CHALMERS
UNIVERSITY OF TECHNOLOGY

Distilled Few-Shot Object Detection for Truck Equipment

Distilling a few-shot state-of-the-art model for truck equipment detection utilizing stable diffusion for data augmentation.

Master's thesis in Electrical engineering

ANDREI BORG
MARCUS JOHANSSON

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025

Master's thesis 2025

Distilled Few-Shot Object Detection for Truck Equipment

Distilling a few-shot state-of-the-art model for truck equipment detection utilizing stable diffusion for data augmentation.

ANDREI BORG
MARCUS JOHANSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2025

Distilled Few-Shot Object Detection for Truck Equipment
Distilling a few-shot state-of-the-art model for truck equipment detection utilizing stable diffusion for data augmentation.
ANDREI BORG, MARCUS JOHANSSON

Supervisor and examiner: Fredrik Kahl, Department of Electrical Engineering
Advisor: Anders Björklund, Volvo Group Trucks Technology

Master's Thesis 2025
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Three cropped images of trucks and annotations of their equipment.

Typeset in LaTeX
Gothenburg, Sweden 2025

Distilled Few-Shot Object Detection for Truck Equipment
Distilling a few-shot state-of-the-art model for truck equipment detection utilizing stable diffusion for data augmentation.
ANDREI BORG, MARCUS JOHANSSON
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Few-shot object detection in specialized domains like truck equipment faces significant challenges due to the limited availability of data and the lack of customized models. This study addresses these issues by developing an efficient, streamlined pipeline for identifying truck equipment in constrained situations. We employ knowledge distillation, transferring knowledge from a teacher, the recent CD-ViTO architecture, to a lightweight YOLOv11 student model. We use Stable Diffusion 3.5 for synthetic image generation with prompt engineering to augment the limited training data. The study evaluates the impact of this approach on accuracy and inference speed. The results show that knowledge distillation with generated data significantly boosts the student models' accuracy, particularly in low-shot settings such as 1- and 5-shot, with one configuration outperforming the teacher. Critically, the distilled YOLOv11 student models achieve drastically reduced inference times (83 times speed increase), enabling real-time application. This pipeline demonstrates a successful strategy for creating performant object detection systems in environments with limited data.

Keywords: Few-Shot Object Detection, Knowledge Distillation, Machine Learning, Stable Diffusion, Image Generation, Truck Equipment, Data Augmentation, Real-Time Object Detection, Cross-Domain Vision Transformer, You Only Look Once.

Acknowledgements

This Master's thesis was completed in spring 2025 at Volvo Group Trucks Technology in Gothenburg with the Services team, and we would like to extend our sincere gratitude to several individuals and groups whose support and contributions were invaluable to its completion.

First and foremost, we want to express our sincere appreciation to Fredrik Kahl, our supervisor at Chalmers. His insightful guidance and consistent supervision were crucial throughout the research and writing process.

Our heartfelt thanks go to Anders Björklund, our supervisor at Volvo Group, for his invaluable project direction and unwavering support. His expertise and encouragement were a constant source of motivation.

We are also thankful to Malin Larking, our responsible line manager at Volvo Group, for her efficient handling of administrative matters, her continuous support and her role in the planning stages of this thesis.

Finally, we acknowledge the DSL support staff and the Lundby personnel for their generous provision of GPU resources. These resources were essential for this research and we greatly appreciate their support.

Andrei Borg and Marcus Johansson, Gothenburg, 2025-06-11

List of Acronyms

Below is a list of acronyms that have been used throughout this thesis in alphabetical order:

AI	Artificial Intelligence
C2PSA	CSP with Spatial Attention
C3K2	CSP with Kernel size 2
CD-FSOD	Cross-Domain FSOD
CD-ViTO	Cross-Domain Vision Transformer
CNN	Convolutional Neural Network
CSP	Cross Stage Partial
DE-ViT	Detect Everything Vision Transformer
FPS	Frames Per Second
FSOD	Few-Shot Object Detection
IoU	Intersection over Union
mAP	Mean Average Precision
ML	Machine Learning
ODE	Ordinary Differential Equation
PSA	Partial Spatial Attention
ReLU	Rectified Linear Unit
ROI	Region Of Interest
SOTA	State-of-the-art
SPP	Spatial Pyramid Pooling
SPPF	SPP - Fast
ViT	Vision Transformers
YOLO	You Only Look Once

Contents

List of Figures

List of Tables

1

Introduction

With each passing year, society is becoming more digitalized. Today, cars, surveillance cameras and even kitchen appliances house embedded systems to be more convenient for their users. These systems often have a real-time component and are also computationally constrained, requiring programs to be carefully thought through so that they remain small and are optimized for the given hardware. At the same time, machine learning (ML) and artificial intelligence (AI) models are also being used in more contexts. Using these models in embedded systems requires them to be small and fast, but the most accurate models are often large and slow. Special techniques must be used to minimize the models while maintaining high accuracy. Both training models and minimizing them usually require lots of data, volumes that might not be available, especially in niche environments. In these cases, solutions must be employed to circumvent this problem.

An area that poses the stated challenges is the detection of truck equipment. One major challenge is the lack of data on truck equipment. Here, traditional detection methods struggle because their effective training typically relies on large amounts of annotated data for reliable and precise predictions [1], [2]. When data is scarce, a typical solution is to use few-shot object detection (FSOD). In short, FSOD can generalize to unseen tasks with minimal supervision [3], [4].

Although FSOD has seen improvement in recent years [3], [4], the emphasis has been on accuracy, with run times being considered secondary. In their current state, FSOD models only manage up to 4 frames per second (FPS) [5], which is insufficient for use in real-time applications. To enable deployment in real-time scenarios or onto embedded systems, an ML method called knowledge distillation can be used to transfer knowledge from a big, slow model into a smaller, faster one, called teacher and student, respectively [6]. Furthermore, to ensure sufficient data for the distillation, the dataset can be artificially expanded with an image generation model. An example of such models is diffusion models [7], [8].

1.1 Purpose

This thesis aims to create an end-to-end real-time object detection system for detecting truck equipment. Truck equipment detection presents unique challenges, primarily due to the lack of publicly Available datasets tailored to this niche market. Consequently, researchers face the demanding and time-intensive task of gathering

new data to train ML models. To address the issue of data scarcity, the thesis proposes a novel approach that involves training a real-time object detection model using a small, originally collected dataset. To enhance the model's performance, this dataset will be supplemented with artificially generated data, with the hope to improve its accuracy and efficiency.

1.2 Problem statement

To achieve the project's purpose, a system will be created in a data-restricted environment using knowledge distillation to transfer knowledge from a teacher FSOD model to a student real-time object detection model. The goal is to utilize generated images to improve the student enough to reach or even surpass the teacher. With this goal in mind, this work seeks to answer two questions. The first is determining how the knowledge distillation of a state-of-the-art (SOTA) FSOD impacts accuracy. The second is to find out the distillation's computational performance impact compared to not using it.

Figure 1.1: Example pictures of trucks with several equipment attached. The images were taken from a pedestrian bridge over a local highway.

1.3 Limitations

The final model should, as mentioned, be a compact detector; however, testing it in an embedded setting is not within the project scope and will only be considered if time permits. The only model architecture considered for the larger models used in the project will be the current SOTA FSOD architecture, called Cross-Domain Vision Transformer (CD-ViTO), while the smaller models will be of another architecture that performs well in regular object detection. Furthermore, the project will

focus on detecting the selected classes of truck equipment in images such as the ones in Figure 1.1 and evaluating them against the SOTA model.

1.4 Related work

Several papers treat similar topics to what this thesis is doing. In [9], controlled image generation was used in remote sensing imagery to expand datasets in FSOD settings. Generating images from few examples has been done with techniques like Textual Inversion [10], TINT [11] and DreamBooth [12]. In the work developing TINT, the generated images were used for knowledge distillation in few-shot image classification [11]. Recently, significant progress has been made in FSOD with the development of Detect Everything Vision Transformers (DE-ViT) [5] and the improvements done in CD-ViTO [13]. The usability of object detection models in constrained settings has also seen improvements recently with the eleventh version of the YOLO models, surpassing previous model versions [14].

1.5 Our contributions

This work proposes a method to train real-time object detection models in a few-shot setting, where real-time models are assumed to be ones that can manage processing at least 30 FPS. By distilling knowledge from a SOTA FSOD model to a real-time object detection model using generated images, the accuracy of the student showed a significant increase compared to only fine-tuning it on the real images, even doubling on some occasions. In addition, the student also surpassed the teacher on some tests.

2

Theory

In this chapter, central concepts needed to understand the methods are presented. It begins by addressing the issue of data scarcity in the AI field. As a method for developing models that can function effectively with limited data, few-shot learning is introduced, diving into FSOD and the principal learning strategies that enable it. Following this, the chapter details contemporary object detection techniques, focusing significantly on the You Only Look Once (YOLO) framework. This subsection explores the evolution of YOLO from the first to the latest YOLOv11 iteration. Afterwards, the core attention mechanism is introduced and the impact of transformer architectures on computer vision is explained. Building upon the transformer, the chapter presents the CD-ViTO, a SOTA model for FSOD, highlighting its framework and benchmark performance. Furthermore, the chapter explores the field of image generation, focusing on diffusion models. With this, the principles of forward and backward diffusion processes are discussed, focusing on Stable Diffusion 3.5 and its rectified flow formulation for producing high-fidelity images. In conclusion, the theory chapter outlines key model optimization and data enhancement methodologies. Specifically, it will discuss knowledge distillation, which helps with creating more compact models while maintaining performance, as well as strategies for artificially expanding datasets.

2.1 Few-shot learning

AI applications generally need large amounts of data to yield accurate and generalizable models. In avenues where data is not abundantly available, techniques called few-shot learning have been developed to create models that can manage this obstacle. When these techniques are used in object detection, they're called FSOD. In FSOD, the classes are split into base classes and novel classes. The base classes are abundant, while instances of the novel classes are significantly scarcer.

To solve the problem of detection classes of few instances, different learning techniques have been used to enhance the detection of the novel classes. These can be categorized as transfer learning, meta learning or metric learning [4]. Transfer learning uses weights pretrained on the base classes and then fine-tunes this knowledge with the novel classes [4], [15], [16]. In meta learning, the model learns to generalize to new tasks and data that can then be applied to the desired few-shot task [4], [15], [16]. Metric learning, on the other hand, tries to learn an embedding where similar

contents are close to each other in the embedding while others are far away [4], [16].

2.2 You Only Look Once

The YOLO architecture was introduced in 2016 as a redefined object framework that treated object detection as a single regression problem [17]. Unlike region-based detections that use separate components for proposal generation and classification, YOLO aims to unify these steps within a single Convolutional Neural Network (CNN). With this approach, YOLO achieves real-time object detection while maintaining a high average precision.

Over the years, the YOLO framework has evolved through several versions. Speed, accuracy, or network architecture have improved with each adaptation. To understand the YOLO model, Section 2.2.1 outlines the original architecture design, while Section 2.2.2 provides a brief description of the key additions and innovations in the recent YOLOv11.

