FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Using DINOv2 for Road Damage Anomaly Detection

Luís Pereira dos Santos Conceição

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Nuno Alexandre Cruz

Second Supervisor: Luís Filipe Teixeira

October 31, 2024

Resumo

Com a necessidade crescente de transportes seguros e eficientes, a deteção de anomalias em ambientes rodoviários tornou-se essencial para a manutenção proactiva das infra-estruturas e a prevenção de acidentes. Esta tese explora a aplicação do DINOv2, um modelo fundacional baseado numa arquitetura de transformadores visuais estado de arte, para a deteção de anomalias em ambientes rodoviários. Investiga o desempenho do DINOv2 em comparação com outros backbones, especificamente o ResNet18, utilizando o MVTec AD e dois conjuntos de dados distintos de danos em estradas. Procura fornecer conhecimentos sobre a deteção de anomalias em ambientes não controlados e a utilização do Grounded SAM para segmentar as regiões em cada imagem, eliminando o fundo desnecessário.

Os resultados indicam que o DINOv2 demonstra capacidades claras de deteção de anomalias no conjunto de dados MVTec AD. No entanto, o estudo conclui que, embora o DINOv2 supere o ResNet18, as melhorias de desempenho não são uniformemente significativas em comparação com outros modelos. A análise revela que a qualidade dos dados, incluindo questões como fundos de imagem inconsistentes e a escala relativa das anomalias em relação ao tamanho da imagem, desempenha um papel crucial na eficácia destes modelos.

Além disso, o estudo destaca a importância da qualidade dos dados no treino e avaliação de sistemas de deteção de anomalias. A variabilidade e a complexidade das condições rodoviárias no mundo real apresentam desafios significativos que devem ser abordados em trabalhos futuros. Os resultados contribuem para a compreensão das capacidades dos transformadores visuais em ambientes dinâmicos.

Palavras-chave: Deteção de Anomalias, DINOv2, ResNet18, MVTec AD, Danos em Estradas, Grounded SAM.

Abstract

With the growing need for safe and efficient transportation, anomaly detection in road environments has become essential for proactive infrastructure maintenance and accident prevention. This thesis explores the application of the DINOv2, a visual foundation model based on a state-of-the-art visual transformer architecture, as a backbone for anomaly detection in road environments. The research investigates the performance of DINOv2 compared to other backbones, specifically ResNet18, using MVTec AD and two distinct road damage datasets. It tries to give insights into Anomaly Detection in uncontrolled environments and the utilization of grounded SAM to segment the regions of interest in each image to eliminate the background.

The results indicate that DINOv2 demonstrates precise anomaly detection capabilities in the MVTec AD data set. However, the study finds that while DINOv2 surpasses ResNet18, the performance gains are not uniformly significant compared to other backbones. The analysis reveals that data quality, including issues like inconsistent backgrounds and the relative scale of anomalies to image size, plays a critical role in the effectiveness of these models.

Furthermore, the study underscores the importance of data quality in training and evaluating anomaly detection systems. The variability and complexity of real-world road conditions present significant challenges that must be addressed in future work. The findings contribute to the understanding of visual transformer capabilities in dynamic environments.

Keywords: Anomaly Detection, DINOv2, Resnet18, MVTec AD, Road Damage, Grounded SAM.

Sustainable Development Goals

SDG	Target	Contribution	Performance Indicators and Metrics
9	9.1	Development of advanced anomaly detection technologies to improve road infrastructure maintenance and safety.	Number of roads monitored with the new technology, reduction in incidents due to undetected defects, response time for repairs after anomaly identification.
11	11.2	Improvement in urban mobility through efficient infrastructure maintenance, contributing to safe, accessible, and sustainable transport systems.	Proportion of road incidents avoided, assessment of road accessibility and safety, community feedback on road conditions.
13	13.1	Reduction of negative climate impacts by optimizing maintenance operations, decreasing the need for emergency repairs, and reducing associated carbon emissions.	Reduction in carbon emissions due to fewer emergency maintenance interventions, energy efficiency of maintenance processes.

Table 1: Project Contributions to the SDGs

Agradecimentos

Gostaria de expressar os meus agradecimentos a todos aqueles que me apoiaram durante esta dissertação. Agradeço à Ocean Infinity pela oportunidade de desenvolver este trabalho, assim como pelo apoio e recursos disponibilizados, em especial ao Filipe Ferreira pela sua constante disponibilidade e ajuda ao longo destes meses. Um especial agradecimento aos meus orientadores, o professor Nuno Alexandre Cruz e o professor Luís Filipe Teixeira pela disponibilidade em orientar esta dissertação e por me guiarem durante a sua realização. Agradeço também à minha família e amigos, pelo apoio, e por me proporcionarem as condições de realizar o meu percurso académico.

Luís Pereira dos Santos Conceição

Contents

List of Figures

List of Tables

Abbreviations and Symbols

AUROC	Area Under the Receiver Operating Characteristic Curve
CUDA	Compute Unified Device Architecture
OpenCV	Open Source Computer Vision Library
RDD	Road Damage Detector
SAM	Segment Anything Model
SDG	Sustainable Development Goals
SK	Sinkhorn-Knopp
TIMM	PyTorch Image Models
VS Code	Visual Studio Code

Chapter 1

Introduction

1.1 Motivation

In recent years, the application of anomaly detection in various industries has garnered significant attention due to its potential to enhance operational efficiency and safety. Anomaly detection is a critical task in domains ranging from industrial manufacturing to infrastructure monitoring, where identifying deviations from normal patterns can prevent failures, reduce downtime, and improve overall safety. Roads are particularly vital among critical infrastructure systems as they facilitate transportation and economic activities, making maintenance crucial.

To this end, this research uses the DINOv2 backbone, a state-of-the-art visual transformer architecture and foundation model, to enhance anomaly detection in road imagery and address the real-world challenges of uncontrolled environments.

1.2 Context

Anomaly detection has been gaining popularity recently due to its potential to assist in various industries such as manufacturing [41], finance [19], medicine [8], and cybersecurity [40]. This is because of its significant advantages over common methods of detecting anomalies, which usually involve manual monitoring and predefined rules. These methods can be time-consuming and prone to errors. Additionally, they rely on human intervention to identify and understand anomalies, which limits their effectiveness and scalability, particularly as the scale of data increases due to technological advancements. Automated anomaly detection offers the following benefits:

- **Scalability:** Automated systems can process large volumes of data in real time, far beyond human capabilities. This scalability is essential to constantly monitor extensive road networks.
- **Accuracy and Consistency:** Machine learning models can learn complex patterns, identify subtle anomalies, and ensure consistent monitoring without fatigue, reducing the likelihood of missed detections.

- **Speed:** Automated systems can quickly detect and respond to anomalies, crucial for time-sensitive situations such as road hazards or accidents.
- **Cost-Effectiveness:** Reducing the need for manual monitoring can decrease operational costs and free up human resources for more strategic tasks.
- **Adaptability:** Advanced models can adjust to new patterns over time, enhancing their performance as they encounter more data, while traditional methods often require manual updates to rules and thresholds.

1.2.1 Benefits of Anomaly Detection in Roads

It offers significant benefits in road maintenance.

On Roads:

- **Safety:** Early identification of anomalies such as accidents, obstacles, or hazardous road conditions can immediately prompt action, preventing further incidents and ensuring the safety of road users.
- **Infrastructure Monitoring:**Monitoring the condition of road infrastructure through anomaly detection can help with timely maintenance, preventing deterioration, and ensuring longevity.

1.2.2 Foundation Models

Foundation models like DINOv2 are pre-trained on large and diverse datasets, which allows them to capture a broad range of visual features that can generalize across multiple applications. This generalizability is especially useful in the detection of road anomalies, as it allows the model to recognize familiar and novel patterns in varied and unpredictable conditions, such as changes in lighting, weather and vehicles' or debris occlusions. Building on the robust and transferrable representations that DINOv2 provides, this research aims to improve detection accuracy and resilience in the complex setting of road infrastructure maintenance.

1.3 Objective

The primary objective of this thesis is to explore the effectiveness of the DINOv2 backbone in enhancing anomaly detection within the context of road environments; it involves a comparative analysis with traditional convolutional neural network (CNN) backbones, particularly ResNet-18, to evaluate their respective performances in detecting anomalies in road images. The research aims to:

1. **Measure the Performance of DINOv2:** Assess the performance differences between DINOv2 and other previously implemented backbones.

2. **Develop and Utilize Road Damage Datasets:** Create comprehensive datasets representing various road conditions, including normal and damaged scenarios. These datasets will include diverse environmental factors, such as weather conditions and lighting variations, to ensure the models' robustness and generalizability and to test DINOv2.

1.4 Expected Contributions

This thesis aims to contribute to the field of anomaly detection by providing the following.

1. A comparative analysis of the effectiveness of DINOv2 versus other backbones, namely Resnet18.
2. A set of comprehensive road damage datasets that can serve as benchmarks for future research in this area.
3. Insights into applying visual transformers in anomaly detection tasks, particularly in dynamic and uncontrolled environments like roads.

The findings of this research are expected to offer valuable information both for the academic community and for practical applications, contributing to the development of more reliable and efficient road maintenance and monitoring systems.

Chapter 2

Literature Review

2.1 Overview of Anomaly Detection

According to [34], an anomaly is an observation that differs significantly from a particular concept of normality. It can also be referred to as an outlier or novelty and described as unusual, irregular, atypical, inconsistent, unexpected, rare, erroneous, faulty, fraudulent, malicious, unnatural, or simply strange, depending on the context. Anomaly detection (outlier or novelty detection) examines and identifies such unusual observations using data-based methods, models, and algorithms. Finding the correct type of anomaly detection model for the task requires carefully acknowledging the different benefits each brings.

Each algorithm approaches problems differently, and finding the best solution requires carefully evaluating existing models. Existing benchmarks [17], [34], [39], [25] offer valuable insights into the performance of popular and advanced models, aiding in the selection of the suitable model for the task.

2.1.1 Supervision

Anomaly detection methods can be broadly classified according to the type of supervision involved in the training process: **supervised**, **semi-supervised**, and **unsupervised**. Within these categories, there are various types of architecture, each with its advantages and disadvantages [17], [39], [34], [25].

In **supervised** methods, the model requires completely labeled data for training, akin to regular image classification. This is the most uncommon paradigm due to two main factors: The labeling process often involves significant expenses and time-consuming processes.; Labeled anomalies rarely fully represent the high variability in real-world anomalies that may not be present in training. [34]

According to [34], **unsupervised** methods use only unlabeled data and assume that the data is mostly normal with some error and noise. These methods are instrumental in real-world situations where labeled data may be hard to come by and expensive. They are more widely applicable but may result in errors due to the lack of guidance.

Semi-supervised methods use a combination of a small amount of labeled data n and a large amount of unlabeled data m where $n \gg m$. This approach can improve the model's ability to predict anomalies while requiring less labeled data from experts than the supervised method.

Based on the characteristics of each paradigm described above, it is preferable to use a semi-supervised or unsupervised approach when selecting a model for the task. This approach allows for better generalization in real-world applications and high variability of uncontrolled environments while reducing the need for expert-labeled data.

2.1.2 Existing Models

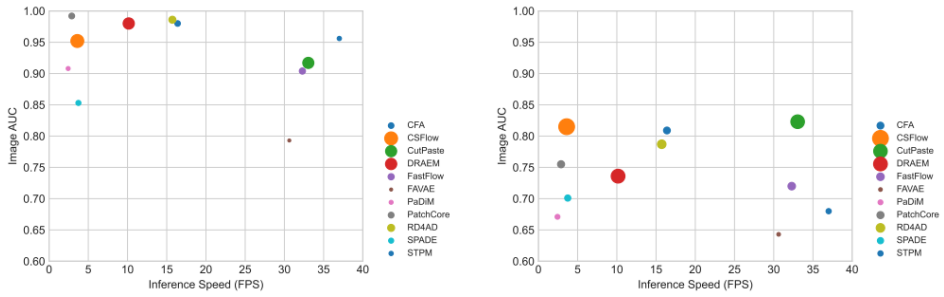
In their analysis, [25] and [39] provide valuable insights into industrial imaging anomaly detection. They examine various models, primarily unsupervised or semi-supervised, and their performance in several common industrial imaging anomaly detection datasets such as MVTec AD [3], MVTec AD Loco [4], MPDD [21], BTAD [27], MTD [20], VisA [45], and DAGM [28].

These datasets offer structured, well-labeled environments ideal for benchmarking anomaly detection models due to their clear definitions of normal and defective samples, as well as comprehensive ground truth labels that enable both image-level and pixel-level performance evaluations.

Although focused on industrial defect detection, these datasets contribute valuable insights for evaluating model performance on the controlled and clearly defined anomalies often found in road damage detection tasks. The structured, high-quality images allow for consistent benchmarking of essential model factors, including robustness, inference speed, efficiency, memory usage, and performance at both the image and pixel levels. Using industrial datasets as a starting point enables a clear comparison of anomaly detection models before deploying them in real-world road environments, where image backgrounds are less predictable, lighting conditions vary, and anomalies can take many forms.

Among these datasets MVTec AD dataset, while primarily focused on industrial anomaly detection, offers a well-established benchmark for evaluating the performance of anomaly detection models. It includes various types of anomalies under controlled conditions with clear ground truth segmentation, making it ideal for initial performance assessments of anomaly detection backbones such as DINOv2.

When selecting a model, essential factors include model robustness, speed, efficiency, image-level performance, pixel-level performance, approach, supervision type, learning paradigm, implementation difficulty, adaptability, and memory usage.



