and macro F1-score observed in results. Another recurring pattern involves visually ambiguous samples. For example, surface oxidation and other irregularities can resemble defects, even though they do not correspond to actual damage. These cases are inherently difficult to classify, as the visual distinction between defect and non-defect is subtle even for human experts. As a result, some model predictions that appear incorrect may in fact be reasonable given the visual input.



(a) Missclassification due to over-exposed image (reflection on top of material)

(b) Missclassification due to optical illusion of defect (shadow)

(c) Missclassification due to some surface oxidation (not defect)

Figure 8.5: Example of non-defect images that, by the models were, incorrectly classified as defects.

An additional challenge observed during the analysis of misclassified images relates to image quality, such as variability in lighting conditions across the dataset. Several misclassified samples were found to be overexposed, resulting in a loss of visible surface detail. In such cases, it becomes difficult for both the model and human observers to identify meaningful features. Similarly, defects located near the edges of the rod are often captured under poor lighting conditions, making them less distinguishable from the background. These factors reduce the amount of usable information available to the model and contribute to misclassification. Importantly, many of these problematic samples appear consistently across different models and training strategies, including both MAE and DINO, as well as linear probing and fine-tuning. This suggests that the errors may not primarily be due to the choice of model but rather stem from inherent challenges in the dataset.

Overall, this analysis highlights that model performance is strongly influenced by data quality. A potential approach to mitigate this issue would be to incorporate data augmentation techniques that simulate varying lighting conditions, such as brightness and contrast adjustments. By exposing the model to a wider range of illumination scenarios during training, it may be possible to improve robustness and enable more consistent detection of defects regardless of lighting conditions. Alternatively, preprocessing techniques aimed at normalizing image brightness could

be explored to reduce variability across samples. These approaches may help the model focus on structural features of the defects rather than being influenced by differences in lightning conditions.

8.2 Achieving Research Goals

This thesis investigated the effectiveness of self-supervised learning (SSL) for industrial surface defect classification, with a focus on evaluating different SSL approaches, training strategies and initialization schemes under realistic industrial conditions.

The first research question examined whether self-supervised learning on unlabeled data can improve defect classification performance compared to standard supervised baselines. The results show that while SSL methods such as MAE and DINO achieve strong performance and learn highly transferable representations, they do not surpass the fully supervised YOLO baseline when sufficient labeled data is available. In particular, the supervised model achieves high macro F1-score which indicates more consistent class-wise performance. However, the relatively small performance gap demonstrates that SSL remains a competitive alternative, especially in scenarios where labeled data is limited or costly to obtain.

For the second research question, the comparison between MAE and DINO shows that both methods are effective under industrial conditions characterized by class imbalance and subtle texture variations. However, differences in performance and behavior can be observed depending on training configuration and data availability. Overall, both approaches are shown to be suitable for the task which highlights that both generative and distillation-based SSL methods can successfully learn transferable representations in this specific domain.

The third research question addressed to what extent self-supervised learning approaches remain effective when labeled data is scarce. The results demonstrate that both MAE and DINO maintain relatively strong performance even at low label fractions, indicating that the learned representations are meaningful and transferable with limited supervision. At very low levels of labeled data, linear probing achieves competitive performance, suggesting that the pretrained features already capture useful structure in the data. However, as the amount of labeled data increases, fine-tuning consistently outperforms linear probing which highlights the importance of adapting the full model to the downstream task in order to achieve optimal performance. Across all label fractions, DINO shows stronger performance than MAE, indicating that its representations are more robust in low-data regimes. Nevertheless, performance improves steadily with increasing amounts of labeled data which shows that while self-supervised learning provides a strong foundation, access to additional labeled data remains important. Overall, these findings demonstrate that self-supervised learning approaches remain effective under limited labeled data conditions, but benefit significantly from both fine-tuning and increased supervision.

8.3 Future Work

While this thesis have demonstrated the potential of SSL approaches for industrial defect classification, several directions for future work remain. First, our work focuses on two representative SSL methods, MAE and DINO, and although these methods capture two distinct areas of the SSL domain, future research could explore a broader range of approaches. In particular, methods such as SimCLR and BYOL have shown strong potential for providing insights into how different learning objectives influence performance in earlier literature on defect classification in industrial domains.