2.2.1 Network design

The YOLO network divides the given input image into an $S \times S$ grid, shown in Figure 2.1, where each grid cell predicts B bounding boxes and C class probabilities, $\Pr(Class_i|Object)$ for class i, if the center of an object falls within it. Additionally, each bounding box contains five predicted components: $(x, y, w, h, \text{confidence})$. The (x, y) coordinates denote the box's center relative to the grid cell's bounds, while (w, h) are relative image dimensions. The confidence score for the box is defined as $\Pr(object) \cdot IoU_{pred}^{truth}$ and the scores should result in zero if no object was found in that cell. Whenever an object is predicted, the confidence score is based on the intersection over union (IoU) between the ground truth and the predicted box.

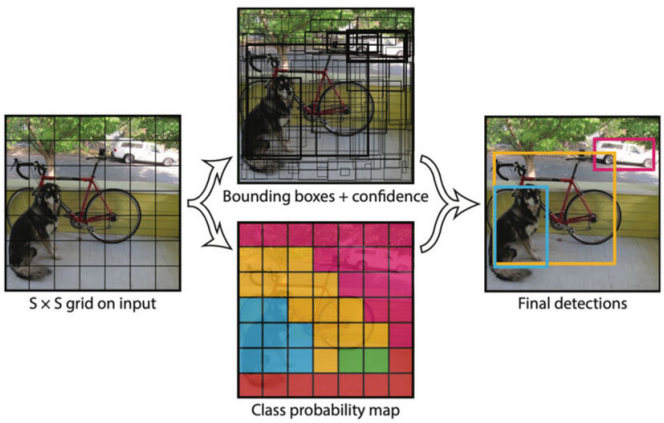


Figure 2.1: YOLO model visualization. The regression approach divides the image into an $S \times S$ grid. Each cell predicts B bounding boxes, their confidence scores and C class probabilities. Due to the bounding boxes containing five components, the output tensor gets the shape $S \times S \times (5B + C)$ [17]. Copyright © 2016, IEEE.

The neural network architecture used in YOLO draws inspiration from GoogLeNet, initially built for image classification [18] and is comprised of 24 convolutional layers followed by two fully connected layers, as shown in Figure 2.2. Convolutional layers extract features, while fully connected layers predict class probabilities and box coordinates. Furthermore, the model includes layers that alternate between 1×1 and 3×3 filters, used for feature space reduction and feature extraction, respectively. Pretraining is done on ImageNet [19], with 224×224 as input resolution for classification and double the resolution for detection. YOLO uses the leaky rectified linear unit (ReLU) as the activation function in all layers except the final output layer.

The leaky ReLU is defined as:

$$\phi(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.1x, & \text{otherwise.} \end{cases} \tag{2.1}$$

The final layer of the network uses a linear activation function, allowing unrestricted outputs for bounding box coordinates and class scores.

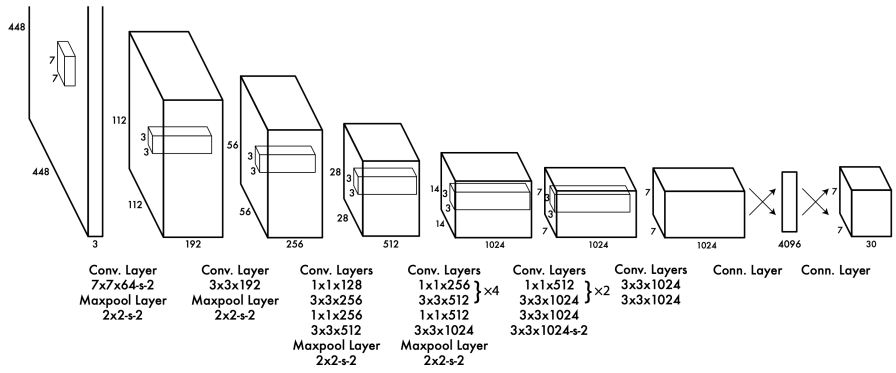


Figure 2.2: YOLO architechture visualization [17]. The detection network consists of 24 convolutional layers and two fully connected layers. It uses alternating 1×1 convolutional layers to reduce the feature space from preceding layers. The convolutional layers are pretrained on ImageNet at a resolution of 224×224 before being adapted to a higher resolution of 448×448 for detection. Copyright © 2016, IEEE.

The loss function in YOLO uses a multi-part sum-squared error loss:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \tag{2.2}$$

$$+\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \right] \tag{2.3}$$

$$+\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \tag{2.4}$$

$$+\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 \tag{2.5}$$

$$+\sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \tag{2.6}$$

where $\lambda_{coord} = 5$ and $\lambda_{noobj} = 0.5$ are used to balance localization and object confidence loss. Furthermore, $\mathbb{1}_i^{obj}$ represents whether an object is present in the cell i, while $\mathbb{1}_{ij}^{obj}$ indicates that the jth bounding box predictor in cell i is responsible for that given prediction.

2.2.2 Key features of the YOLOv11 architecture

The YOLOv11 architecture contains three key components: backbone, neck and head [2], [14]. The backbone extracts features from raw images using CNNs. The neck aggregates and enhances these features across different scales. The head generates the final outputs for object localization and classification based on the refined features. The whole architecture of YOLOv11 is shown in Figure 2.3 and the available model sizes are presented in Table 2.1, where the mean average precision (mAP) at specific thresholds is a metric that measures the model's accuracy at that given threshold.

One of the primary innovations in YOLOv11 is the Cross Stage Partial (CSP) with kernel size 2 (C3K2) block, an evolution of the CSP bottleneck introduced in earlier versions. This architecture uses smaller 3×3 convolutional kernels to reduce computational overhead while maintaining local spatial features. This also enhances the overall efficiency of the architecture in terms of the number of trainable parameters.

Another core component of YOLOv11 is the Spatial Pyramid Pooling - Fast (SPPF) module, which improves upon the traditional SPP block from YOLOv4. The SPPF module utilizes several max-pooling operations with different kernel sizes to collect contextual information at multiple scales. With this approach, even smaller objects are guaranteed to be detected as it effectively merges information from various resolutions.

Furthermore, YOLOv11 introduces a novel attention mechanism called CSP with Spatial Attention (C2PSA). This block incorporates attention mechanisms, where two Partial Spatial Attention (PSA) modules operate on different branches of the

feature map, which are later concatenated. This enhances the models ability to concentrate on significant areas within an image by highlighting spatial relevance in the feature maps. Such areas can include smaller or partially occluded objects.

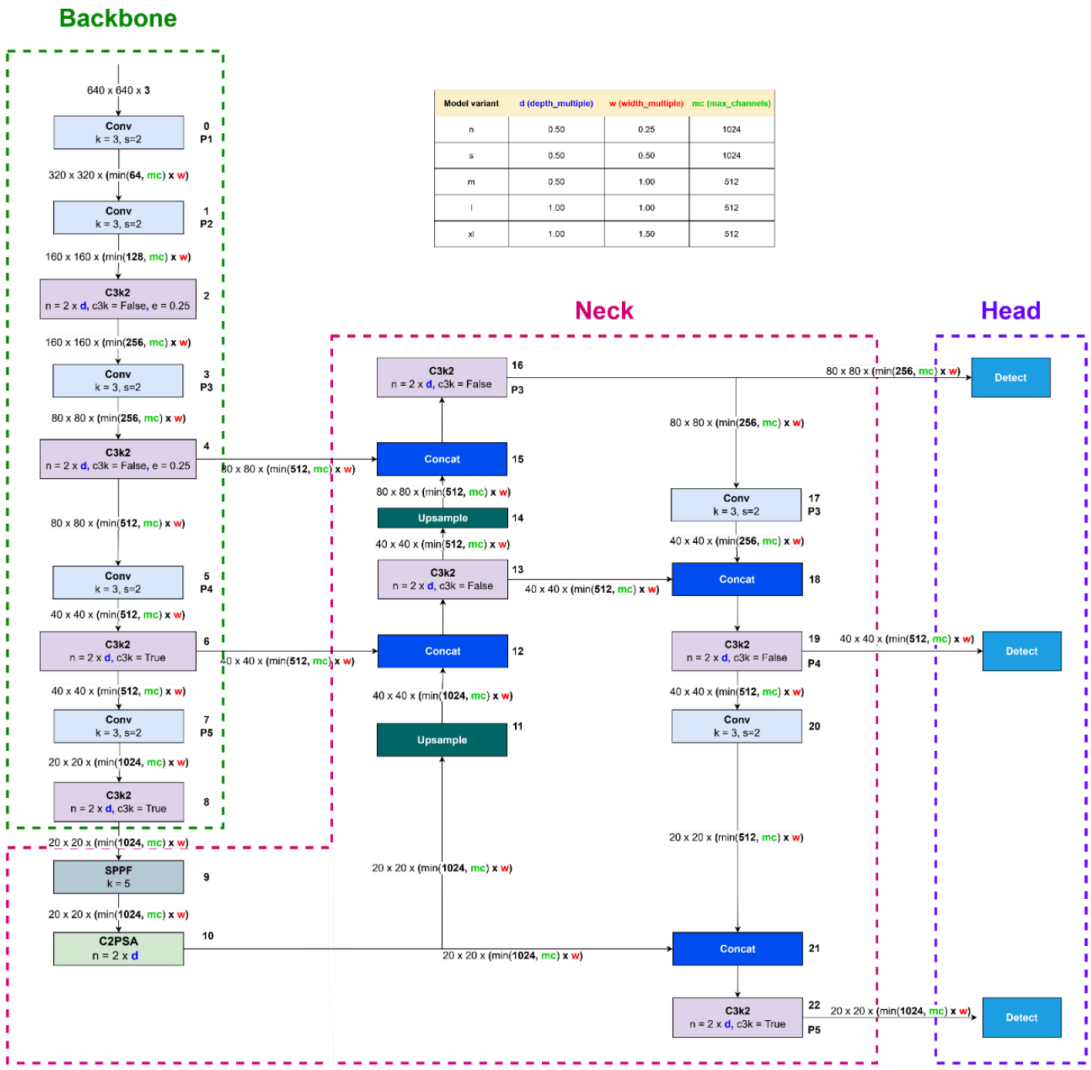


Figure 2.3: YOLOv11 architecture visualization [20].

2.3 Transformers

The following sections describe the theory behind the famous transformer architecture [21], which was initially developed for natural language processing. In computer vision, the transformer architecture has been adapted to vision transformers (ViTs) [22], offering a powerful alternative to traditional CNNs.

2.3.1 Attention

Attention is a method in ML where the relative importance of the separate components of a sequence is determined. There are several methods of calculating attention,

Table 2.1: Performance metrics for object detection for different YOLOv11 model sizes at 640 × 640 resolution [14]. The different sizes include nano (n), small (s), medium (m), large (l) and extra-large (x).