(a) MVTec AD (b) MVTec LOCO-AD

Figure 2.1: Visualization of Anomaly Detection Models, image AUROC performance on the Y-axis, inference speed on the X-axis, and GPU memory consumption denoted by the circle size for each model where a small circle means less memory consumption. [39]

Table 2.1: Comparison of Industrial Anomaly Detection algorithms with various datasets and metrics. The best and second-best results are marked in red and blue, respectively. [39]

Dataset	Metric ↑	CFA [23]	CS-Flow [33]	CutPaste [24]	DRAEM [43]	FastFlow [42]	FAVAE [12]	PaDiM [11]	PatchCore [32]	RD4AD [13]	SPADE [9]	STPM [38]
MVTec AD [3]	Image AUROC	0.981	0.952	0.918	0.981	0.905	0.793	0.908	0.992	0.986	0.854	0.924
	Image AP	0.993	0.975	0.965	0.990	0.945	0.913	0.950	0.998	0.995	0.940	0.954
	Pixel AUROC	0.971	–	–	0.975	0.955	0.889	0.966	0.994	0.978	0.954	0.954
	Pixel AP	0.538	–	–	0.689	0.307	0.452	0.561	0.580	0.542	0.452	0.518
	Pixel PRO	0.898	–	–	0.921	0.856	0.749	0.913	0.943	0.939	0.895	0.879
MVTec LOCO-AD [4]	Image AUROC	0.814	0.814	0.734	0.798	0.639	0.623	0.780	0.835	0.867	0.687	0.679
	Image AP	0.944	0.942	0.915	0.933	0.837	0.805	0.927	0.985	0.987	0.921	0.844
	Pixel AUROC	0.908	–	–	0.942	0.796	0.944	0.987	0.990	0.971	0.971	0.883
	Pixel AP	0.219	–	–	0.209	0.149	0.159	0.221	0.261	0.342	0.164	0.150
	Mean sPRO	0.581	–	–	0.426	0.357	0.446	0.521	0.343	0.637	0.520	0.428
MPDD [21]	Image AUROC	0.923	0.973	0.771	0.941	0.887	0.570	0.706	0.948	0.927	0.784	0.876
	Image AP	0.922	0.968	0.800	0.961	0.881	0.551	0.704	0.948	0.921	0.780	0.872
	Pixel AUROC	0.948	–	–	0.918	0.808	0.955	0.990	0.987	0.992	0.955	0.970
	Pixel AP	0.283	–	–	0.288	0.088	0.088	0.455	0.481	0.342	0.288	0.318
	Pixel PRO	0.832	–	–	0.781	0.706	0.848	0.939	0.953	0.920	0.848	0.939
BTAD [27]	Image AUROC	0.938	0.936	0.917	0.895	0.803	0.950	0.936	0.967	0.986	0.937	0.989
	Image AP	0.980	0.890	0.953	0.974	0.867	0.986	0.937	0.988	0.989	0.974	0.976
	Pixel AUROC	0.959	–	–	0.874	0.965	0.949	0.977	0.978	0.955	0.937	0.937
	Pixel AP	0.517	–	–	0.159	0.349	0.535	0.520	0.521	0.517	0.441	0.401
	Pixel PRO	0.702	–	–	0.629	0.725	0.713	0.798	0.752	0.723	0.745	0.667
MTD [20]	Image AUROC	0.913	0.887	0.830	0.782	0.847	0.895	0.798	0.884	0.906	0.870	0.889
	Image AP	0.959	0.945	0.912	0.885	0.947	0.867	0.788	0.988	0.947	0.920	0.847
	Pixel AUROC	0.731	–	–	0.660	0.710	0.735	0.768	0.836	0.693	0.742	0.642
	Pixel AP	0.246	–	–	0.146	0.208	0.190	0.603	0.218	0.342	0.218	0.172
	Pixel PRO	0.528	–	–	0.541	0.568	0.632	0.798	0.686	0.623	0.623	0.478
VisA [45]	Image AUROC	0.920	0.744	0.819	0.857	0.823	0.829	0.891	0.906	0.960	0.814	0.893
	Image AP	0.935	0.787	0.848	0.905	0.882	0.880	0.952	0.958	0.962	0.843	0.973
	Pixel AUROC	0.843	–	–	0.935	0.832	0.856	0.882	0.956	0.965	0.847	0.873
	Pixel AP	0.268	–	–	0.156	0.182	0.309	0.388	0.277	0.215	0.169	0.151
	Pixel PRO	0.551	–	–	0.724	0.598	0.598	0.859	0.912	0.709	0.659	0.620
DAGM [28]	Image AUROC	0.948	0.752	0.839	0.870	0.874	0.695	0.949	0.906	0.930	0.707	0.829
	Image AP	0.878	0.781	0.680	0.790	0.911	0.670	0.826	0.971	0.975	0.880	0.859
	Pixel AUROC	0.942	–	–	0.868	0.911	0.870	0.975	0.917	0.978	0.870	0.949
	Pixel AP	0.495	–	–	0.306	0.342	0.170	0.492	0.517	0.534	0.133	0.151
	Pixel PRO	0.870	–	–	0.710	0.799	0.600	0.906	0.893	0.930	0.707	0.668

Based on Figure 2.1, Cutpaste [24], along with CFA [23] and Fastflow [42], is among the top performers in terms of image AUROC and inference speed, it is not integrated into the anomalib library and has higher GPU memory consumption, and although CFA [23] generally produces better performance results, it was only implemented with convolutional neural networks as backbones. As shown in 2.1, with results taken from [39], Patchcore [32] showed the best performance across

the datasets, but has a slower inference speed, which could hinder the development phase. In addition, Fastflow[42] shows competitive results in Image-Level metrics, as the lack of ground-truth labels in anomaly detection can make Pixel-Level evaluation impossible. Considering the factors mentioned above, Fastflow [42] seems to be a good option. This is primarily due to the ability to utilize visual transformers like DINOv2 [29] for feature extraction, one of the fastest inference speeds, and relatively efficient GPU memory usage in the scope of current anomaly detection models.

Nevertheless, Patchcore and CFA could be considered valid alternatives.

2.1.3 Fastflow

One practical approach in unsupervised anomaly detection involves utilizing deep neural networks to extract features from normal images and then using statistical methods to model the distribution. During the training stage, only normal images are observed, but the normal and abnormal images simultaneously appear in inference, thus abnormal samples with distinct distributions are detected.

This methodology consists of two primary components: the feature extraction module and the distribution estimation module; this way, FastFlow can be used as a plug-in model with various feature extractors. Figure 2.2 exemplifies these two components.

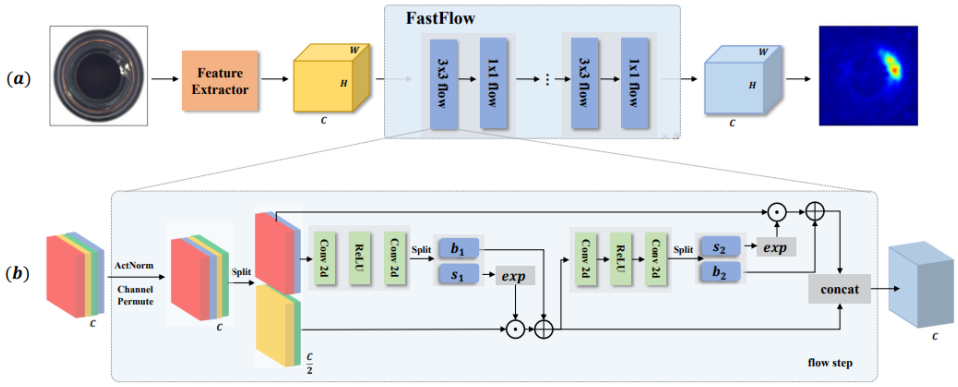


Figure 2.2: In a), it is possible to see the Fastflow Pipeline. A backbone model extracts the features of the input image and feeds it to the Fastflow Model, which outputs a Predicted Heat Map of the anomaly. One flow step of the Fastflow model is represented in b) [42]

2.1.4 Feature Extractor

Recent unsupervised anomaly detection techniques have benefited from the advancements in deep learning, utilizing deep neural networks as feature extractors to generate more promising results. Fastflow's authors achieved results using four different backbones: ResNet18 and Wide-ResNet50-2, variants of the Convolutional Neural Network Resnet [18], and two visual transformers ViT [15]. The ViT models have a global receptive field and can learn the relationship between global and local features more effectively. The authors applied DeiT [36], which introduces a teacher-student strategy specific to transformers, and CaiT [37], a proposed architecture designed

in the spirit of an encoder/decoder architecture. The backbone feature extractor is responsible for extracting features from the input image and feeding them to the 2D Flow Model. [42]

2.1.5 Distribution Modeling in FastFlow

After extracting features from the training dataset $D = \{x_1, x_2, \ldots, x_N\}$ where each x_i is a sample from the distribution $p_X(x)$, FastFlow aims to learn a parameter θ in the parameter space Θ to map all x_i from the raw distribution $p_X(x)$ into the standard normal distribution $p_Z(z)$. This mapping is achieved using a series of bijective and invertible transformations. Anomalous instances, which do not conform to this normal distribution, are effectively identified as they map out of the distribution $p_Z(z)$.

As shown in Figure 2, the 2D flow $f : X \to Z$ is used to project the image features $x \in p_X(x)$ into the hidden variable $z \in p_Z(z)$ with a bijective invertible mapping. For this bijection function, the change of the variable formula defines the model distribution on X by:

$$p_X(x) = p_Z(z) \left| \det \left(\frac{\partial z}{\partial x} \right) \right| \tag{2.1}$$

We can estimate the log-likelihoods for image features from $p_Z(z)$ by:

$$\log p_X(x) = \log p_Z(z) + \log \left| \det \left(\frac{\partial z}{\partial x} \right) \right| = \log p_Z(f_\theta(x)) + \log \left| \det \left(\frac{\partial f_\theta(x)}{\partial x} \right) \right| \tag{2.2}$$

where $z \sim \mathcal{N}(0, I)$ and $\frac{\partial f_\theta(x)}{\partial x}$ is the Jacobian of a bijective invertible flow model that $z = f_\theta(x)$ and $x = f_\theta^{-1}(z)$, θ is the parameter of the 2D flow model. During inference, the features of anomalous images should be out of distribution and hence have lower likelihoods than normal images, and the likelihood can be used as the anomaly score. Precisely, the two-dimensional probabilities of each channel are summed to get the final probability map and upsample it to the input image resolution using bilinear interpolation.

In actual implementation, the flow model f_{2d} is constructed by stacking multiple invertible transformations blocks f_i in a sequence that:

$$X \xrightarrow{f_1} H_1 \xrightarrow{f_2} H_2 \xrightarrow{f_3} \cdots \xrightarrow{f_K} Z, \tag{2.3}$$

and

$$X \xleftarrow{f_1^{-1}} H_1 \xleftarrow{f_2^{-1}} H_2 \xleftarrow{f_3^{-1}} \cdots \xleftarrow{f_K^{-1}} Z, \tag{2.4}$$

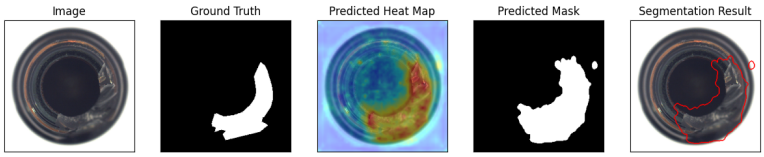
where the 2D flow model is $f_{2d} = f_1 \circ f_2 \circ f_3 \circ \cdots \circ f_K$ with K transformation blocks. Each transformation block f_i consists of multiple steps. Affine coupling layers are employed in each block, and each step is formulated as follows:

$$y_a, y_b = \text{split}(y) y'_a = y_a y'_b = s(y_a) \odot y_b + b(y_a) y' = \text{concat}(y'_a, y'_b), \tag{2.5}$$

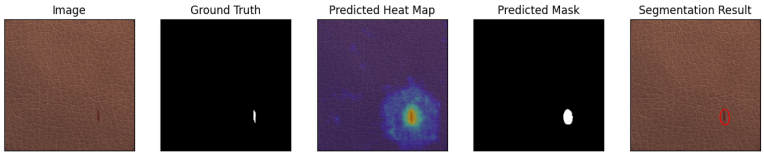
where $s(y_a)$ and $b(y_a)$ are outputs of two neural networks. The `split` and `concat` functions perform splitting and concatenation operations along the channel dimension. The two subnets $s(\cdot)$ and $b(\cdot)$ are usually implemented as fully connected networks in the original normalizing flow model and need to flatten and squeeze the input visual features from 2D to 1D which destroys the spatial position relationship in the feature map. To convert the original normalizing flow to a 2D manner, two-dimensional convolution layers are adopted in the default subnet to reserve spatial information in the flow model and adjust the loss function accordingly. In particular, a fully convolutional network in which 3×3 convolution and 1×1 convolution appear alternately is used, which reserves spatial information in the flow model. [42]

2.1.5.1 Results

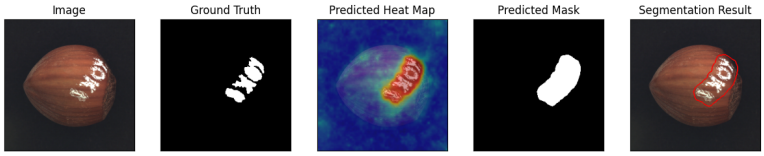
It is possible to visualize in Tables 2.2, 2.3, 2.4, and 2.5 quantitative results of Fastflow in the MVTEC AD dataset with the four backbones proposed by the authors of Fastflow, available in anomalib [1] for future comparison with the results of this research. Nonetheless, it is essential to note that the numbers were produced with early stopping callback with patience: 3 and that it might be possible to achieve higher metrics by increasing the patience. In Figure 2.3, we show a sample of the resulting segmentation result for three distinct categories of the MVTEC AD dataset with the Fastflow Model available in anomaly. While the MVTEC AD has ground truth labels, thus making it possible to evaluate pixel-wise performance, Fastflow still outputs a Predicted Heat Map and Mask even without ground truth masks, which can give a better understanding of qualitative results datasets without labeling.



(a) Bottle



(b) Leather



(c) Hazelnut

Figure 2.3: Sample Fastflow results on different categories from MVTec AD [1]

Table 2.2: Fastflow Image Level AUROC for mvtec-ad [1]

	ResNet-18	Wide ResNet50	DeiT	CaiT
Bottle	1.000	1.000	0.905	0.986
Cable	0.891	0.962	0.942	0.839
Capsule	0.900	0.963	0.819	0.913
Carpet	0.979	0.994	0.999	1.000
Grid	0.988	1.000	0.991	0.979
Hazelnut	0.846	0.994	0.900	0.948
Leather	1.000	0.999	0.999	0.991
Metal nut	0.963	0.995	0.911	0.963
Pill	0.916	0.942	0.910	0.916
Screw	0.521	0.839	0.705	0.791
Tile	0.967	1.000	0.993	0.998
Toothbrush	0.844	0.836	0.850	0.886
Transistor	0.938	0.979	0.993	0.983
Wood	0.978	0.992	0.979	0.989
Zipper	0.878	0.951	0.981	0.977
Average	0.907	0.963	0.925	0.944

Table 2.3: Fastflow Pixel Level AUROC for mvtec-ad [1]

	ResNet-18	Wide ResNet50	DeiT	CaiT
Bottle	0.983	0.986	0.991	0.984
Cable	0.954	0.972	0.973	0.981
Capsule	0.985	0.990	0.979	0.991
Carpet	0.983	0.991	0.991	0.992
Grid	0.985	0.992	0.980	0.979
Hazelnut	0.953	0.980	0.989	0.993
Leather	0.996	0.996	0.995	0.996
Metal nut	0.972	0.988	0.978	0.973
Pill	0.972	0.976	0.985	0.992
Screw	0.926	0.966	0.945	0.979
Tile	0.944	0.966	0.951	0.960
Toothbrush	0.979	0.980	0.985	0.992
Transistor	0.964	0.971	0.949	0.960
Wood	0.956	0.941	0.952	0.954
Zipper	0.965	0.985	0.978	0.979
Average	0.968	0.979	0.975	0.980

Table 2.4: Fastflow Image F1 score for mvtec-ad [1]

	ResNet-18	Wide ResNet50	DeiT	CaiT
Bottle	0.976	0.952	0.741	0.977
Cable	0.851	0.918	0.848	0.835
Capsule	0.937	0.952	0.905	0.928
Carpet	0.955	0.983	0.994	0.973
Grid	0.941	0.974	0.982	0.948
Hazelnut	0.852	0.979	0.828	0.900
Leather	0.995	0.974	0.995	0.963
Metal nut	0.925	0.969	0.899	0.916
Pill	0.946	0.949	0.949	0.616
Screw	0.853	0.893	0.868	0.979
Tile	0.947	0.994	0.976	0.994
Toothbrush	0.875	0.870	0.833	0.833
Transistor	0.779	0.854	0.873	0.909
Wood	0.983	0.968	0.944	0.967
Zipper	0.921	0.975	0.958	0.933
Average	0.916	0.947	0.906	0.911

Table 2.5: Fastflow Pixel F1 score for mvtec-ad [1]

	ResNet-18	Wide ResNet50	DeiT	CaiT
Bottle	0.670	0.733	0.753	0.725
Cable	0.547	0.564	0.487	0.608
Capsule	0.472	0.490	0.399	0.497
Carpet	0.573	0.598	0.586	0.606
Grid	0.412	0.481	0.393	0.410
Hazelnut	0.522	0.545	0.643	0.706
Leather	0.560	0.576	0.504	0.516
Metal nut	0.728	0.754	0.766	0.737
Pill	0.589	0.611	0.709	0.617
Screw	0.061	0.660	0.269	0.370
Tile	0.569	0.660	0.655	0.660
Toothbrush	0.479	0.481	0.524	0.535
Transistor	0.558	0.573	0.527	0.567
Wood	0.557	0.488	0.614	0.572
Zipper	0.492	0.621	0.522	0.504
Average	0.519	0.589	0.557	0.575

The metrics shown in Tables 2.1, 2.2, 2.3, 2.4, 2.5, and the sample of the qualitative results in Figure 2.3, show that there is room for improvement, specifically in some categories from MVTec AD such as Screw, Toothbrush, and in pixel-level evaluation as a whole. By leveraging different backbones than the ones proposed by the authors, it might be possible to obtain better results.