Second, this thesis is limited to image-level classification. A potential extension could be to investigate localization-based tasks such as object detection or semantic segmentation. This would enable identification of the spatial extent and precise location of defects. Such an approach could be informative in the industrial inspection setting as understanding the defect position and size could be of advantage for downstream decision making. In this project it has also become evident that the model struggles when defects appear in certain parts of the images.

Another important direction is the exploration of data augmentation strategies tailored to industrial data. While this thesis employs carefully chosen augmentations, SSL methods are known to be sensitive to such choices. Future work could investigate the impact of augmentations such as Gaussian blur, brightness and contrast variations, and other transformations designed to improve robustness to noise and varying imaging conditions, while ensuring that subtle defect patterns are preserved.

Furthermore, the impact of model architecture and scaling remains an open question. This study focuses on ViTs, but alternative architectures such as hierarchical Transformer models, including Swin Transformers, may offer improved performance by capturing both local and global features more effectively. Evaluating such architectures could provide further insights into how structural inductive biases influence defect detection performance.

Finally, the experiments are conducted on a single industrial dataset provided by Swerim. Evaluating the proposed approaches on multiple datasets or across different industrial domains would provide a better understanding of generalization capabilities and robustness. Moreover, the dataset used in this study exhibits significant class imbalance, which poses challenges for reliable model evaluation and performance on minority defect classes. Future work could investigate more advanced techniques for handling class imbalance such as data resampling, class-balanced loss functions or synthetic data generation in order to improve performance on underrepresented defect categories.

Overall, these potential future directions highlight several opportunities to further develop and refine SSL methods for industrial defect detection, both from a methodological and application-oriented perspective.

Bibliography

[1] A. Poddar, M. Sureka, R. Chajtterjee, M. K. Gourisaria, S. Jain, and D. S. Roy, "Industrial surface defect detection using machine leaning approach," in *2025 7th International Conference on Information Systems and Computer Networks (ISCON)*, 2025, pp. 1–7. DOI: 10.1109/ISCON65210.2025.11341674.

[2] H. Wei, L. Zhao, R. Li, and M. Zhang, "Rfaconv-cbm-vit: Enhanced vision transformer for metal surface defect detection," *The Journal of Supercomputing*, vol. 81, no. 1, p. 155, 2025.

[3] H. He et al., "Industrial image anomaly detection method based on improved mae," in *2023 28th International Conference on Automation and Computing (ICAC)*, 2023, pp. 1–6. DOI: 10.1109/ICAC57885.2023.10275293.

[4] S. Hu, X. Ma, Y. Zhang, and W. Xu, "Application of self-supervised learning in steel surface defect detection," in *Journal of Machine Intelligence*, OAE Publishing, 2025. [Online]. Available: https://www.oaepublish.com/articles/jmi.2025.21.

[5] J. Yang, R. Xu, Z. Qi, and Y. Shi, "Visual anomaly detection for images: A systematic survey," *Procedia Computer Science*, vol. 199, pp. 471–478, 2022, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2022.01.057. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050922000576.

[6] J. Zhang, L. Yang, S. M. Sajjadi Mohammadabadi, and F. Yan, "A survey on self-supervised learning: Recent advances and open problems," in *Neurocomputing*, Elsevier, 2025, p. 131 409. DOI: 10.1016/j.neucom.2025.131409. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231225020818.

[7] J. Gui et al., "A Survey on Self-Supervised Learning: Algorithms, Applications, and Future Trends," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, IEEE, 2024, pp. 1–23. DOI: 10.1109/TPAMI.2023.3330154. [Online]. Available: https://ieeexplore.ieee.org/document/10304281.

[8] A. Jaiswal, A. R. Babu, A. Zadeh, D. Banerjee, and F. Makedon, "A Survey on Contrastive Self-Supervised Learning," in *Technologies*, MDPI, 2020, p. 41. DOI: 10.3390/technologies9010041. [Online]. Available: https://www.mdpi.com/2227-7080/9/1/41.