Model	$\mathbf{mAP}^{\mathbf{val}}_{50-95}$	**Speed T4 TensorRT10 (ms)**	**params (M)**	**FLOPs (B)**
YOLOv11n	39.5	1.5 ± 0.0	2.6	6.5
YOLOv11s	47.0	2.5 ± 0.0	9.4	21.5
YOLOv11m	51.5	4.7 ± 0.1	20.1	68.0
YOLOv11l	53.4	6.2 ± 0.1	25.3	86.9
YOLOv11x	54.7	11.3 ± 0.2	56.9	194.9

one of which is the scaled dot-product attention introduced by [22], where the input consists of queries, keys and values of dimensions d_q, d_k and d_v, respectively. and are stacked into matrices Q, K and V, respectively. The attention is then calculated by taking the dot-product of a query and all keys, divide by $\sqrt{d_k}$ and apply a softmax function, defined for $z \in \mathcal{R}^N$ where $N > 1$ as

$$\text{softmax}(z) = \frac{\exp[z]}{\sum_{i=1}^{N} e^{z_i}}. \tag{2.7}$$

The result of the softmax function is then used as weights applied to the values. The attention function then becomes

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V. \tag{2.8}$$

Furthermore, the authors also proposed another method called multi-head attention, where the d_{model}-dimensional queries, keys and values are linearly projected to dimensions d_k, d_k and d_v, respectively, h times by different learned parameters. The attention is then performed in parallel on the projected inputs, giving outputs of dimension d_v that are then concatenated before another learned linear projection is applied. For i^{th} head, the learned projection weight matrices are $W_i^Q \in \mathcal{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathcal{R}^{d_{\text{model}} \times d_k}$ and $W_i^V \in \mathcal{R}^{d_{\text{model}} \times d_v}$ with the parameters for the linear projection of the output given by $W^O \in \mathcal{R}^{hd_v \times d_{\text{model}}}$. Letting $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$ for $i = 1, \ldots, h$, the multi-head attention can be described as

$$\text{Multihead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O. \tag{2.9}$$

The location of the multi-head attention layers in a classic transformer is illustrated in Figure 2.4.

2.3.2 Vision Transformers

Inspired by the success of transformers in natural language processing, [21] adapted the transformer model for use in computer vision. To do this, the image $\mathbf{x} \in \mathcal{R}^{H \times W \times C}$ is reshaped to 2D image patches that are flattened $\mathbf{x}_p \in \mathcal{R}^{N \times (P^2 \cdot C)}$, where H, W and C, are the height, weight and number of channels, respectively, with $N = HW/P^2$ and (P, P) being the number of patches and the size of each patch,

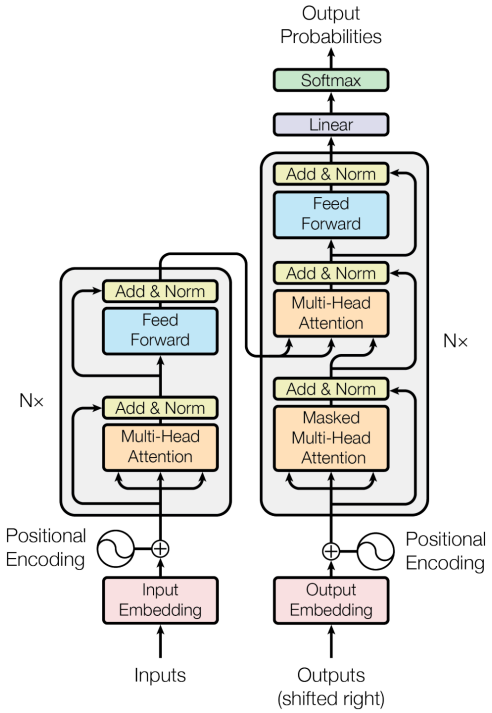


Figure 2.4: An illustration of the basic transformer model architecture [22]. Copyright © 2017, NIPS.

respectively. The patches are then mapped to the dimension D used by the transformers before positional embeddings are added to the patch embedding and then being used as input the transformers.

2.4 Cross-Domain Vision Transformer

The CD-ViTO [13] architecture is an accurate SOTA FSOD model, built upon the previous transformer-based open-set detector DE-ViT [5]. CD-ViTO enhances the generalization ability in larger domain gaps by adding new modules into the DE-ViT architecture. Several novel measures have also been established for evaluating a new benchmark for Cross-Domain FSOD (CD-FSOD). The benchmark results reveal that modern approaches often fail with cross-domain generalization and show that the proposed techniques collectively contribute to an improved cross-domain model. Figure 2.5 shows a technical motivation for the CD-ViTO model.

2.4.1 Framework overview

CD-ViTO proposes a new CD-FSOD benchmark, shown in Figure 2.6, to address whether open-set detection methods generalize to cross-domain target datasets without a decrease in performance. In the benchmark, several object detection datasets are combined: COCO as a source training set [23], ArTaxOr [24], Clipart1K [25], DIOR [26], DeepFish [27], NEU-DET [28] and UODD [29] as novel datasets.

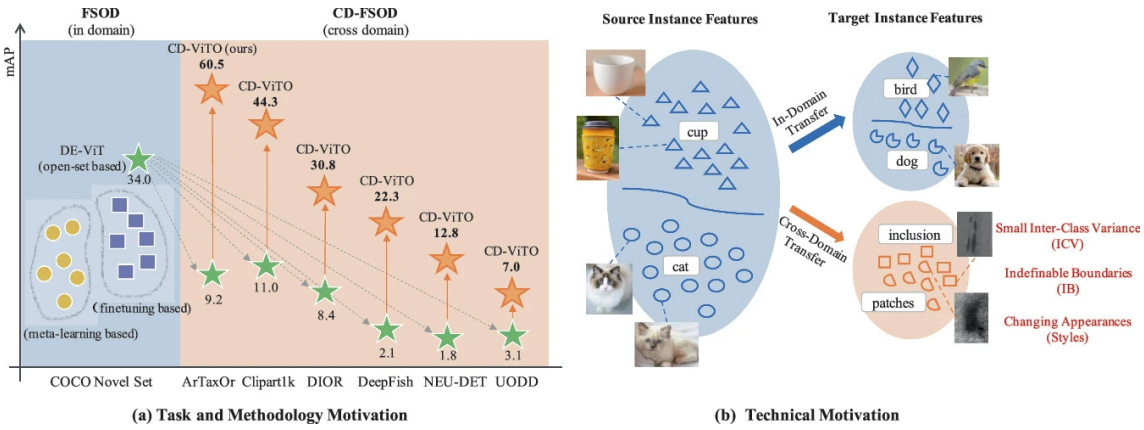


Figure 2.5: CD-ViTO methodology and technical motivation [13].

The authors also introduced the three metrics Inter-Class Variance (ICV), Indefinable Boundaries (IB) and style, to capture and benchmark challenges in cross-domain scenarios. The style metric includes domain-related style differences, where some common styles include cartoon, sketch, photorealistic, etc. ICV is a commonly utilized measure in learning that assesses the differences between classes, with elevated ICV values suggesting that semantic labels are recognized more easily. Datasets with coarse granularity, such as COCO, typically demonstrate higher ICV, while datasets with finer granularity show lower ICV values. Lastly, IB refers to the confusion between a target object and its background when identifying it. For instance, spotting a person against a simple background is easy compared to identifying a fish in a coral reef.

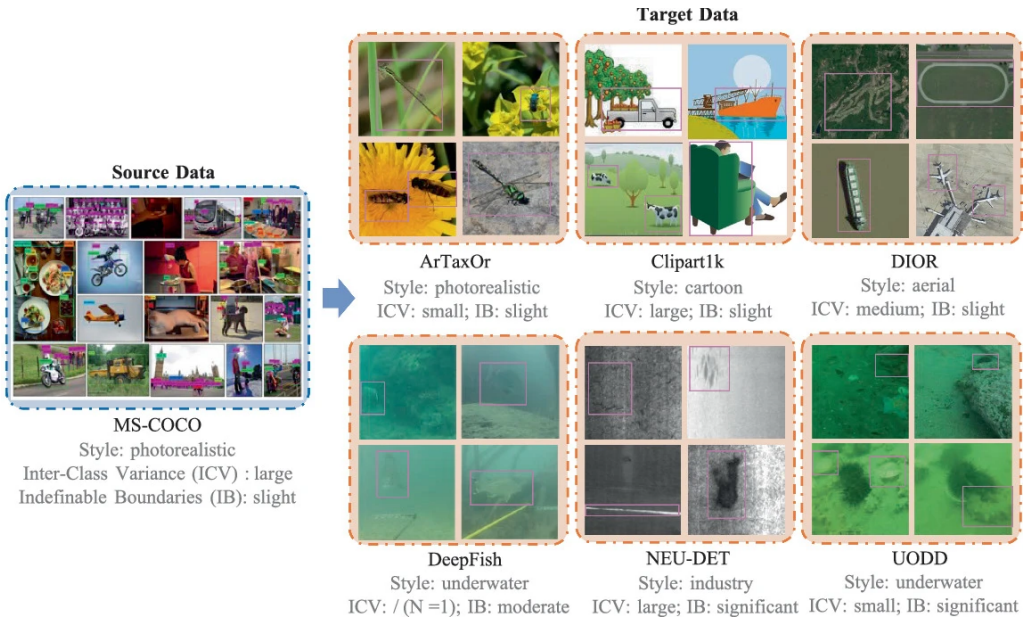


Figure 2.6: The CD-FSOD benchmark consists of the COCO [23] dataset as the training data source and six other datasets are used as novel target datasets [13]. All datasets contain unique variations of the metrics style, ICV and IB.

To enhance the open-set methods present in the DE-ViT detector, CD-ViTO im-

plements three new modules: learnable instance features, an instance reweighting module and a domain prompter, see section 2.4.2 for detailed descriptions. The whole pipeline, including the novel modules, is visualized in Figure 2.7. Blue modules and arrows represent the original DE-ViT architecture, while new additions are shown in orange. The whole architecture includes the following parts: a pretrained DINOv2 ViT [30], a region proposal network M_{RPN}, an region of interest (ROI) align module M_{ROI}, a detection head M_{DET}, a one-vs-rest classification head M_{CLS}, as well as the learnable instance features module, the instance reweighting module and the domain prompter.

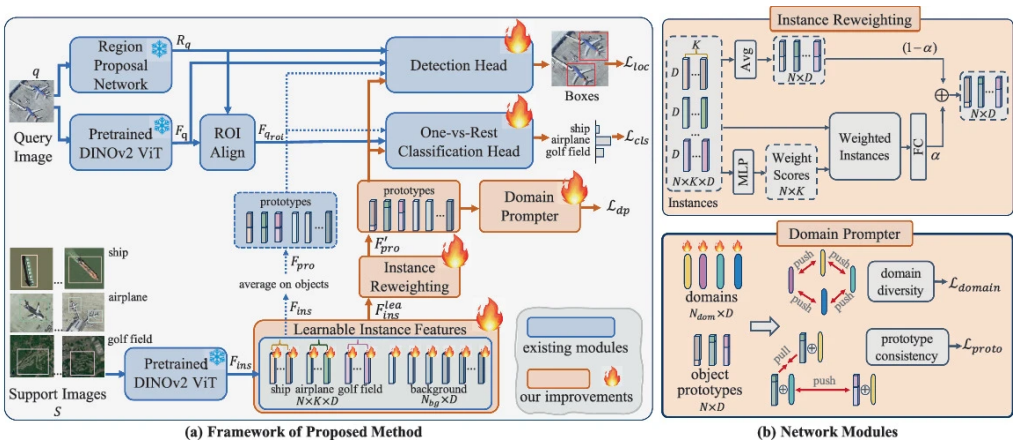


Figure 2.7: CD-ViTO framework visualization [13]. The framework is based on the open-set detector DE-ViT, with blue modules being inherited from DE-ViT while new additions are colored as orange modules. Novel modules include learnable instance features, instance reweighting, domain prompter and finetuning.