2.2 DINOv2

Feature extraction is a critical component of the FastFlow model, significantly impacting its ability to detect anomalies accurately and efficiently. The model can better understand and identify deviations from standard patterns by converting raw image data into meaningful features. Robust feature extraction ensures that essential details relevant to anomalies are highlighted, improving the model's sensitivity and specificity.

Foundation models are typically trained on vast amounts of diverse data and can be fine-tuned for specific applications. They are characterized by their ability to generalize across various domains and tasks without requiring task-specific training from scratch. DINOv2 [29] is an image foundation model that extracts features combining both local and global relations to enable the model to capture different types of anomalies. Furthermore, integrating DINOv2 as a backbone in FastFlow allows pre-trained models to b have already captured a wide range of visual features. Pretraining can significantly accelerate the development and deployment of anomaly detection systems, providing a solid starting point that only requires slight adjustments to excel in specific tasks.

2.2.1 DINOv2 Self-Supervised Training and Architecture

DINOv2 employs a distinctive self-supervised method to learn feature representations during training, incorporating elements from DINO [7] and iBOT's [44] techniques, together with SwAV's [6] centering mechanism. A feature distribution regularizer and a brief training phase with high-resolution images are included.

For image-level objectives, DINOv2 utilizes a cross-entropy loss that compares features of a student network with those of a teacher network, both derived from class tokens of a Vision Transformer (ViT) [15] processing different crops of the same image. The student's class token passes through a DINO head, a multilayered perceptron that outputs a vector called "prototype scores," followed by a softmax function to generate p_s. A similar process, including a softmax and a sinkhorn-Knopp or moving average centering algorithm [6], is applied to the teacher's class token to produce p_t. The DINO loss is calculated based on these prototype scores.

$$L_{\text{Detection}} = \frac{1}{|\mathscr{X}_{\text{det}}|} \sum_{(x,y)\in\mathscr{X}_{\text{det}}} \ell_{\text{BCE}}(f_\theta(x), y) \tag{2.6}$$

For patch-level objectives, DINOv2 introduces randomness by masking some patches in the input to the student network but not in the teacher's, applying respective iBOT heads to both, and then performing softmax and centering operations to calculate the iBOT loss based on masked patch indices.

$$L_{\text{Segmentation}} = \frac{1}{|\mathscr{X}_{\text{seg}}|} \sum_{(x,y)\in\mathscr{X}_{\text{seg}}} (\ell_{\text{BCE}}(f_\theta(x), y) + \alpha \cdot \text{Dice}(f_\theta(x), y)) \tag{2.7}$$

Furthermore, DINOv2 adopts the Sinkhorn-Knopp (SK) [6] centering technique from substituting the standard softmax centering in both DINO and iBOT methods with SK batch normalization, executing the SK algorithm for three iterations for the student side.

Moreover, introducing the KoLeo regularizer [35] by DINOv2 encourages evenly distributed feature representations within a batch. This regularizer computes the negative logarithm of the minimum distance between each vector and its nearest neighbor within the batch, with all features before calculation.

Finally, DINOv2 addresses the challenge of training with high-resolution images by temporarily increasing the resolution to 518x518 during the late pretraining phase. This strategy proves essential for detail-oriented tasks like segmentation or detection.

2.2.2 Feature Extraction with DINOv2

When the raw images are input into the pre-trained DINOv2 model, they undergo processing through a ViT-like architecture. This process yields local and global feature representations, which the anomaly detection models then utilize.

2.2.3 Anomaly Dino

DINOv2 has been recently adapted with great success in few-shot anomaly detection with Anomaly Dino [10]. The model leverages DINOv2 to extract meaningful patch-level features from the few training samples to a memory bank and augments them to enrich the information available from training. Then, in the testing phase, it compares the distance of the retrieved patch-level features of the test image to the ones stored in the memory bank to compute the anomaly score.

Table 2.6: Performance of AnomalyDINO under different settings on MVTec AD dataset. All results in % [10]

Setting	Detection				Segmentation		
	AUROC	F1-max	AP		AUROC	F1-max	PRO
1-shot	$96.6^{\pm 0.3}$	$96.6^{\pm 0.4}$	$98.2^{\pm 0.7}$	/	$96.3^{\pm 0.3}$	$58.7^{\pm 0.5}$	$91.9^{\pm 1.0}$
2-shot	$96.6^{\pm 0.3}$	$96.6^{\pm 0.4}$	$98.2^{\pm 0.7}$	/	$96.8^{\pm 0.4}$	$58.7^{\pm 0.5}$	$92.1^{\pm 1.0}$
4-shot	$97.7^{\pm 0.4}$	$97.1^{\pm 0.3}$	$98.5^{\pm 0.3}$	/	$96.8^{\pm 0.4}$	$58.7^{\pm 0.5}$	$92.1^{\pm 1.0}$
8-shot	$98.2^{\pm 0.3}$	$97.5^{\pm 0.5}$	$99.1^{\pm 0.2}$	/	$96.8^{\pm 0.4}$	$58.7^{\pm 0.5}$	$92.1^{\pm 1.0}$
16-shot	$98.4^{\pm 0.1}$	$97.8^{\pm 0.2}$	$99.2^{\pm 0.2}$	/	$97.8^{\pm 0.2}$	$60.2^{\pm 1.2}$	$93.0^{\pm 0.2}$

While few-shot anomaly detection and unsupervised anomaly detection are different paradigms, few-shot focuses on leveraging the power of large generalizable models to detect anomalies with

few or no prior training [25], which differs from unsupervised training, the results published by the authors of Anomaly Dino, see Table 2.6, can serve as comparison for results obtained with Fastflow with DINOv2 Backbone.

Using DINOv2 as a feature extraction backbone in FastFlow exemplifies the powerful synergy between advanced foundation models and specialized anomaly detection frameworks. This combination can enhance the robustness, accuracy, and efficiency of detecting anomalies across various applications.

2.3 Road Damage Dataset

2.3.1 The Gap Between Anomaly Detection in Controlled vs. Uncontrolled Environments

While significant progress has been made in anomaly detection, existing benchmarks [17], [34], [39] , [25], exemplified in Figure 2.4 focused on existing controlled datasets and its performance can deviate from what is expected in an uncontrolled setting. However, applying these techniques to uncontrolled environments like roads presents unique challenges. Controlled environments, such as those in the benchmarks, are characterized by predictable conditions, well-defined parameters, and consistent imaging conditions. Anomaly detection models in these settings benefit from the following:

- **Consistent Data:** Data in controlled environments is often stable, with minimal unexpected variations.
- **Clear Anomalies:** Anomalies are typically well-defined and easier to identify against the background of consistent normal patterns.
- **Structured Settings:** Controlled environments are structured with limited variables, making it easier to model normal behavior.

(a) Undamaged Bottle (b) Damaged Bottle (c) Damage Ground Truth

Figure 2.4: Sample of Bottle category from MVTEC AD [3]

Figure 2.5: Image from RDD dataset [2]

In comparison, uncontrolled environments, such as in Figure 2.5, pose distinct challenges when compared to controlled ones.

- **Variability:** Road conditions are highly variable, influenced by weather, traffic, and human behavior. This variability makes it difficult to establish a consistent baseline for normal behavior due to the introduction of different imaging and environmental conditions, which deviate from what would be expected in controlled environments.

- **Unpredictability:** Anomalies on roads can take many forms, from accidents and obstacles to sudden traffic flow changes, requiring highly adaptable and robust models.

- **Background:** Road images often contain background clutter, which can confuse and divert attention from the model's target.

Overcoming such boundaries is one of the main objectives of the dissertation.

2.3.2 Existing Datasets

Multiple datasets exist for road damage classification, each with different characteristics in terms of damage type. One major challenge in creating an anomaly detection dataset for road damage is that most existing road damage classification datasets only contain images of damaged examples. Since most current anomaly detection models are unsupervised or semi-supervised, the dataset should primarily consist of normal, undamaged images, with additional damaged images for validation and testing.

It's important to consider whether the damaged images have a ground truth mask, enabling pixel-wise evaluation compared to cases with no segmentation labels. While some image classification datasets contain multiple classes for road damage, anomaly detection is only focused on detecting abnormalities, so this differentiation is not strictly necessary. However, having numerous damage types allows for better generalization of potential real-world anomalies and creates a more comprehensive dataset.

The dataset should include images from different countries to evaluate a model's performance in general road environments. This provides a broader representation of anomalies due to other countries' varying building styles and elements.

It's also essential to understand the impact of data augmentation on model performance.

We present four different datasets:

2.3.2.1 Cracks-and-Potholes

The Cracks-and-Potholes [30] dataset provides 2235 manually selected pictures captured between 2014 and 2017 in Brazil, with cracks and potholes. It doesn't contain vehicles or people, and no capture problems exist. More importantly, each image has three correspondent masks: a road lane mask, a crack damage mask, and a pothole damage mask. This makes this dataset extremely valuable as it possibilitates pixel-wise evaluation.

2.3.2.2 RDD

Road Damage Detector (RDD) [2] dataset contains images from various countries and four different types of damages: D00: Longitudinal Crack, D10: Transverse Crack, D20: Aligator Crack, D40: Pothole. The diversity of source countries provides excellent images for the dataset due to various conditions present.

Table 2.7: Number of Total Images in the RDD Dataset by Country

Country	**#Total Images**
Japan	13133
India	9665
Czech	3538
Norway	10201
US	6005
China (MotorBike)	2477
China (Drone)	2401
Total	**47420**

2.3.2.3 Clean Littered

The Clean Littered [14] dataset is focused on image classification between clean and littered roads. While the littered images are unusable due to the lack of viewable road lanes, the clean roads can provide good, undamaged photos to train the model.

Table 2.8: Categories of the Clean Littered dataset

Category	**Clean**	**Littered**	**Total**
Number of images	113	124	237

2.3.2.4 SIH

Sih [16] provides four different types of classification. This dataset can be valuable as it contains roads in good conditions for model training and bad conditions for model evaluation.

Table 2.9: Categories in SIH dataset

Category	**Good**	**Satisfactory**	**Poor**	**Very Poor**	**Total**
Number of images	845	515	396	318	2074

Figure 2.10 shows a summarized version of the value each of these collections of images can provide to a road damage dataset for anomaly detection, while Tables 2.7, 2.8, 2.9 show the composition of some the source image datasets.

Table 2.10: Overview of Road Damage Datasets

Dataset Name	**Normal Images**	**Damaged Images**	**Ground Truth Mask**
RDD	X	✓	X
SIH	✓	✓	X
C_L	✓	X	X
C_P	X	✓	✓

Combining the advantages of each of these datasets related to anomaly detection makes it possible to create two separate datasets. $Dataset_1$ involving pixel-wise evaluation using the Cracks-and-Potholes damaged images as abnormal but having less variation in the anomalous category and $Dataset_2$ with broader anomaly generalization but no ground truth masks for segmentation analysis.

2.4 Image Segmentation

As discussed above, in uncontrolled environments, a significant challenge is the variation in backgrounds. Accurately separating roads from their backgrounds in images is essential for applications like autonomous driving, road condition monitoring, and urban planning. Eliminating background noise in the images allows the model to concentrate on the vital parts. Grounding techniques are crucial for this process as they help focus on the image's relevant parts while disregarding irrelevant background details.

2.4.1 Grounded SAM

Grounded SAM [31] combines Grounding DINO [26] as an open-set object detector with the Segmentate Anything Model (SAM) [22]. This allows it to detect and segment arbitrary regions within images based on arbitrary text inputs. SAM and Grounding DINO deliver strong zero-shot performance, meaning they can perform without prior knowledge about the images or any previous training.

2.4.1.1 Grounding Dino

Grounding DINO [26] is an open-set object detector that can detect any objects given a text prompt. Nonetheless, the model needs text as inputs and can only detect boxes with corresponding phrases.

2.4.1.2 Segment Anything

Segment Anything Model (SAM) [22] is an open-world segmentation model that can segment an object in an image. Still, it requires point or box prompts to identify masked objects and cannot be done based on arbitrary text input.

2.4.1.3 Model Pipeline

As Figures 2.6 and 2.7 demonstrate, given a text input, the model uses Grounding Dino to generate precise boxes for the regions of interest fed to SAM to generate precise mask segmentations. Leveraging both models' strengths together enables a precise segmentation of regions or objects in areas without prior training.

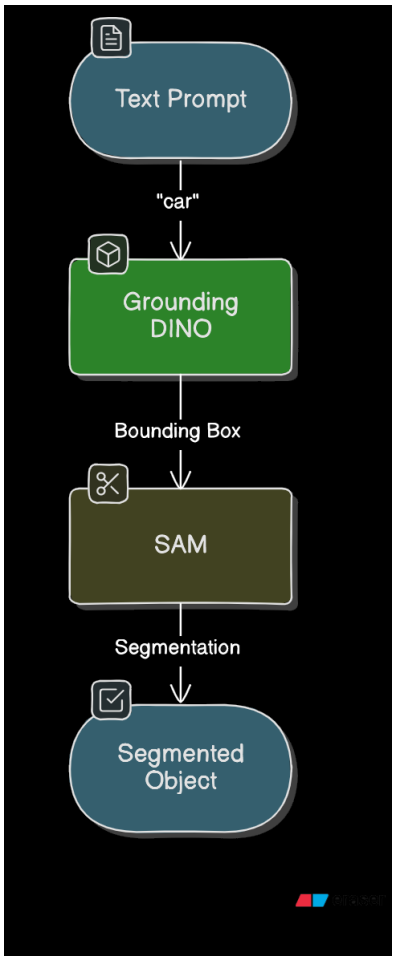


Figure 2.6: Grounded SAM pipeline.