[9] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022, pp. 15 979–15 988. DOI: 10.1109/CVPR52688.2022.01553.

[10] M. Caron et al., "Emerging properties in self-supervised vision transformers," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 9630–9640. DOI: 10.1109/ICCV48922.2021.00951.

[11] W. Jin, F. Guo, Q. Wu, and L. Zhu, "Incremental self-supervised learning based on transformer for anomaly detection and localization," *Engineering Applications of Artificial Intelligence*, vol. 160, p. 111 978, 2025, ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2025.111978. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197625019864.

[12] A. Chabrol, "Automated steel defect detection using machine learning," Swerim, Stockholm, Sweden, Tech. Rep., 2025.

[13] X. Chen and K. He, "Exploring simple siamese representation learning," in *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 15 745–15 753. DOI: 10.1109/CVPR46437.2021.01549.

[14] I. Goodfellow, Y. Bengio, and A. Courville, "Deep Learning," in *MIT Press*, MIT Press, 2016, pp. 1–775. [Online]. Available: https://www.deeplearningbook.org.

[15] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer New York, NY, 2006.

[16] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2010. DOI: 10.1109/TKDE.2009.191.

[17] Y. LeCun, "A Path Towards Autonomous Machine Intelligence," in *OpenReview*, OpenReview, 2022, pp. 1–52. DOI: 10.48550/arXiv.2203.16027. [Online]. Available: https://openreview.net/forum?id=BZ5a1r-kVsf.

[18] T. Uelwer, "A Survey on Self-Supervised Methods for Visual Representation Learning," in *Machine Learning*, Springer, 2025, pp. 1–45. DOI: 10.1007/s10994-024-06573-8. [Online]. Available: https://link.springer.com/article/10.1007/s10994-024-06573-8.

[19] S. Albelwi, "Survey on Self-Supervised Learning: Auxiliary Pretext Tasks and Contrastive Learning Methods in Imaging," in *Entropy*, MDPI, 2022, pp. 1–29. DOI: 10.3390/e24091303. [Online]. Available: https://www.mdpi.com/1099-4300/24/9/1303.

[20] X. Liu et al., "Self-supervised learning: Generative or contrastive," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 1, pp. 857–876, 2023. DOI: 10.1109/TKDE.2021.3090866.

[21] A. van den Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," *ArXiv*, vol. abs/1807.03748, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:49670925.

[22] V. Rani, S. T. Nabi, M. Kumar, A. Mittal, and K. Kumar, "Selfsupervised learning: A succinct review," in *Archives of Computational Methods in Engineering*, vol. 30, 2023, pp. 2761–2775. DOI: https://doi.org/10.1007/s11831-023-09884-2.

[23] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *Proceedings of the 37th*

International Conference on Machine Learning, ser. Proceedings of Machine Learning Research, vol. 119, PMLR, 13–18 Jul 2020, pp. 1597–1607.

[24] G. Alain and Y. Bengio, *Understanding intermediate layers using linear classifier probes*, 2018. arXiv: 1610.01644 [stat.ML]. [Online]. Available: https://arxiv.org/abs/1610.01644.

[25] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds., vol. 27, Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/file/532a2f85b6977104bc93f8580abbb330-Paper.pdf.

[26] A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[27] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://openreview.net/forum?id=YicbFdNTTy.

[28] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou, "Training data-efficient image transformers amp; distillation through attention," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., vol. 139, PMLR, 2021, pp. 10 347–10 357. [Online]. Available: https://proceedings.mlr.press/v139/touvron21a.html.

[29] M. Raghu, T. Unterthiner, S. Kornblith, C. Zhang, and A. Dosovitskiy, "Do vision transformers see like convolutional neural networks?" In *Advances in Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021. [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/652cf38361a209088302ba2b8b7f51e0-Paper.pdf.

[30] M. Naseer, K. Ranasinghe, S. Khan, M. Hayat, F. S. Khan, and M.-H. Yang, "Intriguing properties of vision transformers," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, ser. NIPS '21, 2021.