In short, a query image q is analyzed by DINOv2, M_{RPN} and the M_{ROI} to extract the region proposals R_q, visual features F_q and ROI features $F_{q_{roi}}$. The M_{DET} head then handles localization, while M_{CLS} performs classification. The complete localization and classification task is supervised by $\mathcal{L}_{loc}$ and $\mathcal{L}_{cls}$, respectively.

In the meantime, the lower path of the CD-ViTO framework uses support images S to extract instance features F_{ins} with the pretrained DINOv2 ViT model. These are then adapted to F_{ins}^{lea} and made learnable with the learnable instance features module. With the instance reweighting module, object weights in F_{ins}^{lea} are reweighted, forming $F_{pro}^{'}$. In addition to the previous two novel modules, the domain prompter ensures cross-domain robustness and is supervised by $\mathcal{L}_{dp}$. Every parameter (over 350 million) is collectively optimized by the object $\mathcal{L}$ function:

$$\mathcal{L} = \mathcal{L}_{loc} + \mathcal{L}_{cls} + \mathcal{L}_{dp}. \tag{2.10}$$

2.4.2 Key components

The first component is the learnable instance features (M_{LIF}) module. This component addresses the challenge of small ICV by improving the separability of instance features across different classes. Given a set of precalculated instance fea-

tures $F_{ins} \in \mathcal{R}^{(N\times K+N_{bg})}$, where N is the number of object classes, K is the number of shots, N_{bg} is the number of background instances and D is the visual feature dimension, the M_{LIF} module initializes a learnable matrix F_{ins}^{lea} using F_{ins}. This learnable matrix is then optimized using the standard localization loss $\mathcal{L}_{loc}$ and classification loss $\mathcal{L}_{cls}$. In short, the primary objective is to align these instance features more closely with their corresponding categories, thereby reducing confusion among classes.

The second module, instance reweighting (M_{IR}), is introduced to address the variability in instance quality, particularly in cases of significant IB. After applying M_{LIF}, the object-related features are reshaped into $F_{ins}^{0} \in \mathcal{R}^{N\times K\times D}$. As seen in Figure 2.7, M_{IR} contains a two-way path connected in a residual way. The lower path takes F_{ins}^{0} and feeds it through a multilayer perceptron (MLP) to extract a weighted score, which is then used to acquire a weighted sum $F_{pro}^{att} \in \mathcal{R}^{N\times D}$ on the original prototypes. The upper path is a simple average calculation on F_{ins}^{0}, resulting in $F_{pro}^{avg} \in \mathcal{R}^{N\times D}$. The combined paths together compute the final objective prototype:

$$F_{pro}^{ob'} = \alpha fc(F_{pro}^{att}) + (1-\alpha)F_{pro}^{avg}, \tag{2.11}$$

where $fc(\cdot)$ is a fully connected layer and $\alpha \in [0,1]$ is a hyperparameter. Together with the background features, the M_{IR} module outputs the formed prototype $F_{pro}^{'}$.

The third component is the domain prompter (M_{DP}), which aims to improve cross-domain robustness by introducing synthetic domain perturbations. The model defines a set of learnable domain vectors $F_{domain} \in \mathcal{R}^{N_{dom}\times D}$, where N_{dom} is a hyperparameter specifying the number of virtual domains. These vectors are added to type object prototypes $F_{pro}^{ob'}$ to generate perturbed prototypes. To train these domain vectors, a domain diversity loss $\mathcal{L}_{domain}$ is applied to ensure that domain vectors e.g., f^{d_i} and f^{d_j} are distinct from each other. The $\mathcal{L}_{domain}$ is implemented by using a InfoNCE-style loss:

$$\mathcal{L}_{domain} = -\frac{1}{N_{dom}} \sum_{i=1}^{N_{dom}} \left(\log \frac{\exp(f^{d_i} \cdot f^{d_i}/\tau)}{\sum_{j=1}^{N_{dom}} \exp(f^{d_i} \cdot f^{d_j}/\tau)} \right), \tag{2.12}$$

where the hyperparameter τ regulates the temperature. Similarly, a $\mathcal{L}_{proto}$ loss is introduced to randomly perturb a prototype f_{p_i}, taken from $F_{pro}^{ob'}$, with two different domains f^{d_k} and f^{d_m} sampled from F_{domain}. This creates the two "confused" prototypes $f_{p_i}^{d_k}$ and $f_{p_i}^{d_m}$, which is achieved by adding together the features e.g., $f_{p_i}^{d_k} = f_{p_i} + f^{d_k}$, hence the formula:

$$\mathcal{L}_{proto} = -\frac{1}{N} \sum_{i=1}^{N} \left(\log \frac{\exp(f_{p_i}^{d_k} \cdot f_{p_i}^{d_m}/\tau)}{\sum_{j=1}^{N} \exp(f_{p_i}^{d_k} \cdot f_{p_j}^{d_m}/\tau)} \right). \tag{2.13}$$

Finally, along the two losses $\mathcal{L}_{domain}$ and $\mathcal{L}_{proto}$, a third (cross-entropy) loss $\mathcal{L}_{proto_{cls}}$ is applied to perturbed features to ensure they retain correct class semantics. Together, the three losses $\mathcal{L}_{domain}$, $\mathcal{L}_{proto}$ and $\mathcal{L}_{proto_{cls}}$ form the full domain prompt loss:

$$\mathcal{L}_{dp} = \mathcal{L}_{domain} + \mathcal{L}_{proto} + \mathcal{L}_{proto_{cls}}. \tag{2.14}$$

2.4.3 Benchmarks

The hyperparameters in CD-ViTO are based on DE-ViT. For all the target novel datasets the hyperparameter N_{bg}, α ratio of M_{IR} module, τ temperature for $\mathcal{L}_{\text{proto}}$, τ temperature for $\mathcal{L}_{\text{domain}}$ are set as 530, 0.7, 2, 0.1, respectively. The value N_{dom} is set as $2 * N$ and refers to the number of virtual domains, depending on classes N. The Top-K hyperparameter is set as 5. The trainable parameters on 1-shot are tuned around 80 epochs, and for the 5- and 10-shot, they are tuned around 40 epochs. The SGD optimizer has a learning rate of 0.002. A single augmentation technique of random flipping is used. Four RTX 3090 GPUs were used to conduct the experiments.

CD-ViTO achieves SOTA performance on several novel datasets, as shown in Table 2.2. As the k-shot grows, the performance increases. Features are most effectively embedded with the largest ViT-L/14 backbone.

Table 2.2: The 1/5/10-shot (mAP for novel classes) results of CD-ViTO on six novel target datasets [13]. The average result is denoted as "Avg.".

	Backbone	ArTaxOr	Clipart1k	DIOR	DeepFish	NEU-DET	UODD	Avg.
1-shot	ViT-S/14	16.4	7.8	14.0	3.3	3.8	2.6	8.0
	ViT-B/14	13.6	13.6	15.7	13.2	**4.0**	2.4	10.4
	ViT-L/14	**21.0**	**17.7**	**17.8**	**20.3**	3.6	**3.1**	**13.9**
5-shot	ViT-S/14	36.2	25.5	20.4	20.4	10.5	6.1	19.9
	ViT-B/14	43.1	36.3	22.8	20.2	10.7	**7.4**	23.4
	ViT-L/14	**47.9**	**41.1**	**26.9**	**22.3**	**11.4**	6.8	**26.1**
10-shot	ViT-S/14	42.5	29.8	23.6	21.2	12.8	5.9	22.6
	ViT-B/14	54.8	40.4	27.4	19.5	**13.1**	6.2	26.9
	ViT-L/14	**60.5**	**44.3**	**30.8**	**22.3**	12.8	**7.0**	**29.6**

2.5 Diffusion models

Generating images can be done in a variety of different ways. One of which is with diffusion models [31], [32]. These models are characterized by a forward and backward process. The forward process takes an initial sample x_0 from a data distribution p_0 and incrementally applies noise at time steps $t \in [0, T]$ until the distribution converges to a sample x_T of a known noise distribution p_T, the prior state x_T. The backwards process tries to do the opposite, i.e. iteratively recover the original image through a de-noising process. The de-noising process uses a model, typically a neural network that is trained to acquire the original images, or ones looking similar to the originals.

2.5.1 Stable Diffusion 3.5

A diffusion model that is currently one of the best open source image generation models is Stable Diffusion 3.5 [33]. The models use a rectified flow formulation of the forward process, which associates data and noise through a straight line.

For these models, the mapping between x_1 and x_0 is defined in terms of an ordinary differential equation (ODE), i.e.

$$dy_t = v_\Theta(y_t, t)dt \tag{2.15}$$

where $t \in [0, 1]$ with v being parameterized by the neural network weights Θ. To solve the ODE in an efficient manner, a vector field u_t is regressed that generates a probability path between p_0 and p_1.