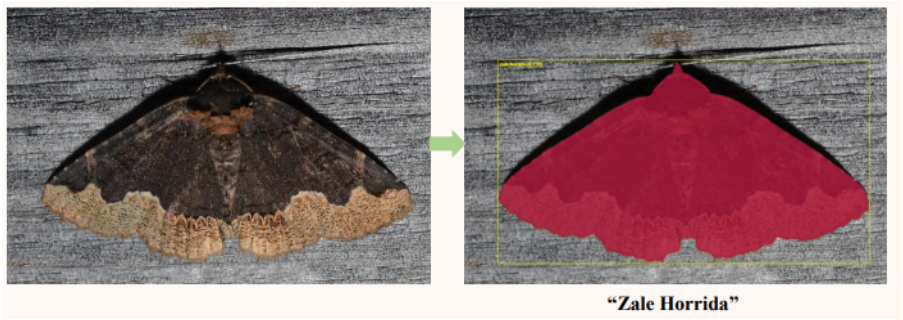


Figure 2.7: Visual example of Grounded SAM segmentation. [31]

2.4.2 Integration with Anomaly Detection

Using Grounded SAM, prompting an input text such as "road" and running the model. Grounded SAM enables dataset preprocessing by taking raw images from curated datasets and transforming them into only road lane images, effectively separating the road from the background, which can

potentially enhance model accuracy, efficiency, and reliability. This method introduces an additional processing layer to Anomaly Detection, which might be crucial for the task in uncontrolled backgrounds

2.5 Summary

Anomaly Detection enhances road safety by detecting anomalies like accidents or hazardous conditions early and assists in infrastructure monitoring for timely maintenance.

The Fastflow model can have good inference speed, low GPU memory usage, and ease of visual transformer implementation as a backbone. Patchcore and CFA are also viable options.

It is essential to emphasize the differences between uncontrolled and controlled environments in anomaly detection and create a comprehensive Road Damage Dataset to evaluate the model's ability to predict said anomalies.

Grounding Sam can enhance anomaly detection by effectively segmenting the road section of the images, thus removing unnecessary background.

Chapter 3

Methodology

3.1 Introduction

The methodology chapter outlines the research objective and design, including the theoretical reasoning behind the choice of the elements to achieve its goals and the qualitative and quantitative methods proposed to analyze the impact of the research.

This research has two main objectives:

The first objective is to measure the impact of different backbone feature extractors, namely, DINOv2 and Resnet18, in Fastflow anomaly detection. Thus, we must detail how to set up the Experiment so that the relevant metrics are used, analyze their impact, and ensure equal conditions to guarantee the validity of the experiment conclusions.

Our second aim is to construct and utilize an adequate road anomaly detection dataset. For this, it is essential to define the characteristics of a robust anomaly detection dataset, identify the sources used for its compilation, and what preprocessing is applied. This step is crucial for enhancing the dataset's validity and applicability.

3.2 Tools

This section will detail the tools used throughout the research and the reasoning behind them.

3.2.1 Graphics Processing Unit

The primary computational hardware used for this research was an NVIDIA RTX A2000 8GB Laptop GPU.

Specifications:

- **Model:** NVIDIA RTX A2000 8GB Laptop GPU
- **Driver Version:** 31.0.15.5186
- **Driver Date:** 3/12/2024

- **DirectX Version:** 12 (FL 12.1)
- **Memory:** 8.0 GB

This GPU ensured that the computational demands of model training, testing, and data processing were met locally without resorting to cloud-based environments like Google Colab [5].

3.2.2 Software

3.2.2.1 Python

Python was chosen as the primary programming language for this research due to its extensive libraries, ease of use, and strong community support. Specifically, Python version 3.12.2 was used.

3.2.2.2 Anomalib

Anomalib [1] is a deep learning library aiming to collect state-of-the-art anomaly detection algorithms for benchmarking public and private datasets. Anomalib provides several ready-to-use implementations of anomaly detection algorithms described in the recent literature and tools that facilitate the development and implementation of custom models. The version used was 1.1.0

3.2.2.3 Visual Studio Code

This research used Visual Studio Code (VS Code) version 1.87.2 as the primary code editor. It offers an intuitive interface and powerful debugging capabilities. The editor supports numerous extensions that enhance productivity, including tools for Python development, Jupyter Notebooks, and various data science libraries.

3.2.2.4 Hugging Face

Hugging Face is a leading provider of tools and models for natural language processing and machine learning. This research used the Hugging Face Hub version 0.23.4 to load model configurations for the Grounding SAM framework.

3.2.2.5 TIMM

TIMM (PyTorch Image Models) is a library that provides a collection of state-of-the-art, pretrained models for computer vision tasks. In this research, TIMM version 0.9.16 was utilized to create backbone feature extractor models, namely DINOv2.

3.2.2.6 CUDA

CUDA (Compute Unified Device Architecture) is a parallel computing platform and application programming interface (API) model created by NVIDIA. It allows developers to utilize NVIDIA

GPUs for general-purpose processing, providing significant speedups for computational tasks. In this research, CUDA was employed to accelerate the training and evaluation of machine learning models. By leveraging the parallel processing capabilities of the GPU, CUDA enabled more efficient handling of large datasets and complex computations, which are integral to anomaly detection tasks. Integrating CUDA into the research workflow significantly reduced the time required for model training and inference, enhancing overall productivity.

Specifically, CUDA 12.4 was used.

3.2.2.7 Jupyter Notebooks

Jupyter Notebooks is an open-source web application that allows researchers and developers to create and share documents that contain live code, equations, visualizations, and narrative text. They provide an interactive environment well-suited for data analysis, visualization, and machine learning. Jupyter Notebooks were extensively used in this research for data preprocessing, exploratory data analysis, model development, and result visualization. The ability to run code in small, manageable blocks and see immediate outputs facilitated a more iterative and exploratory approach to the research tasks.

3.2.2.8 OpenCV

OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library. In this research, OpenCV version 4.9.0.80 was utilized to handle image preprocessing and augmentation tasks essential for preparing the datasets used in anomaly detection.

3.2.2.9 Pytorch

PyTorch is an open-source deep learning framework that provides a flexible and efficient platform for developing and training neural networks. PyTorch version 2.2.1+cu121 was used in this research, leveraging its robust model development and training capabilities. Facilitated by CUDA 12.1, it allowed for efficient handling of complex computations and large datasets.

3.2.2.10 Git

Git is a distributed version control system widely used for tracking changes in source code during software development. In this research, Git version 2.44.0.windows.1 was primarily utilized to import repositories, ensuring access to various external codebases and resources necessary for the project.

3.3 Fastflow

Fastflow proved to be the better candidate amongst Fastflow, Patchcore, and CFA for our adaptation due to three main factors: Fastflow and CFA showed better inference speed than Patchcore, which can be crucial during development of the process of training and development, and to further future implementations of real-time operations; Fastflow has known compatibility with ViT architectures for the backbone. Specifically, DEIT and CAIT give us another comparison point other than typically implemented CNNs like Resnet18; during initial trials, Patchcores' memory requirements were shown to be too high for our hardware capacity without having to reduce batch size and image size significantly, which would reduce the validity of our results.

The model selection was restricted to those available in anomalib [1] to ensure consistent experiment settings throughout the research.

3.4 Model Deployment

To implement our customized Fastflow with the DINOv2 model, we used the Anomalib library and TIMM.

We used Anomalib's classes to utilize their modular architecture and integrate them with anomaly detection models, datasets, metrics, engines, and benchmarks. This ensures reproducibility.

3.4.1 Training and Testing Pipeline

To ensure reproducibility and consistent conditions across the experiments, we created a simple training and testing pipeline with the classes available in anomaly. This pipeline is composed of:

1. **Data Transformation Composition**: In the appropriate experiments, we applied Gaussian blur to make the model more robust to slight variations and noise, random rotation to prevent bias towards object orientation, random cropping to promote the learning of localized features, and image resizing to maintain the necessary image size for the models. We used the Torchvision library for these transformations.

2. **Datamodule Setup**: Using the Folder class in anomaly to create the data module, we specify the dataset path, name, image size, batch size, and task type. The task type can be segmentation or classification depending on the availability of ground truth masks. The batch size is 16, and the image size depends on the backbone used. DINOv2 is only compatible with 518x518 images. We chose image size 512x512 for Resnet18 to maintain coherence between models, the closest compatible input size to 518.

3. **Model Initialization**: We initialize the model by specifying the backbone and the number of flow steps. We use 5 for DINOv2 and 8 for Resnet18, the number proposed by the authors.

4. **Engine Setup**: We use the anomaly engine, which serves as a high-level interface that orchestrates the training, testing, and validation processes for anomaly detection. We specify the callbacks for early stopping when the image AUROC does not improve for three epochs and model checkpoint to save the best weights for the best image AUROC. We also define which metrics we want to track, such as the image AUROC and F1 score, and configure the GPU for the computations.

5. **Model Training**: We train the model using the engine capabilities. We need to specify the data module and the model. The engine handles the rest automatically.

6. **Model Testing**: Again, like in training, we specify the datamodule and the model. This process outputs the chosen metrics in the engine and saves the visual results in a folder.

3.4.2 Backbone Adaptation

To adapt DINOv2 to Fastflow, a few changes had to be made to the model's source code on Anomalib. We need to initialize the model, the scale, and the number of channels. These two variables are related to the shape of the DINOv2 feature outputs.

```
if backbone == "vit_large_patch14_dinov2.lvd142m":
            self.feature_extractor = timm.create_model(backbone, pretrained=True)
            channels = [1024]
            scales = [14]
```

Listing 3.1: Backbone initialization

We need to define the function that retrieves the features from DINOv2. The function extracts and processes features from an input tensor using a Vision Transformer (ViT) model based on the DINOv2 architecture. It begins by embedding the input tensor into patches and adds a class token. If present, a distillation token is also added. The combined tokens are augmented with positional embeddings and passed through 8 transformer blocks, which apply multi-head self-attention and feed-forward layers. After normalization, the method removes the class token, permutes and reshapes the tensor into a feature map format, and normalizes the features by their mean and standard deviation. The final output is a list containing the processed feature tensor.

```


    def _get_dino_features(self,input_tensor: torch.Tensor) -> list[torch.Tensor]:
        feature = self.feature_extractor.patch_embed(input_tensor)
        cls_token = self.feature_extractor.cls_token.expand(feature.shape[0], -1,
            -1)
        try:
            if self.feature_extractor.dist_token is None:
                feature = torch.cat((cls_token, feature), dim=1)
```

```
            else:
                feature = torch.cat(
                    (
                        cls_token,
                        self.feature_extractor.dist_token.expand(feature.shape[0],
                            -1, -1),
                        feature,
                    ),
                    dim=1,
                )
        except:
            feature = torch.cat((cls_token, feature), dim=1)

        feature = self.feature_extractor.pos_drop(feature + self.feature_extractor.
            pos_embed)

        for i in range(8): # The features are extracted from the 8th block
            feature = self.feature_extractor.blocks[i](feature)

        feature = self.feature_extractor.norm(feature)
        feature = feature[:, 1:, :]
        batch_size, _, num_channels = feature.shape
        feature = feature.permute(0, 2, 1)

        feature = feature.reshape(batch_size, num_channels, 518 // 14, 518 // 14)

            # Normalize the feature tensor
        mean = torch.mean(feature, dim=(0, 2, 3), keepdim=True)
        std = torch.std(feature, dim=(0, 2, 3), keepdim=True)
        feature = (feature - mean) / (std + 1e-6)  # Adding a small epsilon for
            numerical stability

        return [feature]
```

Listing 3.2: DINOv2 feature extraction function

The DINOv2 large patch model outputs features in a (1370, 1024) tensor, where 1370 is the total number of tokens, including a special class token, and 1024 is the feature dimension of each token. To prepare these features for use in the Fastflow framework, they must be reshaped into a feature map with the format $(batchsize, numberofchannels, height, width)$. Here, the arrangement of the patch tokens determines the height and width. Excluding the class token, there are 1369 patch tokens, which can be arranged into a 37x37 grid (since $\sqrt{1369} = 37$). Given an input image size of 518, each patch corresponds to $518/37 = 14$ pixels, setting the scale to 14. Thus, when setting up the backbone, we must put the variable channel to 1024 and scale it to 14.

3.5 Dataset Creation

The datasets were created using four source datasets to ensure representation diversity. To make an anomaly detection dataset for unsupervised training, we needed two types of image labels: normal images and damaged images. The dataset should include a large number of normal images for the model to learn their representations and a sufficient number of damaged images to evaluate its performance. The dataset needs to contain diverse instances of normal and abnormal images to enable the model to learn different types of representations and avoid overfitting to a specific type of anomaly. Additionally, the angle of the image perspective should align with what is expected from a vehicle, given that most gathered images and applications are focused on vehicle-related scenarios. Although the primary focus was unsupervised learning, the created datasets can be used for other types of learning supervision.

Two datasets were created, the main difference being the abnormal images they contained. The first dataset,D_1, contains images with ground truth damage masks, enabling pixel-level segmentation. The second dataset,D_2, does not contain ground truth masks, but it includes a wider variety of damaged images, which facilitates better assessment of model performance for image-level classification.

3.5.1 Image Sources

The datasets used were SIH [16], and Clean Littered [14] for the normal images and SIH, Cracks, and Potholes [30], and RDD [2].

- **Cracks and Potholes**: Cracks and Potholes dataset provides images of damages and, more importantly, ground truth damage masks for evaluation. It also contains a lane mask that can crop the relevant part of the image.

- **RDD**: The RDD dataset contains damaged images from Japan, India, Czechia, Norway, the US, and China, showcasing four types of damage. This diversity is extremely valuable for the evaluation phase of the Experiment and dataset. Notably, images taken by motorcycle or drone from China were considered significantly different from the other images and were not used.

- **SIH**: SIH provided normal images for training as well as abnormal images. The dataset is divided initially into four categories: Good, Satisfactory, Poor, and Very Poor. We can use images from the Good and Satisfactory categories for the normal category and images from the Poor category for the anomalous category. The images from the Very Poor category have conditions, such as perspective angle, that differ significantly from the remaining images and thus were not used.

- **Clean and Littered**: This dataset contributed only with normal images. The littered images have low road visibility, resulting in a section too small for the model to analyze.

3.5.2 Collection and Processing of Images

In the first step, we sorted the images into four folders: normal images from $Dataset_1$, abnormal images from $Dataset_1$, normal images from $Dataset_2$, and abnormal images from set $Dataset_2$. We also grouped the ground truth masks from $Dataset_1$ into a separate folder. Initially, the Cracks and Potholes dataset had two separate damage masks – one for cracks and another for potholes. We combined these masks into a single unified damage mask.

3.5.2.1 Grounded Sam

One aspect of our research involves anomaly detection in uncontrolled environments. To prepare the dataset for our Experiment, we organized the raw images into additional folders to address certain limitations. We created folders containing lane masks for each raw image. These masks were generated using Grounded SAM to identify the relevant regions in each image, except for those from the Cracks and Potholes dataset, which already had corresponding lane masks. We then used the lane masks from the raw images to crop the road.

The resulting images only contain the road, eliminating background interference for anomaly detection as exemplified in Figure 3.1.