[31] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 9726–9735. DOI: 10.1109/CVPR42600.2020.00975.

[32] J.-B. Grill et al., "Bootstrap your own latent a new approach to self-supervised learning," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20, 2020.

[33] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised Learning of Visual Features by Contrasting Cluster Assignments," in *Advances in Neural Information Processing Systems*, Curran Associates, 2020, pp. 9912–9924. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/e946209592b4aa848ea3f5e02c53e2d3-Abstract.html.

[34] P. H. Richemond et al., *Byol works even without batch statistics*, 2020. arXiv: 2010.10241 [stat.ML]. [Online]. Available: https://arxiv.org/abs/2010.10241.

[35] K. He, R. Girshick, and P. Dollar, "Rethinking imagenet pre-training," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 4917–4926. DOI: 10.1109/ICCV.2019.00502.

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[37] Y. Chen, H. Peng, L. Huang, J. Zhang, and W. Jiang, "A novel mae-based self-supervised anomaly detection and localization method," *IEEE Access*, vol. 11, pp. 127 526–127 538, 2023.

[38] X. Wan, X. Zhang, and L. Liu, "An improved vgg19 transfer learning strip steel surface defect recognition deep neural network based on few samples and imbalanced datasets," *Applied Sciences*, vol. 11, no. 6, 2021, ISSN: 2076-3417. DOI: 10.3390/app11062606. [Online]. Available: https://www.mdpi.com/2076-3417/11/6/2606.

[39] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, no. 2, Mar. 2021, ISSN: 0360-0300. DOI: 10.1145/3439950. [Online]. Available: https://doi.org/10.1145/3439950.

[40] Q. Wan, L. Gao, and X. Li, "Logit inducing with abnormality capturing for semi-supervised image anomaly detection," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–12, 2022. DOI: 10.1109/TIM.2022.3205674.

[41] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *International Conference on Learning Representations (ICLR)*, OpenReview, 2019, pp. 1–13. DOI: 10.48550/arXiv.1711.05101. [Online]. Available: https://openreview.net/forum?id=Bkg6RiCqY7.

[42] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1412.6980.

[43] M. Zabin, A. N. B. Kabir, M. K. Kabir, H.-J. Choi, and J. Uddin, "Contrastive self-supervised representation learning framework for metal surface defect detection," in *Journal of Big Data*, London, UK: SpringerOpen, 2023, p. 145. DOI: 10.1186/s40537-023-00827-z. [Online]. Available: https://doi.org/10.1186/s40537-023-00827-z.

[44] L. van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of Machine Learning Research*, vol. 9, no. 86, pp. 2579–2605, 2008. [Online]. Available: http://jmlr.org/papers/v9/vandermaaten08a.html.

[45] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, *Yolov4: Optimal speed and accuracy of object detection*, 2020. arXiv: 2004.10934 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2004.10934.

A

Appendix 1

A.1 Implementation Details

This section provides additional implementation details to support reproducibility of the experiments.

All models were implemented in Python using PyTorch. The experiments were conducted on GPU-enabled compute nodes, where training time varied depending on the protocol. Pretraining using MAE typically required approximately 2 to 3 hours for 50 epochs, while fine-tuning could take up to 5-6 hours for larger configurations. Unless otherwise stated, the following default hyperparameters were used for downstream training:

Table A.1: Default Hyperparameters

Parameter	Value
Optimizer	AdamW
Epochs	50
Batch size	32
Learning rate	1.5e-4
Weight decay	0.05

All experiments were conducted using a fixed train/validation/test split to ensure comparability across configurations.

A.2 Dataset and Data Splits

The dataset used in this work consists of labeled defect images organized into predefined training, validation and test splits. In addition to the original folder-based structure, CSV files were created to explicitly define the data splits used during training and evaluation.

Each CSV file contains the file paths and corresponding labels for the images. These CSV-based splits were used consistently across all experiments to ensure reproducibility and to allow flexible data loading independent of the original folder structure.