To create u_t, a forward process for a probability path p_t between p_0 and $p_1 = \mathcal{N}(0, 1)$ is defined as

$$z_t = a_t x_0 + b_t \epsilon \tag{2.16}$$

where $\epsilon \sim \mathcal{N}(0, I)$. Setting $a_0 = 1$, $b_0 = 0$, $a_1 = 0$ and $b_1 = 1$ make the marginals

$$p_t(z_t) = E_{\epsilon \sim \mathcal{N}(0,I)} p_t(z_t|\epsilon) \tag{2.17}$$

consistent with p_0 and p_1. The relationship between z_t, x_0 and ϵ can be described by ψ_t and u_t, which are defined as

$$\psi_t(\cdot|\epsilon) : x_0 \mapsto a_t x_0 + b_t \epsilon \tag{2.18}$$

$$u_t(z|\epsilon) := \psi_t'(\psi_t^{-1}(z|\epsilon)|\epsilon). \tag{2.19}$$

From these, the ODE $z_t' = u_t(z_t|\epsilon)$ can be constructed, which has a solution z_t with initial value $z_0 = x_0$. Due to this solution, $u_t(\cdot|\epsilon)$ generates $p_t(\cdot|\epsilon)$. The conditional vector field can then be regressed using the conditional flow matching

$$\mathcal{L}_{CFM} = E_{t, p_t(z|\epsilon), p(\epsilon)} ||v_\Theta(z, t) - u_t(z|\epsilon)||_2^2 \tag{2.20}$$

as objective. Inserting $\psi_t'(x_0|\epsilon) = a_t' x_0 + b_t' \epsilon$ and $\psi_t^{-1}(z|\epsilon) = \frac{z - b_t \epsilon}{a_t}$ into Equation 2.19 to get the loss in an explicit form yields

$$z_t' = u_t(z_t|\epsilon) = \frac{a_t'}{a_t} z_t - \epsilon b_t \left(\frac{a_t'}{a_t} - \frac{b_t'}{b_t} \right). \tag{2.21}$$

This can then be rewritten with a signal-to-noise ratio $\lambda_t := \log \frac{a_t^2}{b_t^2}$ and $\lambda_t' = 2(\frac{a_t'}{a_t} - \frac{b_t'}{b_t})$ as

$$u_t(z|\epsilon) = \frac{a_t'}{a_t} z_t - \frac{b_t}{2} \lambda_t' \epsilon, \tag{2.22}$$

which is used to re-parameterize Equation 2.5.1 as the noise prediction objective

$$\mathcal{L}_{CFM} = E_{t, p_t(z|\epsilon), p(\epsilon)} ||v_\Theta(z, t) - \frac{a_t'}{a_t} z + \frac{b_t}{2} \lambda_t' \epsilon||_2^2 \tag{2.23}$$

$$= E_{t, p_t(z|\epsilon), p(\epsilon)} \left(- \frac{b_t}{2} \lambda_t' \right)^2 ||\epsilon_\Theta(z, t) - \epsilon||_2^2 \tag{2.24}$$

with $\epsilon_\Theta := \frac{-2}{\lambda_t' b_t}(v_\Theta - \frac{a_t'}{a_t} z)$. To introduce time dependent weighting w_t, the equations were rewritten as

$$\mathcal{L}_w(x_0) = -\frac{1}{2} E_{t\sim\mathcal{U}(t),\epsilon\sim\mathcal{N}(0,I)}[w_t \lambda_t' ||\epsilon_\Theta(\ddagger, t) - \epsilon||^2] \tag{2.25}$$

where $w_t = -\frac{1}{2}\lambda_t' b_t^2$ gives $\mathcal{L}_w = \mathcal{L}_{CFM}$.

Several different flow trajectories were tested in [34], with the best performing one being the rectified flow. Rectified flow defines the path between p_0 and p_1 as

$$z_t = (1-t)x_0 + t\epsilon \tag{2.26}$$

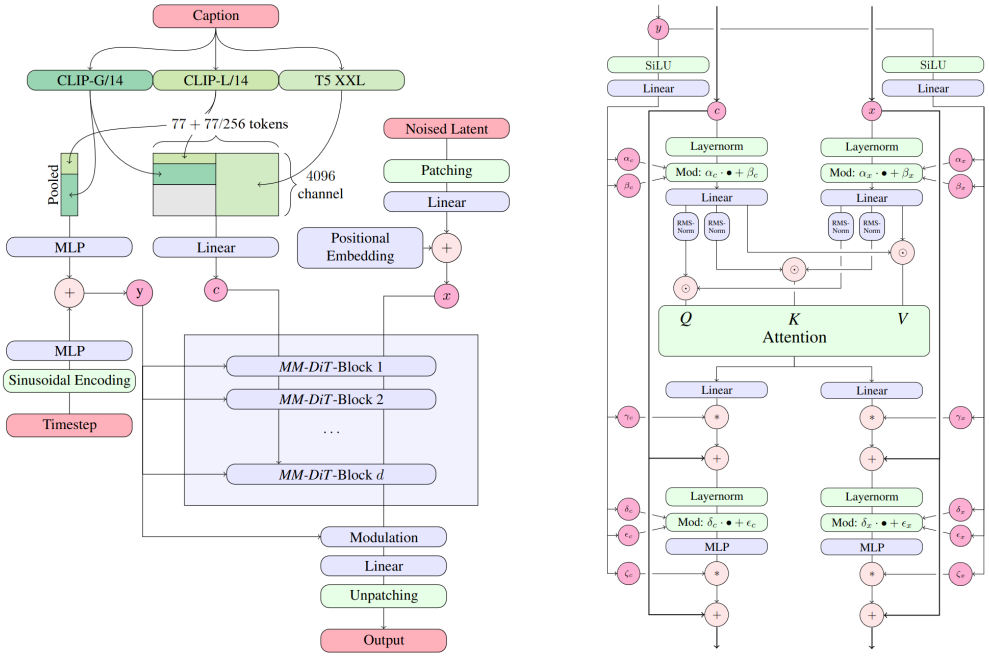
which corresponds to weights $w_t = \frac{t}{1-t}$ in $\mathcal{L}_w$. For the models, predicting the velocity between $t = 0$ and $t = 1$ is harder in general than the two bounds. To address this the weighted loss

$$w_t^\pi = \frac{t}{1-t}\pi(t) \tag{2.27}$$

was introduced, which is equivalent to change the uniform distribution $\mathcal{U}(t)$ to a density $\pi(t)$. The density generating the best results was the logit-normal distribution

$$\pi_{ln}(t; m, s) = \frac{1}{s\sqrt{2\pi}} \frac{1}{t(1-t)} \exp\left(-\frac{(\text{logit}(t) - m)^2}{2s^2}\right) \tag{2.28}$$

with $\text{logit}(t) = \frac{t}{1-t}$, location parameter m and scale parameter s. The architecture of the models used is shown in Figure 2.8.



(a) Overview of all components.

(b) One *MM-DiT* block

Figure 2.8: The model architecture of Stable Diffusion 3 [34]. Copyright © 2024, ICML.

2.6 Knowledge distillation

SOTA ML models are often large and complex and thus also require significant computing resources to use. To decrease the computing resources needed, knowledge distillation techniques have been developed. These aim to transfer knowledge of a large and complex model to a less complex one that doesn't require as many resources to use. The former model is commonly called the teacher, while the latter is called the student. There are different types of knowledge according to [6], one of which is called response-based knowledge.

Response-based knowledge usually means the logits of the last layer of the teacher. The idea of using this knowledge is to make the student imitate the output of the teacher and thus also it's predictions.

2.7 Data augmentation

The performance achieved by ML models is heavily influenced by the quality of the data used in training. The data used should represent the general case and be unbiased, but this is difficult to achieve. To improve the generalization and counteract biases in the data, data augmentation can be performed. Data augmentations are techniques that create new data points through modifying already existing data. Augmenting the data increases its diversity, which leads to the models trained on it being more robust.

When doing object detection, the augmentation done to the images can be categorized as techniques based on geometric transformations, color transformations, random occlusions and deep learning [35]. Geometric transformations change image pixel positions by applying rotations, flipping, scaling, cropping etc. Color transformations on the other hand change the color values of the pixels and includes changing the brightness, contrast, hue, saturation or noise in the image. Random occlusions takes part of the images and cover them, either by erasing the chosen part, or by covering it with another object. Several augmentations can be seen in Figure 2.9.

Figure 2.9: Image augmentation examples [35]. Copyright © 2021, IEEE.

3

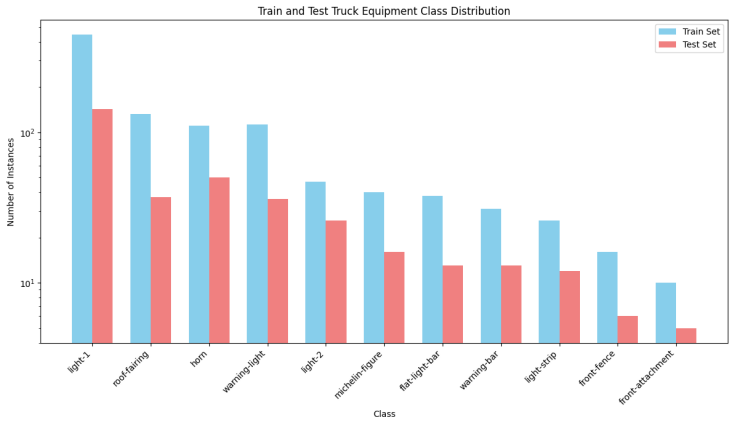
Methods

To tackle the challenges of computational limits and data scarcity in FSOD, knowledge distillation is used as the key technique. This section outlines the proposed distillation framework. It starts with the data collection, where pictures of trucks were captured in public with a system camera before being anonymized and annotated. Then, the training is presented, explaining the procedure for both teacher and student models. The system architecture comes next, where the design of the system is presented. In it, a model is used to detect a truck and then crop the image to only contain the truck before passing it to the next model, which detects its equipment. The models used for these detections are treated in the section as well as the model and prompts used for the image generation. In the last part of the method, the experiments conducted are explained together with their respective purpose.

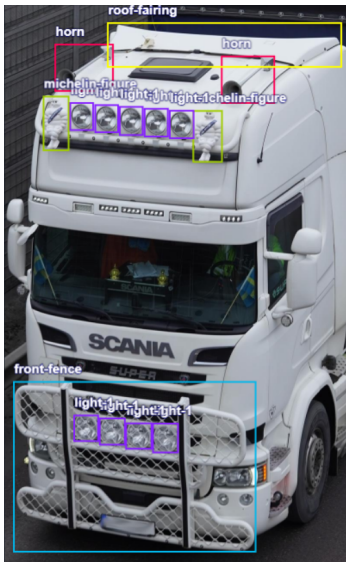
3.1 Data collection

As mentioned in Section 2.1, there are two sets of classes in few-shot learning, one containing base classes and one containing novel classes. The base set used for train the teacher, CD-ViTO, was the COCO [23] dataset, while the novel set consisted of data mainly collected by photographing trucks in public, with a few located in private locations. Most bypassing trucks were photographed from a pedestrian bridge over a highway during the day to ensure a clean photo collection. This way, no light from headlights could disrupt the quality of the images. In total, over 300 pictures were taken to ensure all classes were present, enabling both 10-shot splits and a test set featuring a wide spread of all classes. The full-sized pictures were captured at a resolution of 3368×6000 pixels with a Sony A6400 mirrorless camera, with Figure 1.1 showing three image examples.

Once all images were collected for the novel set, distinguishable faces and license plates were blurred to protect the integrity of the people found in the images. The images were then annotated in Roboflow [36] with the 11 desired classes, which are presented in Figure 3.2. Furthermore, a separate dataset was annotated to teach a YOLO model to crop truck cab fronts, effectively zooming in to the relevant part of an image with attached truck equipment. The class distribution and an annotation example of a cropped truck can be seen in Figure 3.1.



(a) Train and test class distribution (log scale).



(b) Example image with annotations.

Figure 3.1: Combined figure showing train and test truck equipment class distribution on a logarithmic scale and an annotated example from the cropped dataset.