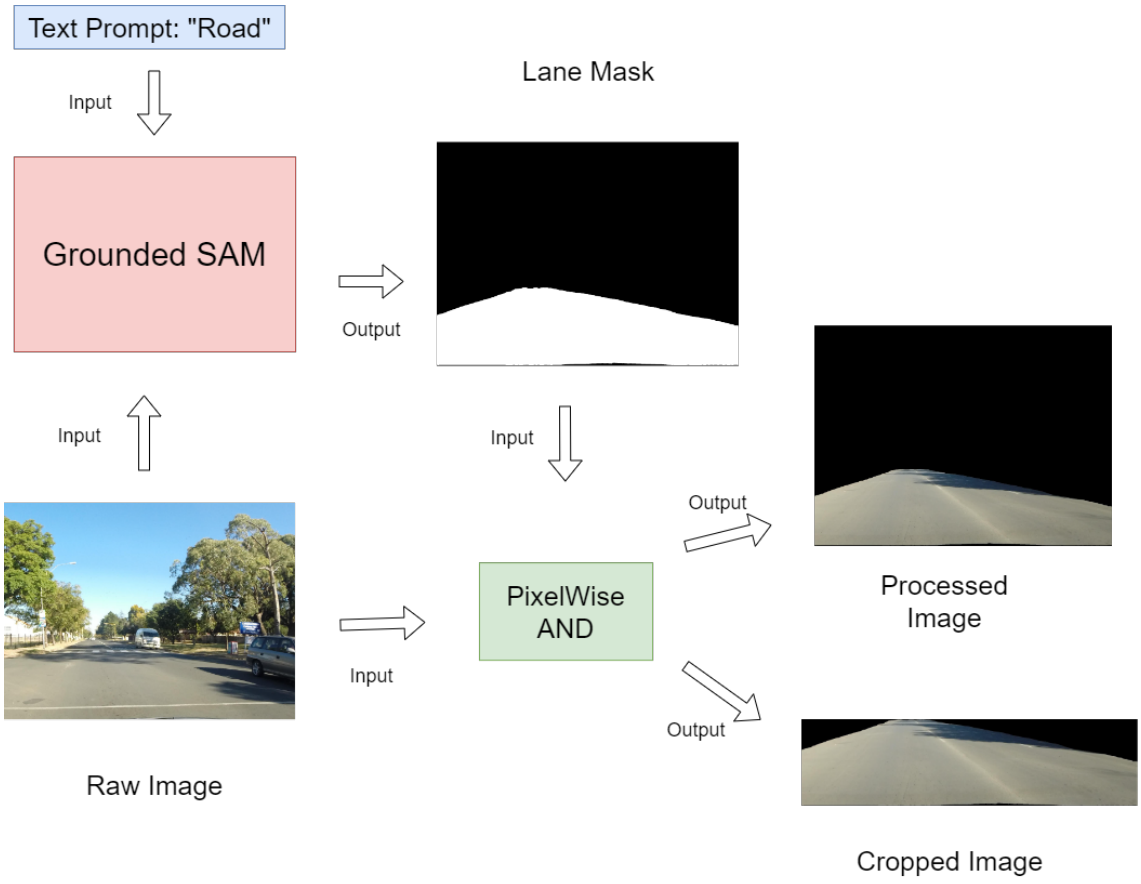


Figure 3.1: Image processing pipeline for our datasets.

Table 3.1: Dataset 1 Source Contribution

Source	Description	Count
Clean Littered	Clean images	113
SIH	Good images	764
Satis	Satisfactory images	515
Cracks and Potholes	Damaged images	2235
Total		**3627**

Table 3.2: Dataset 2 Source Contribution

Source	Description	Count
Clean Littered	Clean images	113
SIH	Good images	764
Satis	Satisfactory images	515
RDD	Additional RDD images	216
Total		**1608**

3.6 Metrics

The primary objectives of this study are to evaluate the effectiveness of integrating the DINOv2 backbone with the Fastflow model and to examine the reliability of our datasets when applied in both controlled and uncontrolled environments for anomaly detection.

Anomaly detection, in its essence, is a binary classification problem. We want to evaluate if the model accurately predicts if there is an anomaly in the image and, if so, where. If the model is not performing as expected, we also want to distinguish where the model is failing. Is it classifying normal images as anomalous or failing to detect anomalous cases?

To do this, we have to separate the possible results into two categories: **quantitatitve** and **qualitative** results.

3.6.1 Quantitative Metrics

The quantitative results were focused on two levels: **pixel** and **image**. Analysing pixel performance. Analyzing pixel performance involves evaluating how well the model detects and segments anomalies on a per-pixel basis. This level of analysis is crucial for applications where the exact location and shape of the anomaly are important, such as detecting cracks or defects in road images. High pixel-level performance ensures that the model can accurately highlight anomalous areas, enabling precise interventions and repairs.

Image-level analysis, on the other hand, evaluates the models. This analysis is particularly useful for tasks where the presence of any anomaly within the image needs to be detected rather than the exact localization. High image-level performance indicates that the model can effectively flag images for further inspection or processing.

- **Pixel F1**: The Pixel F1 Score is similar to the Image F1 Score but is calculated at the pixel level. It measures the precision and recall of anomaly detection on a per-pixel basis within an image. A high Pixel F1 Score signifies that the model accurately segments anomalies down to the pixel level, achieving a precise delineation of anomalous regions with minimal

false positives and false negatives.

$$\text{Pixel F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.1}$$

- **Pixel AUROC**: The Pixel AUROC assesses the model's performance at the pixel level, similar to Image AUROC but on a finer scale. It measures the true positive rate versus the false positive rate for each pixel in the images. A high Pixel AUROC value reflects the model's proficiency in accurately identifying anomalous pixels within the images, ensuring robust and precise anomaly localization.

$$\text{Pixel AUROC} = \int_0^1 \text{TPR}(\text{fpr})\, d\text{fpr} \tag{3.2}$$

where TPR is the True Positive Rate, and FPR is the False Positive Rate.

- **Image F1**: The Image F1 Score is a harmonic mean of precision and recall at the image level. It evaluates how well the model can identify anomalous images. A high Image F1 Score indicates that the model has a good balance between precision (correctly identifying anomalies) and recall (capturing all anomalies), ensuring that it neither misses anomalies nor generates too many false positives.

$$\text{Image F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3.3}$$

- **Image AUROC**: The Area Under the Receiver Operating Characteristic Curve (AUROC) at the image level evaluates the model's ability to distinguish between anomalous and normal images. It plots the true positive rate (sensitivity) against the false positive rate (1-specificity) at various threshold settings. A high Image AUROC value indicates that the model has a strong discriminatory capability, effectively differentiating between anomalous and normal images across different thresholds.

$$\text{Image AUROC} = \int_0^1 \text{TPR}(\text{fpr})\, d\text{fpr} \tag{3.4}$$

3.6.2 Interpretation of Metrics

While these metrics provide valuable insights into the model's performance, it is crucial to interpret them with caution:

3.6.2.1 Chance Classification

Given that anomaly detection is a binary classification problem, there is a possibility that the model might achieve seemingly good performance metrics by chance, especially if the dataset is imbalanced. For example, if the majority of images are normal, the model might simply predict

"normal" for most images and still achieve a high accuracy. Therefore, a comprehensive evaluation should consider the balance of the dataset and the possibility of chance classification.

3.6.2.2 Context and Application

The practical implications of the metrics should also be considered. For instance, in a safety-critical application, a false negative (an undetected anomaly) might be more severe than a false positive. Therefore, the chosen metrics and their thresholds should reflect the specific requirements and consequences of the application.

Combining both pixel-level and image-level analyses provides a comprehensive evaluation of the model's performance, ensuring that it is both accurate in detecting anomalies and precise in segmenting them.

3.6.3 Qualitative Metrics

The qualitative results of our model will be given to us by its visual outputs. Specifically, the Fastflow model outputs three images, as shown in the example from 3.2:

- **Predicted Heat Map**: It visualizes the areas of the image that the model considers anomalous. In this heat map, the intensity of the colors corresponds to the likelihood of anomaly presence. Higher intensity (usually represented in red) indicates regions that are more likely to be anomalous, whereas lower intensity (represented in blue or green) indicates normal regions. This heat map helps in understanding the spatial distribution of anomalies within the image and differentiating anomaly intensity;

- **Predicted Mask**: The Predicted Mask is a binary mask that delineates the exact regions classified as anomalous by the model. In this mask, the anomalous areas are typically marked in white, and the normal areas are marked in black. The Predicted Mask provides a clear and precise segmentation of the anomalies, which is useful if given a ground truth mask for comparison;

- **Segmentation Result**: The Segmentation Result is a composite image that overlays the Predicted Mask onto the original image. This overlay allows for a direct visual comparison between the predicted anomalous regions and the actual image content. The Segmentation Result is crucial for qualitative analysis as it shows how well the model's predictions align with the true anomalies in the image. It compensates for a possible lack of ground truth masks, providing a comprehensive view of the segmentation performance.

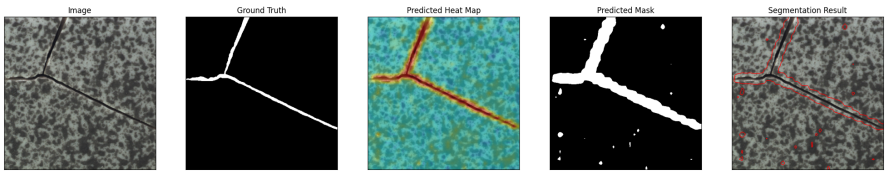


Figure 3.2: Visual outputs of the Fastflow model of an image of the Tile category from MVTec AD. From images 1 to 5: Original Image, Ground Truth mask, Predicted Heat Map, Predicted Mask, Segmentation Result

3.7 Experiments

To assess the effectiveness of using DINOv2, we need to compare its performance with what has previously been achieved using Fastflow. To do this, we are testing the model across all categories of the MVTec AD dataset, for which we have existing benchmark measurements. Using MVTec AD in this study provides a consistent and standardized basis for evaluating DINOv2's detection capabilities, especially before applying it to more complex, uncontrolled road damage data. However, it is recognized that MVTec AD's highly controlled, industrial setting differs from the dynamic and less predictable conditions found in road environments. Consequently, while MVTec AD offers a reliable performance baseline, further testing on real-world road damage datasets is essential to validate DINOv2's effectiveness in an uncontrolled, varied setting. To gain a better understanding of DINOv2's capabilities, we are also conducting experiments to study the impact of adjusting the number of flow steps and the number of blocks used for processing features in the DINOv2 ViT.

Additionally, we are evaluating Fastflow's performance using both DINOv2 and Resnet18 on our Road datasets, incorporating various levels of image preprocessing. Simultaneously, this process is not only about assessing the performance of the DINOv2 backbone but also about testing the suitability of our road dataset for anomaly detection.

We conducted six experiments on road datasets to evaluate various preprocessing techniques and their impact on model performance. $Dataset_1$ contains ground truth masks, allowing for pixel-wise evaluation, unlike $Dataset_2$. The experiments were organized as follows:

1. **Raw Images**: The original dataset without any preprocessing.

2. **Processed Images**: Images resulting from combining the lane mask with the raw image.

3. **Cropped Images**: Images resulting from combining the lane mask with the raw image and cropping to remove as much of the black background as possible.

4. **Transformed Images**: For each image type, an additional set of experiments was conducted with image transformations applied before training.

Experiment No.	Image Type	Description	Transformations Applied
1	Raw Images	Original, unprocessed images	No
2	Processed Images	Lane mask over raw image	No
3	Cropped Images	Cropped processed image to reduce black background	No
4	Raw Images	Original, unprocessed images	Yes
5	Processed Images	Lane mask combined with raw image	Yes
6	Cropped Images	Lane mask combined, cropped to reduce black background	Yes

Table 3.3: Overview of Road Datasets Experiments

Chapter 4

Results

4.1 MVTEC AD

4.1.1 Backbone Performance

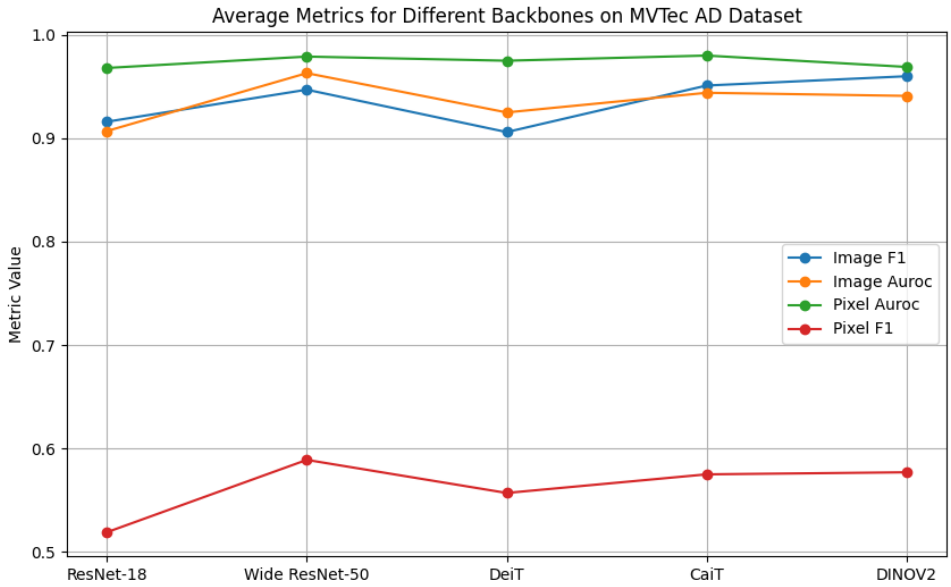


Figure 4.1: Metrics average values for different backbones of Fastflow model.

The visual outputs from the DINOv2 backbone exhibit precise anomaly detection capabilities. The predicted heat maps and segmentation masks align well with the ground truth, indicating the model's ability to localize and identify defects accurately. Figure 4.2 demonstrates that DINOv2 effectively captures subtle and prominent anomalies in different category of objects. DINOv2 has shown proficiency in detecting subtle anomalies that might be challenging to identify with less sophisticated models. For instance, minor texture deviations or small surface imperfections are

often highlighted in the heat maps, demonstrating the model's sensitivity to fine-grained details. For more apparent defects, such as large cracks or missing components, the segmentation masks generated by DINOv2 are precise, with clear boundaries that closely match the ground truth annotations. This accuracy in segmentation underscores the model's capability not only to identify but also to locate defects accurately. From Tables 4.1, 4.2, 4.3, 4.4 and Figure 4.1 we observe that DINOv2 generally performs well across these metrics, often matching or surpassing the performance of other backbones. The consistency in high image auroc and image F1 scores across categories suggests DINOv2's robustness in generalizing across various types of defects. While the DINOv2 backbone has shown promising results in anomaly detection, it is important to acknowledge that the improvements over previous the method are not universally significant across all metrics and object categories. This observation suggests that while DINOv2 is a competitive model, it does not always provide a clear advantage over the existing methods. DINOv2 demonstrates strong performance in anomaly detection on the MVTec AD dataset, excelling in both visual and quantitative evaluations. Its ability to handle a diverse range of defects, coupled with high consistency across metrics, underscores its potential as a reliable model for industrial anomaly detection tasks.