The same dataset was used for both pretraining and downstream tasks. However, for downstream evaluation, only the labeled portion of the data was utilized, following the standard protocol for self-supervised learning.

A.3 Example Training Commands

This section provides example commands used for running the experiments.

A.3.1 index_dataset

Index all data from the first folders and make it into index.csv:

```
python -u -m src.utils.index_dataset
--data_root <data_root>
--out_dir <output_dir>
```

A.3.2 make_splits and make_org_split_csv

Make splits and create train.csv, val.csv and test.csv, either with our chosen split or with the original folder split:

```
python -u -m src.utils.make_splits
--csv <index.csv>
--out_dir <out_dir>
--train_ratio 0.70
--val_ratio 0.15
```

```
python -u -m src.utils.make_org_split_csv
--data_root <data_root>
--out_dir <output_dir>
```

A.3.3 preprocess_checks

Check for faulty paths:

```
python -u -m src.utils.preprocess_checks
--train_csv data/splits/train.csv
--val_csv data/splits/val.csv
--test_csv data/splits/test.csv
--data_root <data_root>
--out data/index/bad_files.csv
```

A.3.4 mae_pretext

Pretraining (MAE, ImageNet):

```
python -u -m src.models.mae_pretext
--train_csv data/splits/train.csv
```

```
--data_root <data_root>
--out_dir <output_dir>
--init imagenet_mae
--epochs 50
--batch_size 32
--lr 1.5e-4
```

A.3.5 dino_pretext

Pretraining (DINO, ImageNet):

```
python -u  -m src.models.dino_pretext
--train_csv data/splits/train.csv
--data_root <data_root>
--out_dir <out_dir>
--epochs 15
--batch_size 32
--num_workers 2
--out_dim 8192
--center_momentum 0.9
--init imagenet
--teacher_temp_final 0.05
--wd_end 0.2
```

A.3.6 train_downstream_vit

Downstream training (MAE, fine-tuning):

```
python -u -m src.training.train_downstream_vit
--train_csv data/splits/train.csv
--val_csv data/splits/val.csv
--data_root <data_root>
--out_dir <output_dir>
--init mae_pretext
--mae_dir <mae_dir>
--epochs 50
--batch_size 32
--lr 1.5e-4
--weight_decay 0.05
--optimizer adamw
--protocol finetune
--label_pct 100
//For test result:
--save_misclassified
--miscls_csv <miscls_csv>
--test_csv data/splits/test.csv
```

Downstream training (DINO, fine-tuning):

```
python -u -m src.training.train_downstream_vit
--train_csv data/splits/train.csv
--val_csv data/splits/val.csv
--data_root <data_root>
--out_dir <output_dir>
--init dino_pretext
--dino_dir <dino_dir/student_backbone.pth>
--epochs 50
--batch_size 32
--lr 1.5e-4
--weight_decay 0.05
--optimizer adamw
--protocol finetune
--label_pct 100
//For test result:
--save_misclassified
--miscls_csv <miscls_csv>
--test_csv data/splits/test.csv
```

Downstream training (Baseline ImageNet, fine-tuning):

```
python -u -m src.training.train_downstream_vit
--train_csv data/splits/train.csv
--val_csv data/splits/val.csv
--data_root <data_root>
--out_dir <output_dir>
--init imagenet_vit
--epochs 50
--batch_size 32
--lr 1.5e-4
--weight_decay 0.05
--optimizer adamw
--protocol finetune
--label_pct 100
//For test result:
--save_misclassified
--miscls_csv <miscls_csv>
--test_csv data/splits/test.csv
```

A.3.7 tsne_epochs

t-SNE plot:

```
python -u -m src.training.tsne_epochs
--ckpt_dir <folder with epoch_000.pt, epoch_005.pt...>
--val_csv data/splits/val.csv
--data root <data_root>
--out_dir <out_dir>
```

```
--batch_size 32
--out_name <out_name>
--num_workers 0
--title <title>
```

These commands are representative examples. Additional variations were used to evaluate different hyperparameters and configurations as described in the main text.