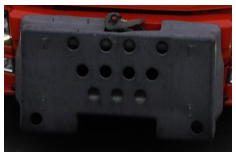
3.2 Training

ML models usually require large amounts of data to train on for the models to be usable. The same is true for knowledge distillation, which was used to get a smaller and faster model. To avoid these problems, a model specialized in FSOD tasks was used as a teacher. This model was pre-trained on the base dataset and then fine-tuned on the novel data. Images containing the novel classes were then generated by an image generation model for the distillation. After being generated, the objects in the images were pseudo-labeled by the teacher. The generated data with their pseudo labels were then used together with the real novel data to fine-tune the student model. A model for detecting trucks was also trained, but considering that there is an abundance of truck datasets online, all trucks found in the collected data were used to train the model.

3.3 System architecture

To enable efficient detection of the desired objects, a specific architecture was constructed for the system. First, to enable easier detection of truck equipment, a model specialized in truck cab detection is used to detect the truck cabs in the input image. This model was relatively small to keep the overall runtime of the system low. The bounding boxes for the truck cab detections are then used to crop the image to the individual trucks. The cropped trucks are then sent to another object detection model trained to detect the individual equipment attached to the trucks. An illustration showing the pipeline is presented in Figure 3.3.

a Flat light bar



b Front attachment

c Front fence

d Horn

e Light strip

f Light type 1

g Light type 2

h Michelin figure

i Warning bar

j Roof fairing

k Warning light

l Truck cab front

Figure 3.2: All 11 truck equipment classes are shown in figures a-k and a truck cab front class figure l.

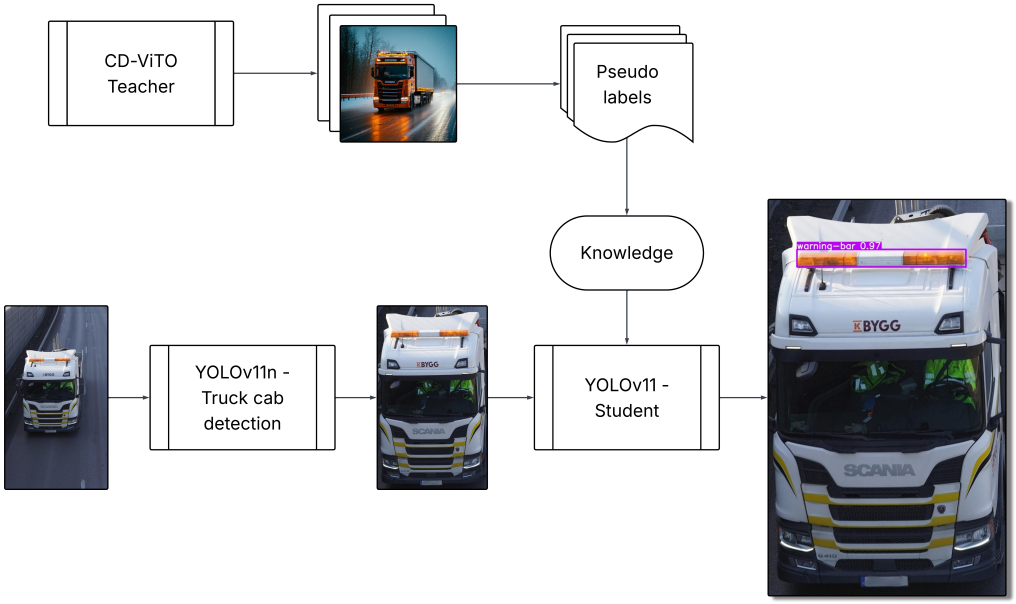


Figure 3.3: Inference pipeline of the proposed method.

3.3.1 Truck cab detection

In the images given as input to the truck cab detection model, the trucks are expected to be easily distinguished from the background without any other objects blocking the camera's line of sight, as Figure 3.4 shows. Based on this, a small and simple model can be used to detect the trucks. The model should also be fast to avoid a bottleneck in the complete pipeline. The YOLOv11 model, which possesses these properties, was chosen for this task. A total of 300 images were used to train the model, with rotation augmentations applied to increase the model's ability to generalize to unseen pictures. Most of the hyperparameters were set to the default values, except for the epochs, which were set to 25. For detailed descriptions of all default parameters, visit [14].

3.3.2 Teacher

In contrast to truck detection, equipment detection is significantly more challenging. This difficulty arises from the smaller size of the objects and the limited number of training examples available. Consequently, the selected model for the equipment detection task was CD-ViTO, which is currently considered the SOTA in FSOD. The model has three variants of varying size, a small, base and large, of which the variant with the highest mAP50 truck detection experiment is presented in Section 3.4. Hyperparameters are set to the default CD-ViTO parameters, with only random flipping used as an augmentation technique because the model performs them by default. See theory Section 2.4.3 for more details about the hyperparameters.

Figure 3.4: A cab front annotation example.

3.3.3 Student

The same model was, however, not used for the student network due to CD-ViTO and its predecessor, DE-ViT, being too slow for real-time applications and the models not seeing much accuracy improvement between 10 and 30-shot [13] and above 30-shot [5], respectively. Therefore, the lack of improvement would likely make the generated images superfluous with these models. Instead, YOLOv11 models were also used here because there is now a lot of data, with the real and generated data combined, to use during training. Once again, the default hyperparameters were used for the YOLOv11 student models, except for setting the epoch count to 25 and removing all augmentation techniques except random flipping to match the teacher's augmentation approach.

3.3.4 Image generation

Stable Diffusion 3.5 Large was used to generate additional images for the distillation. This is because it is among the best image generation models currently that is also open source [33]. The large model variant was chosen over its siblings, the large turbo and the medium, because of the superior quality of the images generated. The amount of VRAM needed was not considered an issue, nor was the speed of the generations.

For the model to generate images of trucks with the desired equipment, one prompt was created for each class. The prompts are presented in Table 3.1 and were chosen so that the model would generate images as close to the novel data as possible. The collected data was captured from a bridge looking over a highway and the prompts were therefore started with "A photo taken at a downward angle..." or "A photo taken from above...". The prompt is then continued for most classes by "...an entire lorry..." because some images would otherwise only contain half a truck. The prompt

then describes the equipment and where it should be placed on the truck. Some examples of images with generated equipment classes are shown in Figure 3.5.

(a) Michelin figure. (b) Warning light. (c) Horn. (d) Front fence.

Figure 3.5: Generated truck images at a 1024×1024 resolution.

3.4 Experiments

To evaluate the performance of the developed system, a variety of model configurations were tested on the test set, which consisted of 50 images. These include different sizes of CD-ViTO, YOLOv11 models trained on only real images as well as YOLOv11 models trained on both real and generated images. Several generated class instances were collected, with the distribution of 10, 25, 50 and 100 instances being tested in total. Several comparisons were then made to determine which models are the best fit for each task and how their performance differs.

Determining the model to use as the teacher was done through comparing the mAP50 of the small, base or large variant of CD-ViTO. The effect of training on cropped images, in contrast to the full ones, was also explored for these models. The mAP50 of the different YOLOv11 models on the cropped images was also used to compare the models' performance with only fine-tuning on the real images and the performance using the generated ones. The inference time of the models was also measured to determine the speedup of the student models compared to the teacher.

Table 3.1: Shows equipment classes and the corresponding prompt used as input to generate images with the equipment included.

Class	**Prompt**
Roof fairing	A photo taken at a downward angle of a lorry.
Light 1	A photo taken at a downward angle of an entire lorry with lights attached to the grille.
Light 2	A photo taken at a downward angle of an entire lorry with white lights on the cab.
Warning light	A photo taken at a downward angle of an entire lorry with two warning lights on the cab.
Light strip	A photo taken at a downward angle of an entire lorry with a white LED strip on the cab.
Flat light bar	A photo taken at a downward angle of an entire lorry with a light strip on the cab.
Front attachment	A photo taken at a downward angle of an entire lorry with an attachment module on the grille.
Front fence	A photo taken at a downward angle of an entire lorry with a grille guard.
Horn	A photo taken at a downward angle of an entire lorry with truck air horns on the cab.
Michelin figure	A photo from above of a lorry with two michelin men on the cab.
Warning bar	A photo taken at a downward angle of an entire lorry with an orange LED strip on the cab.

4

Results

This chapter presents the results gathered from performing the experiments described in Section 3.4 with the aim of showcasing how the presented methods impact the quality of the detections and the inference time of the system. The experiments performed on the CD-ViTO models show that using cropped images instead of full ones greatly increases the mAP50 across all model variants, with the base model performing the best on the cropped images over all few-shot variants tested. Testing the YOLOv11 models on real, cropped images showed that the larger models generally perform better, with the exception of YOLOv11x performing worse than some of the smaller models in the 5- and 10-shot scenarios. Furthermore, the inference times of the YOLOv11 models were in the single-digit milliseconds, while CD-ViTO's were longer than half a second.

4.1 CD-ViTO's accuracy on cropped and full-scale images

The results of the CD-ViTO testing on the full scale and cropped images are shown in Table 4.1. The table shows that cropping the images increases the mAP50 of the models by between 9.54 and 32.4, often doubling the score. It also shows ViT-L/14 being the best-performing model in the 1- and 5-shot tests on full-scale images with mAP50s of 13.9 and 25.7, respectively. ViT-B/14 comes in closely behind on 1-shot, being only 0.6 mAP50 under. In the full-scale 10-shot ViT-B/14 performed the best with an mAP50 of 37.3 and on all the cropped tests, with 31.7, 51.6 and 59.0 mAP50, on the 1-, 5- and 10-shot, respectively, making ViT-B/14 on cropped images the best overall performer.

4.2 YOLOv11's accuracy on cropped images

The accuracy of YOLOv11 was tested in different few-shot settings on cropped images and the results are presented in Table 4.2. For the 1-shot test, the larger the models tested, the better the accuracy, with the largest model, YOLOv11x having an mAP50 of 29.1. Furthermore, the medium and large models performed similarly, only differing by 0.2. In both the 5- and 10-shot tests, the large model had the best mAP50 with 50.1 and 57.2 in the respective tests. The larger models performed better here for the most part, except for the extra-large model YOLOv11x. In the

Table 4.1: The 1/5/10-shot mAP50 results for CD-ViTO on the truck equipment dataset's two versions, featuring full-scale and cropped images. Bold values highlight the best backbone configuration for each k-shot and dataset variant, while the highlighted value in blue shows the best performing combination overall.

	CD-ViTO	Full-scale	Cropped
1-shot	ViT-S/14	9.26	18.8
	ViT-B/14	13.3	**31.7**
	ViT-L/14	**13.9**	26.2
5-shot	ViT-S/14	17.5	44.2
	ViT-B/14	19.2	**51.6**
	ViT-L/14	**25.7**	47.1
10-shot	ViT-S/14	31.5	53.8
	ViT-B/14	**37.3**	**59.0**
	ViT-L/14	35.8	57.3

5-shot setting, the model had an mAP50 below the medium model, while it was below the small one in the 10-shot setting.