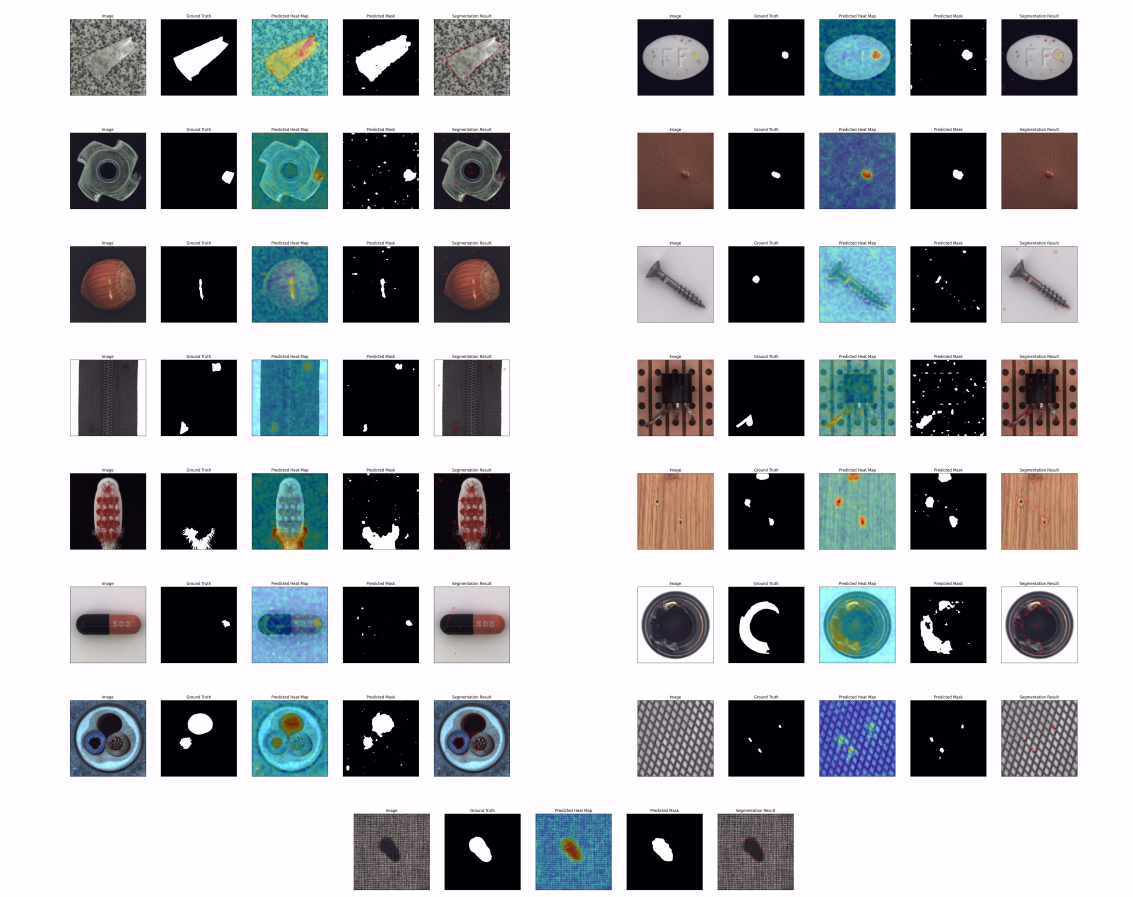


Figure 4.2: Visual results with DINOv2 backbone corresponding to each category from the MVTec AD dataset.

Table 4.1: Image_F1 Comparison. The best and second-best results are in red and blue, respectively.

Object	Resnet18	Wide ResNet50	DeiT	CaiT	DINOv2
Bottle	0.976	0.952	0.741	0.977	0.984
Cable	0.851	0.918	0.848	0.835	0.854
Capsule	0.937	0.952	0.905	0.928	0.926
Carpet	0.955	0.983	0.994	0.973	0.960
Grid	0.941	0.974	0.982	0.948	0.973
Hazelnut	0.852	0.979	0.828	0.900	0.872
Leather	0.995	0.974	0.995	0.963	0.995
Metal_Nut	0.925	0.969	0.899	0.916	0.967
Pill	0.946	0.949	0.949	0.616	0.938
Screw	0.853	0.893	0.868	0.979	0.852
Tile	0.947	0.994	0.976	0.994	0.953
ToothBrush	0.875	0.870	0.833	0.833	0.889
Transistor	0.779	0.854	0.873	0.909	0.780
Wood	0.983	0.968	0.944	0.967	0.967
Zipper	0.921	0.975	0.958	0.933	0.958

Table 4.2: Pixel_Auroc Comparison. The best and second-best results are in red and blue, respectively.

Object	Resnet18	Wide ResNet50	DeiT	CaiT	DINOv2
Bottle	0.983	0.986	0.991	0.984	0.969
Cable	0.954	0.972	0.973	0.981	0.939
Capsule	0.985	0.990	0.979	0.991	0.977
Carpet	0.983	0.991	0.991	0.992	0.986
Grid	0.985	0.992	0.980	0.979	0.979
Hazelnut	0.953	0.980	0.989	0.993	0.919
Leather	0.996	0.996	0.995	0.996	0.996
Metal_Nut	0.972	0.988	0.978	0.973	0.953
Pill	0.972	0.976	0.985	0.992	0.960
Screw	0.926	0.966	0.945	0.979	0.835
Tile	0.944	0.966	0.951	0.960	0.948
ToothBrush	0.979	0.980	0.985	0.992	0.967
Transistor	0.964	0.971	0.949	0.960	0.924
Wood	0.956	0.941	0.952	0.954	0.957
Zipper	0.965	0.985	0.978	0.979	0.979

Table 4.3: Pixel_F1 Comparison. The best and second-best results are in red and blue, respectively.

Object	Resnet18	Wide ResNet50	DeiT	CaiT	DINOv2
Bottle	0.670	0.733	0.753	0.725	0.636
Cable	0.547	0.564	0.487	0.608	0.518
Capsule	0.472	0.490	0.399	0.497	0.439
Carpet	0.573	0.598	0.586	0.606	0.571
Grid	0.412	0.481	0.393	0.410	0.456
Hazelnut	0.522	0.545	0.643	0.706	0.468
Leather	0.560	0.576	0.504	0.516	0.581
Metal_Nut	0.728	0.754	0.766	0.737	0.673
Pill	0.589	0.611	0.709	0.617	0.561
Screw	0.061	0.660	0.269	0.370	0.053
Tile	0.569	0.660	0.655	0.660	0.608
ToothBrush	0.479	0.481	0.524	0.535	0.457
Transistor	0.558	0.573	0.527	0.567	0.512
Wood	0.557	0.488	0.614	0.572	0.577
Zipper	0.492	0.621	0.522	0.504	0.562

Table 4.4: Image_Auroc Comparison. The best and second-best results are in red and blue, respectively.

Object	Resnet18	Wide ResNet50	DeiT	CaiT	DINOv2
Bottle	1.000	1.000	0.905	0.986	0.890
Cable	0.891	0.962	0.942	0.839	0.890
Capsule	0.900	0.963	0.819	0.913	0.880
Carpet	0.979	0.994	0.999	1.000	0.983
Grid	0.988	1.000	0.991	0.979	0.982
Hazelnut	0.846	0.994	0.900	0.948	0.854
Leather	1.000	0.999	0.999	0.991	1.000
Metal_Nut	0.963	0.995	0.911	0.963	0.982
Pill	0.916	0.942	0.910	0.916	0.866
Screw	0.521	0.839	0.705	0.791	0.658
Tile	0.967	1.000	0.993	0.998	0.980
ToothBrush	0.844	0.836	0.850	0.833	0.819
Transistor	0.938	0.979	0.993	0.983	0.862
Wood	0.978	0.992	0.979	0.989	0.954
Zipper	0.878	0.951	0.981	0.977	0.951

4.1.2 Number of Flow steps

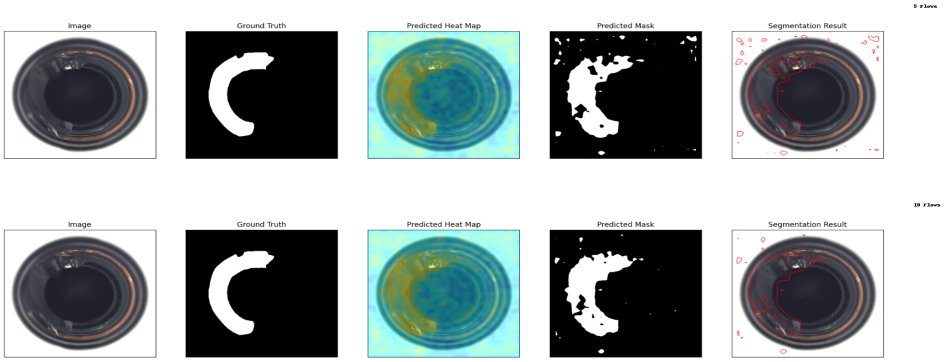


Figure 4.3: Visual outputs from Fastflow differing in flow from which the features were extracted

The analysis of the model's performance with five flow steps and ten flow steps, seen in Tables 4.5, 4.6, 4.7, 4.8, and Figure 4.3, reveals that neither configuration shows a significant advantage over the other in evaluated metrics and overall visual acuity. Several potential reasons can explain this observation:

The concept of diminishing returns may explain the lack of significant improvement when increasing the number of flow steps. As the model complexity grows, the additional capacity provided by more flow steps might not substantially enhance the model's ability to capture relevant features. The model may reach a saturation point at five flows, where it already captures most of the necessary information for effective anomaly detection. Beyond this point, the increase in flow steps does not yield proportionate improvements in performance metrics.

Increasing the number of flow steps can lead to overfitting, where the model becomes overly sensitive to the training data and captures noise or irrelevant patterns. This overfitting can negate the benefits of added complexity. The observed results suggest that the model might be balancing at an optimal point with five flows, where the risk of overfitting is minimized while maintaining sufficient model capacity for generalization.

The complexity associated with more flow steps can lead to numerical instability, especially in operations involving small numerical values. This phenomenon was observed for higher numbers of flow steps using DINOv2, and, further experiments with a higher number of flow steps were not conducted. Such instability can cause inconsistencies in model outputs, reducing the reliability of predictions.

Increasing the number of flows also increases the computational cost. If performance improvement is marginal, the additional computational burden might not be justified, especially in practical applications where efficiency is crucial.

The absence of a significant performance advantage with ten flows over five flows suggests that the model has reached a point of diminishing returns regarding the number of flow steps. This plateau indicates that the current level of complexity is sufficient for the anomaly detection task, and additional flow steps do not provide substantial benefits. This finding underscores the

importance of optimizing model complexity and balancing computational costs, model capacity, and performance gains.

Table 4.5: Pixel_Auroc Comparison for different flow steps

Object	5 Flows	10 Flows
Bottle	0.953	0.959
Cable	0.901	0.889
Capsule	0.961	0.972
Carpet	0.951	0.945
Grid	0.993	0.993
Hazelnut	0.980	0.973
Leather	0.995	0.996
Metal_Nut	0.909	0.930
Pill	0.940	0.903
Screw	0.909	0.944
Tile	0.984	0.979
ToothBrush	0.971	0.917
Transistor	0.915	0.917
Wood	0.972	0.974
Zipper	0.985	0.977

Table 4.6: Pixel_F1 Comparison for different flow steps

Object	5 Flows	10 Flows
Bottle	0.577	0.598
Cable	0.474	0.480
Capsule	0.384	0.418
Carpet	0.627	0.594
Grid	0.574	0.582
Hazelnut	0.639	0.629
Leather	0.642	0.642
Metal_Nut	0.605	0.607
Pill	0.544	0.123
Screw	0.095	0.151
Tile	0.786	0.774
ToothBrush	0.445	0.498
Transistor	0.490	0.498
Wood	0.704	0.714
Zipper	0.653	0.590

Table 4.7: Image_Auroc Comparison for different flow steps

Object	5 Flows	10 Flows
Bottle	0.887	0.915
Cable	0.672	0.663
Capsule	0.624	0.755
Carpet	0.972	0.974
Grid	1.000	1.000
Hazelnut	0.656	0.713
Leather	0.967	0.989
Metal_Nut	0.915	0.913
Pill	0.715	0.660
Screw	0.537	0.696
Tile	0.924	0.950
ToothBrush	0.883	0.594
Transistor	0.698	0.721
Wood	0.949	0.967
Zipper	0.840	0.886

Table 4.8: Image_F1 Comparison for different flow steps

Object	5 Flows	10 Flows
Bottle	0.913	0.931
Cable	0.791	0.789
Capsule	0.909	0.904
Carpet	0.955	0.959
Grid	0.991	0.991
Hazelnut	0.836	0.843
Leather	0.984	0.984
Metal_Nut	0.919	0.923
Pill	0.914	0.849
Screw	0.849	0.857
Tile	0.913	0.950
ToothBrush	0.921	0.817
Transistor	0.619	0.636
Wood	0.920	0.951
Zipper	0.919	0.932

4.1.3 Block Performance

Block	Pixel_Auroc	Pixel_F1	Image_Auroc	Image_F1
4	0.906	0.412	0.963	0.968
6	0.972	0.693	0.890	0.938
8	0.953	0.577	0.887	0.913
10	0.964	0.611	0.893	0.919
12	0.841	0.299	0.908	0.910
14	0.838	0.285	0.915	0.923
16	0.931	0.454	0.894	0.906
20	0.830	0.270	0.994	0.984
23	0.777	0.229	0.990	0.968

Table 4.9: Performance metrics for features extracted from different blocks of DINOv2.

By analyzing the results from Table 4.9, and Figure 4.4, we can observe patterns in block performance that reveal how feature extraction depth impacts anomaly detection accuracy.

Block 4 results suggest that shallow layers capture some useful features but may not be optimal for fine-grained anomaly detection. The features extracted at this stage are limited to basic patterns like edges and textures, which, while valuable, lack the complexity needed to distinguish nuanced anomalies.

Between Blocks 6 and 10, we observe the best performance in visual and pixel-wise anomaly detection. The features extracted in these mid-level layers strike a balance between the abstraction of high-level concepts and the preservation of essential details, resulting in:

- **Better Alignment to Ground Truth**: The features from these layers are detailed enough to capture specific anomalies, providing better alignment with the ground truth.
- **Clearer Boundaries**: Mid-level layers can effectively capture the boundaries of anomalous regions. This capability is likely due to the combination of detailed low-level features and some degree of abstract high-level context, which helps distinguish between anomalous and non-anomalous regions.

Beginning from Block 12, the model begins to lose the ability to capture the anomalous region boundaries but shows better image classification metrics, overall. This phenomenon can be attributed to several factors:

- **Reduced Spatial Resolution**: Deeper layers typically involve pooling operations, reducing the feature maps' spatial resolution. This can lead to the loss of finer details, making the network less sensitive to small or subtle anomalies.
- **Increased Abstraction and Generalization**: Features in deeper layers are more abstract, representing complex concepts and high-level patterns rather than specific details. This

abstraction aids in understanding the overall context of an image, which is beneficial for image-level classification tasks. However, this can blur distinctions between normal and anomalous regions, particularly for small or fine-grained anomalies.

- **Focus on Context Over Detail**: As the network progresses deeper, the focus shifts towards capturing the overall context and semantics of the image. While this is advantageous for classifying an entire image as anomalous or normal, it compromises the network's ability to localize anomalies precisely.