4.3 YOLOv11's knowledge distillation performance

The resulting mAP50s from YOLOv11n/s/m after distilling knowledge from CD-ViTO are presented in Table 4.3 along with the differences compared to the data in Table 4.2 in parentheses. The table presents results for $N = 10, 25, 50, 100$ generated instances used in the distillation process. The best results seen for each few-shot setting, for all classes, were achieved with YOLOv11s and were 30.6 mAP50 with $N = 50$ for 1-shot, 53.8 with $N = 10$ for 5-shot and 57.6 with $N = 10$ for 10-shot. While those were the best results achieved by the small and all models, the nano model performed best for the three corresponding shots with $N = 25, 25, 10$ with mAP50s being 23.1, 41.9 and 50.7. On the other hand, the highest mAP50s achieved by the medium model were 29.6, 47.5 and 53.6 with $N = 100, 10, 25$, respectively; the 5- and 10-shots were worse though than not using any generated data. From the data presented, the results of the best-performing student of each variant for each few-shot setting are presented in Figure 4.1 together with the teacher's best results.

The most significant improvement seen in Table 4.3 compared to not using generated data was 17.4 and was achieved by the small model on 1-shot with 50 generated instances. Furthermore, the worst change was the medium model in 5-shot with 100 generated instances, dropping 10.6 mAP50.

Furthermore, Table 4.4 shows the Precision (P), Recall (R) and mAP50 scores for each class, for each of the best-performing distilled students of Table 4.3. The highest P scores among all k-shots can be seen for the michelin figure and warning light classes, while the highest R scores belong to the front attachment class. The highest mAP50 score also belongs to the front attachment class, over all k-shots.

Table 4.2: The 1/5/10-shot mAP50 results for every YOLOv11 version on the cropped dataset. Bold values highlight the best YOLOv11 size for each k-shot, while the highlighted value in blue shows the best performing combination overall.

	YOLO	Cropped
1-shot	YOLOv11n	6.51
	YOLOv11s	13.2
	YOLOv11m	20.4
	YOLOv11l	20.6
	YOLOv11x	**29.1**
5-shot	YOLOv11n	31.6
	YOLOv11s	38.3
	YOLOv11m	47.7
	YOLOv11l	**50.1**
	YOLOv11x	45.6
10-shot	YOLOv11n	48.0
	YOLOv11s	54.3
	YOLOv11m	55.0
	YOLOv11l	**57.2**
	YOLOv11x	52.4

4.4 Inference times

The inference times recorded from CD-ViTO with the ViT-B/14 as well as the nano, small and medium variants of YOLOv11, when benchmarked with an Nvidia RTX 4000 Ada Generation GPU, are presented in Table 4.5. The table shows CD-ViTO having a much longer inference time than the YOLOv11 models, with CD-ViTO's being 560 ms while the medium, small and nano variants of YOLOv11 having inference times of 4.6, 2.9 and 2.8 ms, respectively. The truck detector, which is a YOLOv11n model, took 3.9 ms per image.

Comparing the inference times of the models, the nano seems to be the fastest, only 0.1 ms faster than the medium model, 1.6 ms faster than the medium and all the YOLOv11 models being more than 500 ms faster than CD-ViTO. Combining the YOLOv11 models for equipment detection with the truck detector gives 6.7, 6.8 and 8.5 ms, respectively, for the nano, small and medium models. The combined system can then manage 149, 147 and 117 images per second, assuming there are no other delays in the system. The teacher, on the contrary, would only manage around 2. Comparing the teachers inference time to the best student from the result in Section 4.3, which would be the small one, the teacher is $(560+3.9)/(2.9+3.9) = 563.9/6.8 \approx 83$ times slower than YOLOv11s.

Table 4.3: The 1/5/10-shot mAP50 distilled results for every YOLOv11 version on the cropped dataset with N added generated class instances, where N ranges from 10 up to 100. The ViT-B/14 backbone in CD-ViTO is chosen as a teacher for all k-shots variations. Bold values highlight the best YOLOv11 size for each k-shot, while the highlighted values in blue show the best k-shot combinations.

	YOLO	Cropped 1-shot	Cropped 5-shot	Cropped 10-shot
N=10	YOLOv11n	12.7 (+6.19)	39.1 (+7.5)	50.7 (+2.7)
	YOLOv11s	18.0 (+4.8)	**53.8** (+15.5)	**57.6** (+3.3)
	YOLOv11m	**18.2** (-2.2)	47.5 (-0.2)	53.2 (-1.8)
N=25	YOLOv11n	23.1 (+16.6)	41.9 (+10.3)	46.4 (-1.6)
	YOLOv11s	**26.6** (+13.4)	**52.8** (+14.5)	52.9 (-1.4)
	YOLOv11m	18.7 (-1.7)	45.0 (-2.7)	**53.6** (-1.4)
N=50	YOLOv11n	12.0 (+5.49)	37.7 (+6.1)	44.5 (-3.5)
	YOLOv11s	**30.6** (+17.4)	**49.5** (+11.2)	**55.8** (+1.5)
	YOLOv11m	19.4 (-1.0)	44.4 (-3.3)	53.2 (-1.8)
N=100	YOLOv11n	16.0 (+9.49)	35.0 (+3.4)	47.6 (-0.4)
	YOLOv11s	24.6 (+11.4)	**43.4** (+5.1)	**54.5** (+0.2)
	YOLOv11m	**29.6** (+9.2)	37.1 (-10.6)	49.5 (-5.5)

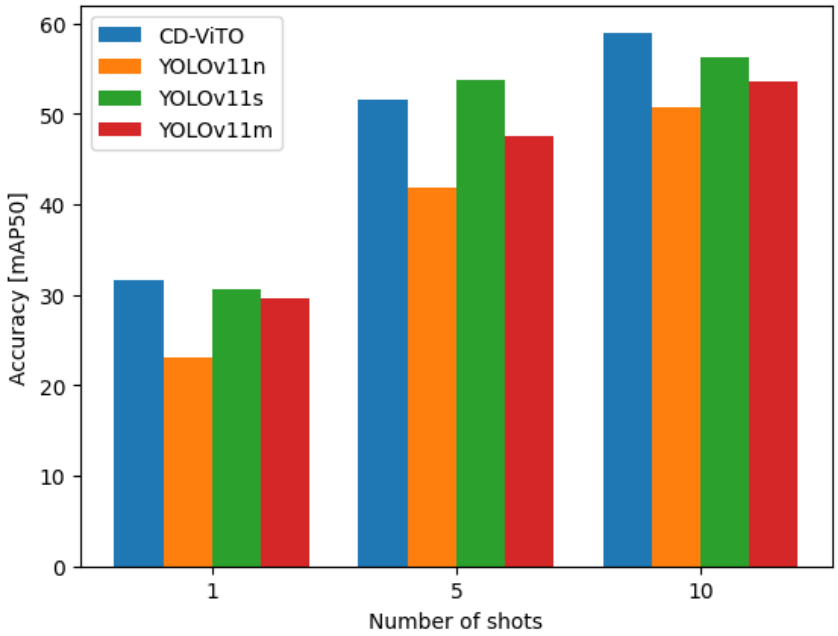


Figure 4.1: The bar graph shows the teacher model compared to the best performing students for each model variant and shot count based on the results in Table 4.3.

Table 4.4: The P/R/mAP50 scores for the best performing distilled YOLOv11 student, for every individual class. Bold values highlight the mAP50 scores for all classes in each k-shot, while the highlighted values in blue show the highest mAP50 scores per class for each k-shot.

Class	**Instances**	**1-shot**	**5-shot**	**10-shot**
all	357	65.6/16.1/**30.6**	76.6/30.2/**53.8**	86.3/36/**57.6**
flat-light-bar	13	100/0/28.5	45.6/23.1/42.8	58.9/30.8/30.7
front-attachment	5	79.6/78.2/78.2	77.7/100/99.5	83.7/100/99.5
front-fence	6	39.3/33.3/47	82.6/79.4/94.8	89.3/100/99.5
horn	50	45/2/2.57	100/16.2/50	100/15.3/34
light-1	143	68/2.99/30.6	100/4.5/61.8	100/1.38/44.9
light-2	26	53.2/7.69/24.9	46.7/3.85/16.4	96.2/19.2/60.1
light-strip	12	67.6/16.7/20.1	63.8/25/43.6	67/25/33.7
michelin-figure	16	100/0/18.5	100/19.9/49.5	100/37.1/90.5
roof-fairing	37	58/4/11.7	83.2/26.9/43.5	100/22.5/45.5
warning-bar	13	10.8/7.69/5.41	43.1/7.69/29.2	53.9/27.2/36.8
warning-light	36	100/24/68.9	100/25.9/60.5	100/17.9/58.3

Table 4.5: The table shows the average inference times over 50 images for the teacher with the ViT-B/14 backbone, the different students, as well as the YOLOv11n model used for the truck detection when run on an Nvidia RTX 4000 Ada Generation GPU.

Model	**Inference time**
CD-ViTO	560 ms
YOLOv11m	4.6 ms
YOLOv11s	2.9 ms
YOLOv11n	2.8 ms
Truck detector	3.9 ms

5

Discussion

In this chapter, the project's results are discussed, possible uncertainties connected to the method and results as well as possible ways it could be improved. Concerning the project's purpose to enable real-time object detection domains with data scarcity problems, the different parts of the system will be examined critically, weighing the positive and negative aspects of the solution against each other. The first part of this chapter, Section 5.1, discusses the results previously presented and attempts to explain the findings. The second part, Section 5.2, presents how this work can possibly be improved in future work. Lastly, in Section 5.3, the conclusion of this thesis is presented.

5.1 Result discussion

From the test performed on CD-ViTO to compare the detection capability of the models on full images with images of only the truck, there was a significant improvement for all model variants, doubling the mAP50 on several occasions. This is likely because the detection model resizes the images to a fixed size and the loss in detail will be larger for larger changes in size. The experiment also shows that the ViT-B/14 variant performs the best on the cropped images, which might be the case because of ViT-S/14 potentially not fitting the data well enough and ViT-L/14 not generalizing well from the training data.

Comparing the results of CD-ViTO to those of the YOLOv11 models on only real images shows CD-ViTO having an mAP50 of 1 to 3 points greater than the best YOLOv11 models in the corresponding k-shots. Shifting the focus from the different architectures to only looking at the different YOLOv11 models, the largest model, YOLOv11x, only performed the best in 1-shot, while being surpassed by second largest, YOLOv11l, in 5- and 10-shot. The cause of this might be the same as with CD-ViTO, that the smaller models do not quite fit the data, while the largest models does not generalize well from the training data.

The distillation procedure appears to work the best during 1-shot and 5-shot, more than doubling the mAP50 on a few of the 1-shot tests, with the 5-shot increasing by similar amounts, but having a higher baseline. The 10-shot, on the other hand, only saw improvements on a few of the tests, with the small model being the one seeing all the improvements. The reason for the 10-shot not improving much, if anything, could be because the models have enough real data to fit it well, while

the annotations on the generated data might be lacking, or the data is not similar enough to the real data for the model to learn anything useful from it. Similarly, the medium model does not seem to improve from the generated images. In fact, in almost all of the tests done on the distilled models, the medium model improved the least compared to the baseline, which probably stems from the larger models fitting the real data better, as previously mentioned. Contrary to the medium model, the small model performed the best on the distilled test, probably because it fits the data better than the nano and medium models, being able to learn valuable details from the generated data.

The P, R and mAP50 scores in Table 4.4 offer some valuable insights into the performance of the distilled students. Across all classes, the trend seems to be that increasing training examples generally leads to improved performance on all three metrics. Regarding the performance across individual classes, some interesting insights can be gathered. The standout performer class is the front-attachment class, achieving an impressive 78.2 mAP50 score even in the 1-shot scenario. It reaches the near-perfect mAP50 at 99.5 and perfect recall at 100 in both 5- and 10-shot settings. This suggests that the front attachment class is highly distinctive and is overall consistent with its visual features, something that is easily learned with minimal data examples. The small number of instances (5) might also imply that these are visually similar. Likewise, the front fence class also shows excellent improvement, indicating that its features of being generally larger in size are also highly learnable.

On the contrary, some examples of challenging classes include the flat light bar, the michelin figure, and the warning light. For the flat light bar and the michelin figure, a P score of 100 and R score of 0 in the 1-shot scenario indicates that the model makes no positive predictions (because it makes no predictions at all or incorrect ones that are suppressed by the confidence threshold). This would suggest that the model is overly cautious, given the mAP50 values of 28.5 and 18.5, respectively. On the other hand, the warning light maintains a P score of 100 across all k-shots. However, its R score is moderate at best and even slightly decreases in the 10-shot scenario, causing the mAP50 score to fluctuate. This again points to a model that is becoming too selective. Overall, many classes achieve high precision, especially with more shots. However, that is often coupled with low or moderate recall, indicating that while the detections are likely correct, the model fails to detect many instances of that class. At last, it is worth discussing that the mAP50 scores provide a balanced view of the results. For instance, while the horn and first light class score 100 in P in 10-shot, their mAP50 scores (34 and 44.9) are not among the highest due to their low R scores at 15.3 and 1.38. Again, this underscores the importance of considering both P and R, which mAP50 encapsulates. Thus, the drop in mAP50 for these classes when going from 5- to 10-shot, despite improved P, is a key finding, suggesting that there might be potential problems of generalization to some features.

Given the inference time of the models and the theoretical system presented, using the same hardware as in the tests, the systems would easily manage images from a typical video stream of 30 or 60 FPS. With the inference times of the small and nano models being as close as they are, the small would probably be the best choice for real-world applications because of its speed and higher mAP50. If instead, the

models are deployed on a system where it is important to save every tenth of a millisecond, the nano might be the best choice.

While the two smaller, distilled models saw the largest improvements compared to just fine-tuning the models on real data, the teacher's mAP50 was only surpassed twice, by the small model in 5-shot with 10 and 25 generated instances. Due to the improvements seen on several of the tests and even surpassing the teacher, the project is deemed to have met its goal.

5.2 Future work

Although the results are considered positive, improvements can still be made to the increased speed and mAP50. Some areas of improvement will be presented together with possible solutions.

One such area is the method used to generate images. The current one involves a lot of manual labor in the choice of which prompts to use, which can be avoided by using a method for automatically creating prompts or using another method altogether for the image generation, for instance something like Textual Inversion [10], DreamBooth [12] or TINT [11]. All three methods take input images and generate similar ones to them given a simple prompt, which enables automating more of the work.

Another area deals with the detection models. The hyperparameters used in the project for the detection models were the ones that come standard with the models. This makes using the models simpler at the risk of potentially not being optimal for the task at hand. This could be addressed by doing hyperparameter optimization, by a grid search for example, which has the possibility of finding a set of parameters that would make the models fit the data better.

One more area where things could have been done differently is the creation of the k-shot splits. The method used was the one that CD-ViTO uses in its training. It splits the data perfectly, but it was discovered during the project that it only uses one or two annotations per image, even though there might be more. This is not a problem for CD-ViTO because it crops the boundary box and only uses the features in the cropped region. It might be a problem for other architectures, like YOLOv11, if it detects an object in the image that the splitting script did not include in the annotation, because the model might be confused when one instance of an object is annotated, but another one is not, leading to suboptimal training. A possible solution to this could be to occlude or blur the contents of some bounding boxes to ensure that the YOLOv11 model does not find any objects that are not properly annotated. Another way might be to, for a k-shot split, instead of strictly having k instances of each class, to have a minimum of k instances.

5.3 Conclusion

The project aimed to create a real-time object detection system for detecting truck equipment. To accomplish this, a method was developed, utilizing knowledge distillation together with generated images to transfer the knowledge of a SOTA FSOD model to a real-time object detection model. The resulting system was evaluated by performing several experiments, which indicate that the best model increases the detection accuracy somewhat compared to the teacher and speeds up detections, allowing inference times 83 times faster than the teacher. The results also show that the generated images helped the most in 1- and 5-shot and for the nano and small variants of the student, suggesting that having 10-shots per class instance is enough for the models to collect enough details to identify the objects. Ultimately, the project is deemed to have succeeded in accomplishing the goals set, providing a model that is both faster and more accurate than its teacher.

Bibliography

[1] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[2] R. Khanam and M. Hussain, "Yolov11: An overview of the key architectural enhancements," *arXiv preprint arXiv:2410.17725*, 2024.

[3] V. Chudasama, H. Sarkar, P. Wasnik, V. N. Balasubramanian, and J. Kalla, *Beyond few-shot object detection: A detailed survey*, 2024. arXiv: 2408.14249 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2408.14249.

[4] M. Köhler, M. Eisenbach, and H.-M. Gross, *Few-shot object detection: A comprehensive survey*, 2022. arXiv: 2112.11699 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2112.11699.

[5] X. Zhang, Y. Liu, Y. Wang, and A. Boularias, *Detect everything with few examples*, 2024. arXiv: 2309.12969 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2309.12969.

[6] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge Distillation: A Survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021, ISSN: 0920-5691. DOI: 10.1007/s11263-021-01453-z.

[7] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2022, pp. 10 684–10 695.

[8] C. Saharia, W. Chan, S. Saxena, *et al.*, "Photorealistic text-to-image diffusion models with deep language understanding," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 36 479–36 494. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/ec795aeadae0b7d230fa35cbaf04c041-Paper-Conference.pdf.

[9] T. Zhang, Y. Zhuang, X. Zhang, G. Wang, H. Chen, and F. Bi, "Advancing controllable diffusion model for few-shot object detection in optical remote sensing imagery," in *IGARSS 2024 - 2024 IEEE International Geoscience and Remote Sensing Symposium*, 2024, pp. 7600–7603. DOI: 10.1109/IGARSS53475.2024.10642625.

[10] R. Gal, Y. Alaluf, Y. Atzmon, *et al.*, *An image is worth one word: Personalizing text-to-image generation using textual inversion*, 2022. arXiv: 2208.01618 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2208.01618.

[11] E. Landolsi and F. Kahl, *Tiny models from tiny data: Textual and null-text inversion for few-shot distillation*, 2024. arXiv: 2406.03146 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2406.03146.

[12] N. Ruiz, Y. Li, V. Jampani, Y. Pritch, M. Rubinstein, and K. Aberman, *Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation*, 2023. arXiv: 2208.12242 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2208.12242.

[13] Y. Fu, Y. Wang, Y. Pan, *et al.*, *Cross-domain few-shot object detection via enhanced open-set object detector*, 2024. arXiv: 2402.03094 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2402.03094.

[14] G. Jocher and J. Qiu, *Ultralytics yolo11*, version 11.0.0, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics.

[15] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM Comput. Surv.*, vol. 53, no. 3, Jun. 2020, ISSN: 0360-0300. DOI: 10.1145/3386252. [Online]. Available: https://doi.org/10.1145/3386252.

[16] Z. Xin, S. Chen, T. Wu, Y. Shao, W. Ding, and X. You, "Few-shot object detection: Research advances and challenges," *Information Fusion*, vol. 107, p. 102 307, 2024, ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2024.102307. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S156625352400085X.

[17] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[18] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[19] O. Russakovsky, J. Deng, H. Su, *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, pp. 211–252, 2015.

[20] P. Hidayatullah, N. Syakrani, M. R. Sholahuddin, T. Gelar, and R. Tubagus, "Yolov8 to yolo11: A comprehensive architecture in-depth comparative review," *arXiv preprint arXiv:2501.13400*, 2025.

[21] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[22] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1706.03762.

[23] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, *Microsoft coco: Common objects in context*, 2015. arXiv: 1405.0312 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1405.0312.

[24] G. Drange, *Arthropod taxonomy orders object detection dataset*, 2020. DOI: 10.34740/KAGGLE/DSV/1240192. [Online]. Available: https://www.kaggle.com/dsv/1240192.

[25] N. Inoue, R. Furuta, T. Yamasaki, and K. Aizawa, "Cross-domain weakly-supervised object detection through progressive domain adaptation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 5001–5009.

[26] K. Li, G. Wan, G. Cheng, L. Meng, and J. Han, "Object detection in optical remote sensing images: A survey and a new benchmark," *ISPRS journal of photogrammetry and remote sensing*, vol. 159, pp. 296–307, 2020.

[27] A. Saleh, I. H. Laradji, D. A. Konovalov, M. Bradley, D. Vazquez, and M. Sheaves, "A realistic fish-habitat dataset to evaluate algorithms for underwater visual analysis," *Scientific reports*, vol. 10, no. 1, p. 14 671, 2020.

[28] K. Song and Y. Yan, "A noise robust method based on completed local binary patterns for hot-rolled steel strip surface defects," *Applied Surface Science*, vol. 285, pp. 858–864, 2013.

[29] L. Jiang, Y. Wang, Q. Jia, *et al.*, "Underwater species detection using channel sharpening attention," in *Proceedings of the 29th ACM International Conference on Multimedia*, 2021, pp. 4259–4267.

[30] M. Oquab, T. Darcet, T. Moutakanni, *et al.*, "Dinov2: Learning robust visual features without supervision," *arXiv preprint arXiv:2304.07193*, 2023.

[31] H. Cao, C. Tan, Z. Gao, *et al.*, "A survey on generative diffusion models," *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 2814–2830, 2024. DOI: 10.1109/TKDE.2024.3361474.

[32] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 6840–6851. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/4c5bcfec8584af0d967f1ab10179ca4b-Paper.pdf.

[33] W.-L. Chiang, L. Zheng, Y. Sheng, *et al.*, *Chatbot arena: An open platform for evaluating llms by human preference*, 2024. arXiv: 2403.04132 [cs.AI].

[34] P. Esser, S. Kulal, A. Blattmann, *et al.*, "Scaling rectified flow transformers for high-resolution image synthesis," in *Forty-first International Conference on Machine Learning*, 2024. [Online]. Available: https://openreview.net/forum?id=FPnUhsQJ5B.

[35] P. Kaur, B. S. Khehra, and E. B. S. Mavi, "Data augmentation for object detection: A review," in *2021 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2021, pp. 537–543. DOI: 10.1109/MWSCAS47672.2021.9531849.

[36] B. Dwyer, J. Nelson, T. Hansen, and et al., *Roboflow (version 1.0)*, Available from https://roboflow.com, Computer Vision Software, 2024.