The observed trend—where mid-level blocks provide the best pixel-wise results while deeper blocks excel in image-level classification highlights a fundamental trade-off between **Pixel-Level performance vs. Image-Level performance** : Mid-level layers, with their detailed yet sufficiently abstract features, are better suited for tasks requiring high precision at the pixel level. However, they may not fully capture the broader context for maximizing image-level classification.

The variation in performance across different blocks of DINOv2 reflects the inherent trade-offs in neural network architectures between capturing detailed local features and abstracting broader contexts. For tasks requiring precise anomaly localization, relying on mid-level features appears optimal, while tasks focused on image-level classification benefit more from deeper, abstracted features. This insight can guide the design and optimization of models for specific anomaly detection tasks, depending on the desired balance between pixel-level and image-level accuracy.

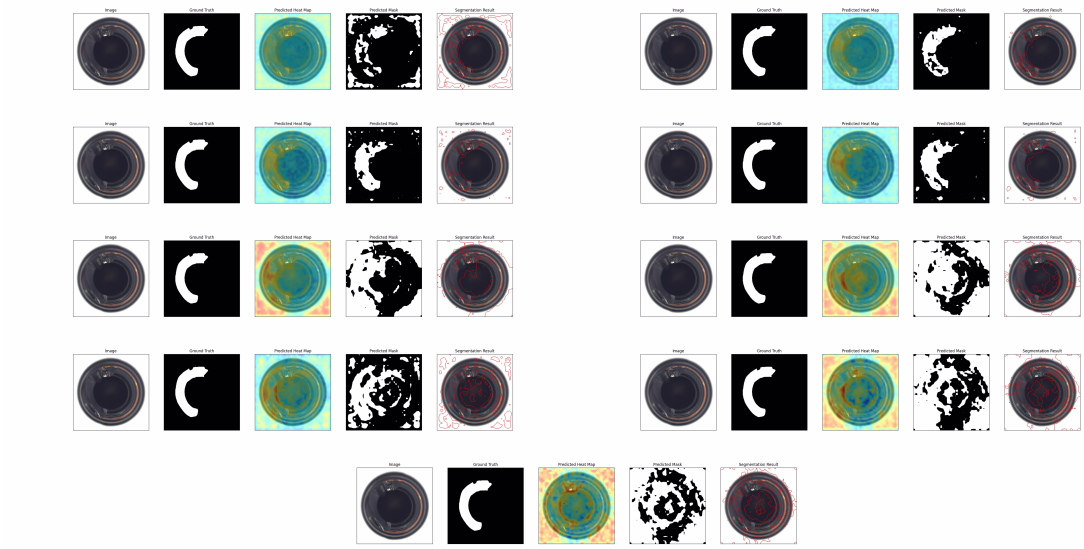


Figure 4.4: Visual outputs from Fastflow differing in the block from which the features were extracted. From left to right and top to bottom: 4 blocks, 6 blocks, 8 blocks, 10 blocks, 12 blocks, 14 blocks, 16 blocks, 20 blocks, and 23 blocks respectively.

4.1.4 Anomaly Dino vs Fastflow DINOv2

Table 4.10: Comparison of AUROC average values in %

Setting	Detection AUROC	Segmentation AUROC
1-shot	96.6 ± 0.3	96.3 ± 0.3
2-shot	96.6 ± 0.3	96.3 ± 0.5
4-shot	97.7 ± 0.4	98.0 ± 0.4
8-shot	98.2 ± 0.3	98.4 ± 0.3
16-shot	98.4 ± 0.1	97.8 ± 0.2
DINOv2	89.93	95.44

Table 4.10 shows that DINOv2 shows slightly lower performance in image classification, but a competitive behavior in segmentation tasks compared to anomaly dino. This is explained by the difference in model architectures, suggesting there is room for improvement with DINOv2 as a backbone.

4.2 Analysis of Model Performance on Road Datasets

4.2.1 Dataset_1

Table 4.11: Pixel_Auroc Comparison - ResNet18 vs DINOv2 $Dataset_1$

Test Type	ResNet18 Pixel_Auroc	DINOv2 Pixel_Auroc
Raw Image	0.538	0.575
Processed Mask	0.504	0.347
Cropped Mask	0.524	0.415
Raw Image Transformed	0.528	0.408
Processed Image Transformed	0.624	0.380
Cropped Image Transformed	0.622	0.474

Table 4.12: Pixel_F1 Comparison - ResNet18 vs DINOv2 $Dataset_1$

Test Type	ResNet18 Pixel_F1	DINOv2 Pixel_F1
Raw Image	0.031	0.036
Processed Mask	0.029	0.026
Cropped Mask	0.027	0.026
Raw Image Transformed	0.031	0.026
Processed Image Transformed	0.037	0.026
Cropped Image Transformed	0.036	0.027

Table 4.13: Image_Auroc Comparison - ResNet18 vs DINOv2 $Dataset_1$

Test Type	ResNet18 Image_Auroc	DINOv2 Image_Auroc
Raw Image	0.958	0.996
Processed Mask	1.000	0.981
Cropped Mask	0.918	0.997
Raw Image Transformed	0.518	0.999
Processed Image Transformed	0.548	0.677
Cropped Image Transformed	0.387	0.999

Table 4.14: Image_F1 Comparison - ResNet18 vs DINOv2 $Dataset_1$

Test Type	ResNet18 Image_F1	DINOv2 Image_F1
Raw Image	0.989	0.997
Processed Mask	0.999	0.999
Cropped Mask	0.982	0.997
Raw Image Transformed	0.973	0.999
Processed Image Transformed	0.975	0.975
Cropped Image Transformed	0.973	0.999

The metrics for the road datasets from Tables 4.11, 4.12, 4.13, 4.14, 4.15, 4.16 present a misleading picture. This is particularly evident in the Pixel F1 scores, seen in Table 4.12, which indicates that the model struggles to accurately identify anomalous areas. This issue is also apparent in the visual analysis of the results in Figures 4.5, 4.6, 4.7, 4.8. The imbalance in classes, both in images and pixels, can contribute to this problem. As a result, greater emphasis is placed

on visual inspection of the results. Precision and recall should have been used. They provide a clearer picture of model performance by focusing on its ability to accurately identify true anomalies (recall) and limit false positives (precision). Precision measures the proportion of correctly identified anomalies out of all anomalies flagged by the model, helping to assess how reliable the flagged instances are. Recall, on the other hand, measures the proportion of actual anomalies that the model successfully identifies, highlighting its sensitivity to anomalous areas.

Using precision and recall would have provided a more nuanced view of the effectiveness of the model, especially in an imbalanced dataset where anomalies are rare. Unlike the Pixel F1 score, which combines these metrics, precision and recall separately address the model's handling of true positives and false positives, allowing a more targeted evaluation of its performance in identifying road damage anomalies.

- **Resnet18 Performance**: When examining the processed and cropped images, Resnet18 shows some areas where anomalies are not detected, especially in regions with black backgrounds, but generally identifies the majority of the image as anomalous. In the case of raw images, Resnet18 typically fails to identify anomalies.

- **DINOv2 Performance**: DINOv2 tends to label most regions as anomalous with very high anomaly scores in the cropped and processed cases. This behavior is observed even in normal images. In the raw case, DINOv2 occasionally captures actual anomalies, particularly more prominent features such as potholes or extensive cracks, more effectively than Resnet18. However, the anomaly scores are not as high as in the cropped and processed images.

In general, DINOv2 performs better than Resnet18, particularly in its ability to detect anomalies in raw images. However, it still faces significant challenges, such as overdetection in processed and cropped images and difficulty correctly identifying and delineating anomaly boundaries. The issue may be attributed to the black backgrounds in these images, which might confuse the model and lead to a high rate of false positives. Additionally, both models struggle to detect and segment the anomalous regions effectively, which devalues the significance of DINOv2's superiority.

Considering the strong performance of Fastflow in MVTec AD, it is apparent that multiple factors may contribute to the poor performance. Further calibration may be necessary to improve the model performance in the raw case. Furthermore, the results in the cropped and processed cases suggest that trying to segment or crop the region of interest may not be a valid solution to this dataset. The results also indicate that alternative types of data preprocessing may be required, or that the normal images may be too dissimilar from the anomalous ones, rendering this dataset unsuitable for anomaly detection, particularly in an unsupervised learning setting.

The utilization of data transforms such as gaussian blur, random rotation, random cropping, and image resizing did not show clear improvements in performance metrics over the original images across all experiments as seen in Tables 4.11, 4.12, 4.13, 4.14, 4.15, 4.16. In some cases the results were significantly lower such as 4.13 . This may be attributed to the inherent complexity

of the images itself and the already present difficulties in the untreated case. The usage of these transforms could have distorted important features for the analysis of anomalies, The amount of images may also be insufficient for the utilization of such augmentations, to prove useful. Overall, we do not recommend the usage of data augmentation in this case.

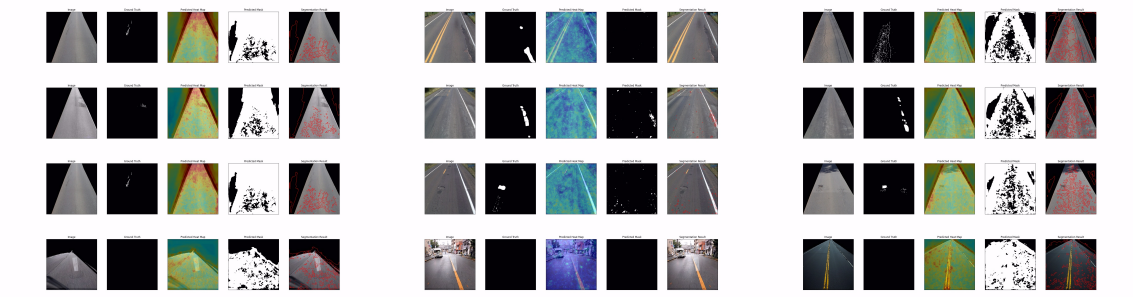


Figure 4.5: Resnet18 visual results of raw images from $Dataset_1$. From left to right, Processed, Raw, and Cropped experiments.

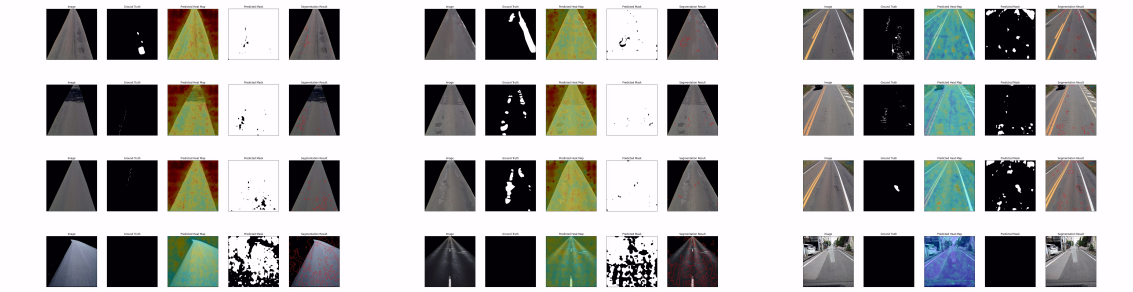


Figure 4.6: DINOv2 visual results from $Dataset_1$. From left to right Processed, Cropped, and Raw experiments

4.2.2 Dataset_2

Table 4.15: Image_Auroc Comparison - ResNet18 vs DINOv2 $Dataset_2$

Test Type	ResNet18 Image_Auroc	DINOv2 Image_Auroc
Raw Image	0.873	0.937
Image Mask	0.659	0.794
Cropped Image	0.828	0.901
Raw Image Transformed	0.583	0.765
Processed Image Transformed	0.929	0.989
Cropped Image Transformed	0.156	0.800

Table 4.16: Image_F1 Comparison - ResNet18 vs DINOv2 $Dataset_2$

Test Type	ResNet18 Image_F1	DINOv2 Image_F1
Raw Image	0.831	0.846
Image Mask	0.788	0.810
Cropped Image	0.805	0.913
Raw Image Transformed	0.774	0.752
Processed Image Transformed	0.907	0.982
Cropped Image Transformed	0.771	0.851

In processed and cropped images, DINOv2 reduces the overdetection observed in $Dataset_1$ but still struggles to capture the anomalies and their boundaries accurately. Similarly, Resnet18 continues to experience difficulties in distinguishing between normal and anomalous regions effectively, as seen in $Dataset_1$.

Contrary to the results from $Dataset_1$, the models now classify fewer processed and cropped images as anomalous. This change could be due to the anomalous and normal settings being more similar in $Dataset_2$ than in $Dataset_1$. Additionally, the results indicate that processing and cropping the images can lead to better visual outcomes, potentially reducing the amount of background incorrectly classified as anomalous.

Although DINOv2 outperforms Resnet18 overall, its performance is still not sufficient. Although the model occasionally identifies anomalies correctly, this is inconsistent and its overall detection capability is not robust. The excessive identification of normal regions as anomalous points to a fundamental issue with the model's ability to discern relevant from irrelevant features, which the diversity and complexity of the background in the images could exacerbate.

The differences observed in the models' performance between the road datasets and the MVTec dataset highlight the significant role that data quality and characteristics play in model performance. The highly curated and structured MVTec dataset provided more reliable and consistent results across various models, including DINOv2 and Resnet18. This contrast brings attention to potential issues with the road datasets:

- **Inconsistent Anomaly Definitions**: The road datasets may contain inconsistencies in what constitutes an anomaly, leading to confusion in the model's training process and evaluation metrics.

- **Background Noise**: Uncontrolled backgrounds and varied lighting conditions in the road datasets can introduce significant noise, complicating the anomaly detection task.

- **Labeling Quality**: The ground truth labels in the road datasets might not be as accurate or consistent as those in MVTec, impacting the models' ability to learn and generalize from these data.

These factors suggest that the suboptimal performance of the models, particularly in terms of F1 scores and visual results, may largely be attributed to data quality and not solely to the limitations of the models themselves. Consequently, there is a need for better-curated datasets that accurately represent the types of anomalies and conditions expected in real-world scenarios, with more precise labeling, different preprocessing such as angle and lighting, color normalization, and consistent definitions of anomalies to improve model training and evaluation. These issues underscore the main difficulties in creating a robust and general dataset of road damage anomalies.

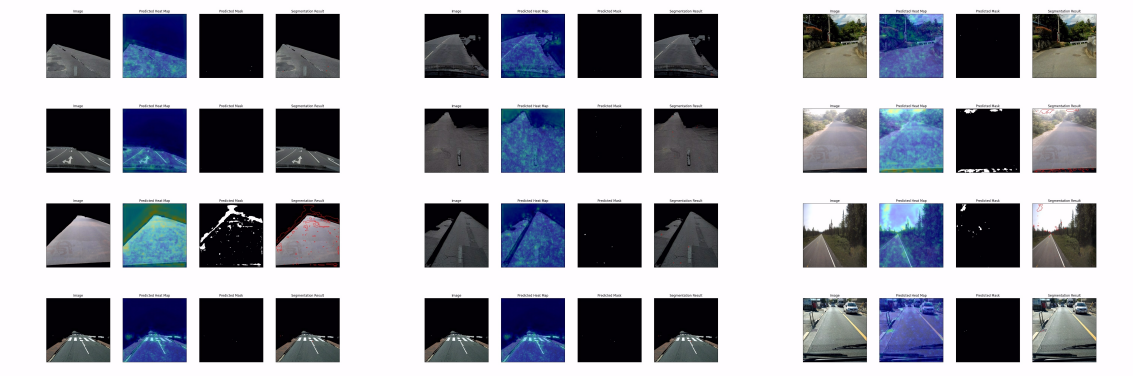


Figure 4.7: Resnet18 visual results from $Dataset_2$. From left to right, Processed, Cropped, and Raw experiments.

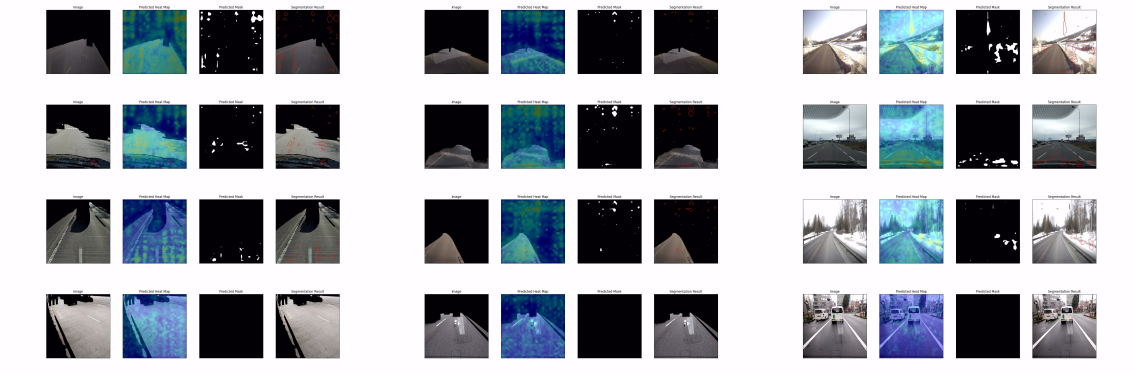


Figure 4.8: DINOv2 visual results from $Dataset_2$. From left to right, Cropped, Processed, and Raw experiments.

Chapter 5

Conclusions

This thesis explored the application of the DINOv2 backbone for anomaly detection in road damage datasets, providing a comprehensive analysis of its performance compared to traditional convolutional neural network backbones such as ResNet 18. The research aimed to evaluate the effectiveness of visual transformers in identifying anomalies under various conditions, including controlled and uncontrolled environments.

5.1 Main Contributions

The hypothesis that foundation models, such as DINOv2, could improve anomaly detection in uncontrolled environments such as roads was not verified for several reasons. Creating a suitable road anomaly detection dataset presented challenges, and current models struggled to accurately identify anomalies under these conditions. As a result, DINOv2 did not outperform traditional models and, within this dataset, remains unsuitable as a practical, efficient, and precise tool for real-world applications. Numerous challenges persist in adapting anomaly detection to dynamic and uncontrolled road environments.

5.1.1 DINOv2 in MVTEC AD

We demonstrated the potential of DINOv2 as a backbone within FastFlow, specifically we showed that:

- DINOv2 achieved comparable results to previously used backbones in FastFlow for the MVTec AD dataset, suggesting that it can be a competitive option for anomaly detection.
- The analysis of DINOv2's feature extraction across various block levels indicated that mid-level feature extraction enhances segmentation performance, whereas higher-level feature extraction suits classification tasks.
- Experiments on flow steps in DINOv2 FastFlow provided insights into optimal settings for balancing speed and accuracy, specifically that there is no improvement between using five

and ten flow steps, and that increasing the number of flow steps would eventually cause numerical instability.

5.1.2 Road Datasets

We provided information on uncontrolled environments for anomaly detection by creating two datasets for road damage anomaly detection and testing with our implementation of DINOv2 Fastflow and ResNet18 Fastflow. We highlight the weaknesses of our datasets and discuss potential mitigations. We also studied the application of region masks to segment the region of interest to eliminate the background in anomaly detection in uncontrolled environments.

Creating uncontrolled environment anomaly detection datasets remains a challenging task, and compiling images from different sources generally creates consistency issues. Although the success of DINOv2 in the MVTEC AD dataset seemingly proves the ability of DINOV2 to be utilized as a feature extraction backbone for anomaly detection, we want to highlight the difference in the difficulty of the usual benchmark datasets in the current literature and possible datasets with challenging settings. Current benchmark datasets are saturated, most models do not clearly surpass other previous architectures, and could better represent the broad spectrum of image anomaly detection possible. Most models might be overfitted to work on these few benchmark datasets, and their performance is already quite stagnated. New datasets could provide better tools to measure improvement between new and past models.

Specifically, we want to highlight the following conclusions of our research on using DINOv2 for road damage detection and the creation of road damage anomaly datasets:

- Two datasets were created for road damage anomaly detection, and used to evaluate the performance of DINOv2 FastFlow against ResNet18 FastFlow. While DINOv2 showed strengths in controlled datasets like MVTec AD, it struggled with uncontrolled and inconsistent conditions in road images. Both DINOv2 and ResNet18 were not able to successfully detect anomalies in the compiled datasets. While DINOv2 occasionally detected some relevant anomalies, usually if they are more pronounced, it does not do it in a consistent manner, and still flags many normal regions as anomalous.

- Overall we concluded that, in this state, DINOv2 with Fastflow did not improve the capability of anomaly detection compared to other models in road environments. This is mostly attributed to the inherent challenges to the images and environment.

- Other metrics should have been used to better grasp the results. The chosen metrics, AUROC and F1 score, while useful for other datasets such as MVTEC AD, were mostly useless for our setting and experiments since they were misleading. Metrics such as precision and recall should have been also used. One of the possibilities is that the model was correctly classifying the images based on different factors than the anomalous regions if the image sources were too distinct.

- The utilization of data transforms such as Gaussian blur, random rotation, random cropping, and image resizing did not show improvements in performance metrics over the original images, and in some cases, the results were lower. This may be attributed to the inherent complexity of the images itself and the already present difficulties in the untreated case. The usage of these transforms could have distorted important features for the analysis of anomalies, The amount of images may also be insufficient for the utilization of such augmentations, to prove useful. Overall, we do not recommend the usage of data augmentation in this case.

- The utilization of Grounded SAM did not improve the model's results. The model has difficulties in adapting to the processed cases if the difference between training and testing images is too high. The model detects anomalies in the removed background in some situations.

5.2 Future Work

Further studies of DINOv2 as a backbone can be conducted:

- A suitable dataset for anomaly detection remains to be created. They are explicitly tackling the challenge of gathering enough diversified training images to enable the model to generalize well and be robust while ensuring that the images are similar enough to measure the models accurately. Approaches to achieve this can range from perspective matching to dataset normalization.

- While our approach focused on using DINOv2 with Fastflow, other anomaly detection architectures could prove viable such as Patchcore, which was not used due to hardware constraints, or CFA.

- The measuring of the model's performance with recall and precision would allow us to verify more easily what percentage of anomalies the model correctly flags and the ratio between true anomalies and false positives.

- The masking of the road lane images in the feature extraction process instead of in the original image itself could potentially lead to better results. Our approach intended to limit the model's attention to the road lane region, but this was not always the outcome. Masking the regions of interest in the feature extraction process with methods such as Region of Interest Pooling or Align.

References

[1] Samet Akcay, Dick Ameln, Ashwin Vaidya, Barath Lakshmanan, Nilesh Ahuja, and Utku Genc. Anomalib: A deep learning library for anomaly detection. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 1706–1710. IEEE, 2022.

[2] Deeksha Arya, Hiroya Maeda, Sanjay Kumar Ghosh, Durga Toshniwal, and Yoshihide Sekimoto. Rdd2022: A multi-national image dataset for automatic road damage detection. *Geoscience Data Journal*, 11(4):846–862, 2024.

[3] Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. The mvtec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection. *International Journal of Computer Vision*, 129:1038–1059, April 2021.

[4] Paul Bergmann, Kilian Batzner, Michael Fauser, David Sattlegger, and Carsten Steger. Beyond dents and scratches: Logical constraints in unsupervised anomaly detection and localization. *International Journal of Computer Vision*, 130:947–969, April 2022.

[5] Ekaba Bisong. *Google Colaboratory*, pages 59–64. Apress, Berkeley, CA, 2019.

[6] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

[7] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9650–9660, Oct 2021.

[8] Xiaoran Chen and Ender Konukoglu. Unsupervised detection of lesions in brain mri using constrained adversarial auto-encoders. *arXiv preprint arXiv:1806.04972*, 2018.

[9] Niv Cohen and Yedid Hoshen. Sub-image anomaly detection with deep pyramid correspondences. *arXiv preprint arXiv:2005.02357*, 2020.

[10] Simon Damm, Mike Laszkiewicz, Johannes Lederer, and Asja Fischer. Anomaly-dino: Boosting patch-based few-shot anomaly detection with dinov2. *arXiv preprint arXiv:2405.14529*, 2024.

[11] Thomas Defard, Aleksandr Setkov, Angelique Loesch, and Romaric Audigier. Padim: A patch distribution modeling framework for anomaly detection and localization. In Alberto Del Bimbo, Rita Cucchiara, Stan Sclaroff, Giovanni Maria Farinella, Tao Mei, Marco Bertini, Hugo Jair Escalante, and Roberto Vezzani, editors, *Pattern Recognition. ICPR International Workshops and Challenges*, pages 475–489, Cham, 2021. Springer International Publishing.

[12] David Dehaene and Pierre Eline. Anomaly localization by modeling perceptual features. *arXiv preprint arXiv:2008.05369*, 2020.

[13] Hanqiu Deng and Xingyu Li. Anomaly detection via reverse distillation from one-class embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9737–9746, June 2022.

[14] Bugrahan Donmez. URL = https://www.kaggle.com/datasets/faizalkarim/cleandirty-road-classification [Acessed: 29 - Jun - 2024].

[15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2021.

[16] Prudhvi GNV. url=https://www.kaggle.com/datasets/prudhvignv/road-damage-classification-and-assessment?resource=download [Acessed: 25 - Jun - 2024].

[17] Songqiao Han, Xiyang Hu, Hailiang Huang, Minqi Jiang, and Yue Zhao. Adbench: Anomaly detection benchmark. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 32142–32159. Curran Associates, Inc., 2022.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[19] Waleed Hilal, S. Andrew Gadsden, and John Yawney. Financial fraud: A review of anomaly detection techniques and recent advances. *Expert Systems with Applications*, 193:116429, 2022.

[20] Yibin Huang, Congying Qiu, Yue Guo, Xiaonan Wang, and Kui Yuan. Surface defect saliency of magnetic tile. In *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*, pages 612–617, 2018.

[21] Stepan Jezek, Martin Jonak, Radim Burget, Pavel Dvorak, and Milos Skotak. Deep learning-based defect detection of metal parts: evaluating current methods in complex conditions. In *2021 13th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*, pages 66–71, 2021.

[22] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4015–4026, October 2023.

[23] Sungwook Lee, Seunghyun Lee, and Byung Cheol Song. Cfa: Coupled-hypersphere-based feature adaptation for target-oriented anomaly localization. *IEEE Access*, 10:78446–78454, 2022.

[24] Chun-Liang Li, Kihyuk Sohn, Jinsung Yoon, and Tomas Pfister. Cutpaste: Self-supervised learning for anomaly detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9664–9674, June 2021.

[25] Jiaqi Liu, Guoyang Xie, Jinbao Wang, Shangnian Li, Chengjie Wang, Feng Zheng, and Yaochu Jin. Deep industrial image anomaly detection: A survey. *Machine Intelligence Research*, 21(1):104–135, January 2024.

[26] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. *arXiv preprint arXiv:2303.05499*, 2024.

[27] Pankaj Mishra, Riccardo Verk, Daniele Fornasier, Claudio Piciarelli, and Gian Luca Foresti. Vt-adl: A vision transformer network for image anomaly detection and localization. In *2021 IEEE 30th International Symposium on Industrial Electronics (ISIE)*. IEEE, June 2021.

[28] German Chapter of the European Neural Network Society (GNNS). url=https://paperswithcode.com/dataset/dagm2007 [Acessed: 6-7-2024].

[29] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2024.

[30] Bianka T. Passos, Mateus J. Cassaniga, Anita M. R. Fernandes, Kátya B. Medeiros, and Eros Comunello. Cracks and potholes in road images. https://data.mendeley.com/datasets/t576ydh9v8/4, 2020.

[31] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.

[32] Karsten Roth, Latha Pemula, Joaquin Zepeda, Bernhard Schölkopf, Thomas Brox, and Peter Gehler. Towards total recall in industrial anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14318–14328, June 2022.

[33] Marco Rudolph, Tom Wehrbein, Bodo Rosenhahn, and Bastian Wandt. Fully convolutional cross-scale-flows for image-based defect detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 1088–1097, January 2022.

[34] Lukas Ruff, Jacob R. Kauffmann, Robert A. Vandermeulen, Gregoire Montavon, Wojciech Samek, Marius Kloft, Thomas G. Dietterich, and Klaus-Robert Muller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109(5):756–795, May 2021.

[35] Alexandre Sablayrolles, Matthijs Douze, Cordelia Schmid, and Hervé Jégou. Spreading vectors for similarity search. *arXiv preprint arXiv:1806.03198*, 2019.

[36] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers amp; distillation through attention. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10347–10357. PMLR, 18–24 Jul 2021.

[37] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 32–42, October 2021.

[38] Guodong Wang, Shumin Han, Errui Ding, and Di Huang. Student-teacher feature pyramid matching for anomaly detection. *arXiv preprint arXiv:2103.04257*, 2021.

[39] Guoyang Xie, Jinbao Wang, Jiaqi Liu, Jiayi Lyu, Yong Liu, Chengjie Wang, Feng Zheng, and Yaochu Jin. Im-iad: Industrial image anomaly detection benchmark in manufacturing. *IEEE Transactions on Cybernetics*, 54(5):2720–2733, 2024.

[40] Yang Xin, Lingshuang Kong, Zhi Liu, Yuling Chen, Yanmiao Li, Hongliang Zhu, Mingcheng Gao, Haixia Hou, and Chunhua Wang. Machine learning and deep learning methods for cybersecurity. *IEEE Access*, 6:35365–35381, 2018.

[41] Yong-Ho Yoo, Ue-Hwan Kim, and Jong-Hwan Kim. Convolutional recurrent reconstructive network for spatiotemporal anomaly detection in solder paste inspection. *IEEE Transactions on Cybernetics*, 52(6):4688–4700, 2022.

[42] Jiawei Yu, Ye Zheng, Xiang Wang, Wei Li, Yushuang Wu, Rui Zhao, and Liwei Wu. Fastflow: Unsupervised anomaly detection and localization via 2d normalizing flows. *arXiv preprint arXiv:2111.07677*, 2021.

[43] Vitjan Zavrtanik, Matej Kristan, and Danijel Skočaj. Draem - a discriminatively trained reconstruction embedding for surface anomaly detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8330–8339, October 2021.

[44] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *arXiv preprint arXiv:2111.07832*, 2022.

[45] Yang Zou, Jongheon Jeong, Latha Pemula, Dongqing Zhang, and Onkar Dabeer. Spot-the-difference self-supervised pre-training for anomaly detection and segmentation. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 392–408, Cham, 2022. Springer Nature Switzerland.